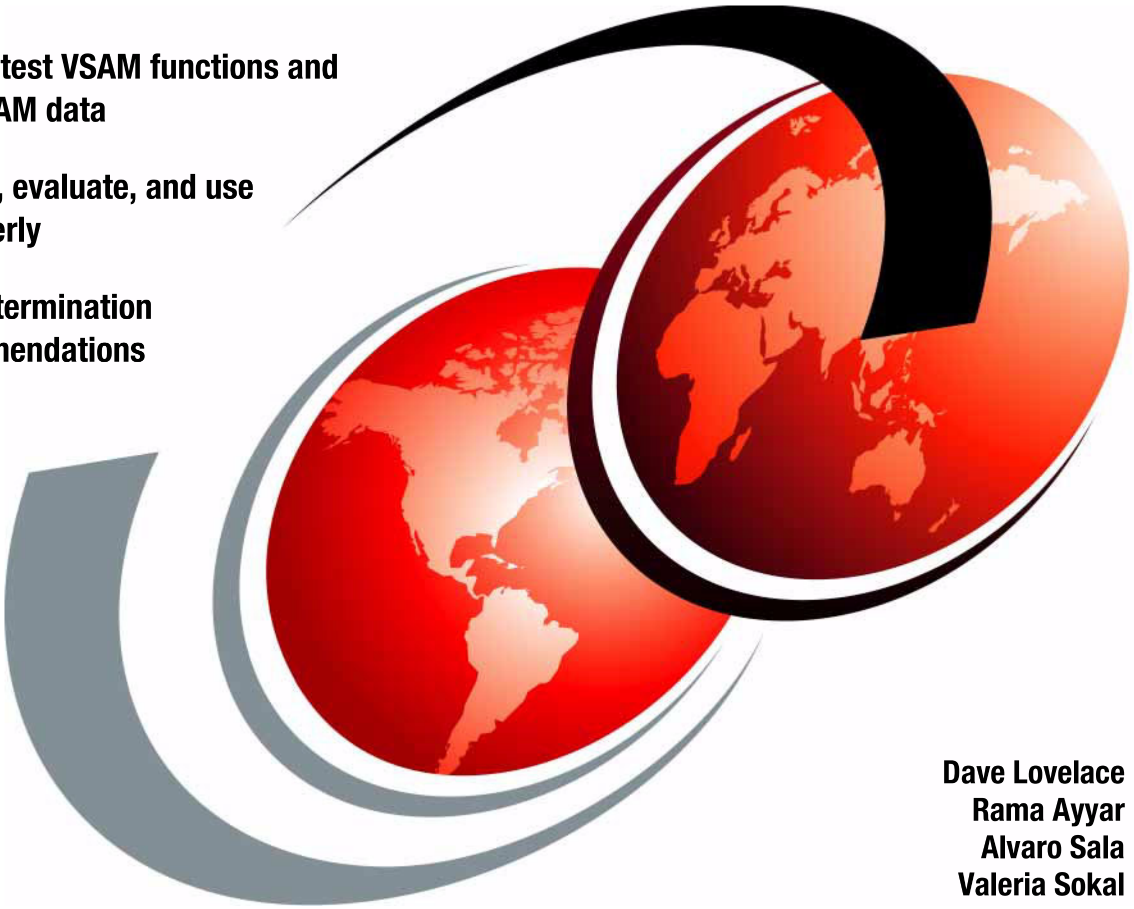
IBM

VSAM Demystified



- Learn the latest VSAM functions and manage VSAM data
- Understand, evaluate, and use VSAM properly
- Problem determination and recommendations

Dave Lovelace
Rama Ayyar
Alvaro Sala
Valeria Sokal

Redbooks

ibm.com/redbooks

International Technical Support Organization

VSAM Demystified

September 2003

SG24-6105-01

Note: Before using this information and the product it supports, read the information in "Notices" on page xix.

Second Edition (September 2003)

This edition applies to Version 1, Release 4 of z/OS (product number 5694-A01) and z/OS V1 DFSMS Transactional VSAM Services (feature number 6330).

Contents

Figures

Tables

Notices

Trademarks

Preface

Virtual Storage Access Method (VSAM) is one of the access methods used to process data. Many of us have used VSAM and work with VSAM data sets daily, but exactly how it works and why we use it instead of another access method is a mystery.

This book helps to demystify VSAM and gives you the information necessary to understand, evaluate, and use VSAM properly. It clarifies VSAM functions for application programmers who work with VSAM. The practical, straightforward approach should dispel much of the complexity associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This IBM® Redbook is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions.

This book also builds upon the subject of Record Level Sharing and the new z/OS® feature called DFSMStvs.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Dave Lovelace is a Project Leader at the International Technical Support Organization, San Jose Center. Before joining the ITSO this year, Dave worked in the IBM Storage Systems Group as Program Director of Business Development and Strategic Alliances. With more than 30 years in the IT industry, Dave has held positions as an IBM Systems Engineer, an MVS™ Systems Programmer, a Marketing Support Representative in OS/390® marketing, Software Packaging (CBIPO) and several years working in client/server environments with the US Military Academy at West Point.

Rama Ayyar is a Senior IT Specialist with the IBM Support Center in Sydney, Australia. Rama has over 20 years of experience with the MVS operating system and has been in the Computer industry for over 30 years. His areas of expertise include TCP/IP, RACF®, DFSMS, configuration management, dump analysis and disaster recovery. Rama holds a master’s degree in Computer Science from the Indian Institute of Technology, Kanpur.

Alvaro Sala is an IBM retiree. He worked in IBM for more than 30 years, always in large systems. Alvaro co-authored many Redbooks and spent many years teaching, from S/360 to S/390®. He has a Chemistry Engineer degree from the University of Sao Paulo, Brazil.

Valeria Sokal is a Business Partner from Brazil. She has 12 years experience as an OS/390 System Programmer. She co-authored these Redbooks: *OS/390 Workload Manager Exploitation and Implementation* and *ABC's for OS/390 System Programmers*.

Thanks to the following people for their invaluable contributions to this project:

Mary Lovelace
ITSO, San Jose

Savur Rao
Terri Menendez
Helen Witter
Ruth Ferziger
James Becker
Storage Systems Group, San Jose

Charlie Burger
Advanced Technical Systems Center, San Jose

Bob Haimowitz
International Technical Support Organization, Raleigh

Paul Rogers
International Technical Support Organization, Poughkeepsie

Become a published author

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  `ibm.com/redbooks`

- Send your comments in an Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

VSAM basics

This chapter reviews the concepts and terminology associated with VSAM, explains how a VSAM data set is different from other data set types, discusses the various types of VSAM data sets, and what makes them unique, and describes how VSAM data is stored and accessed.

1.1 A brief description of VSAM

In the early 1970s, VSAM (Virtual Sequential Access Method) was introduced by IBM as a collection of three data set organizations — sequential, indexed, and direct-access, together with the access methods and utilities to be used on the large scale IBM operating systems.

The word *virtual* means only that VSAM was introduced at approximately the same time as the initial IBM virtual storage operating systems OS/VS1 and OS/VS2. Since then VSAM has been continually improved and enhanced.

1.2 VSAM functions by release level

Some recent improvements to VSAM are extended format and extended addressability which allows a data set to go beyond the 4 GB limitation, system managed buffering (SMB) for improved performance, data compression, data striping, record level sharing (RLS), and most recently, transactional VSAM. Table 1-1 shows the recent VSAM enhancements by DFSMS release level.

Note: DFSMS/MVS® V1R5 was the last independent release. With z/OS V1R3, DFSMS became an integral component of z/OS. This book is written at the z/OS V1R4 DFSMS level.

Table 1-1 VSAM functions by release level

Function	Available since	Section described in
Data compression for VSAM KSDS	DFSMS/MVS V1 R2	"Data compression" on page 94
Extended addressability for VSAM KSDS	DFSMS/MVS V1.R3	"Extended addressability (EA)" on page 228
VSAM RLS	DFSMS/MVS V1.R3	Chapter 5, "VSAM Record Level Sharing" on page 243
RLS support of extended addressabiliy of VSAM KSDS	DFSMS/MVS V1 R4	Chapter 5, "VSAM Record Level Sharing" on page 243
SMB for VSAM KSDS	DFSMS/MVS V1 R4	"System managed buffering (SMB)" on page 82
Data striping for VSAM LDS through an SPE	DFSMS/MVS V1 R5	"VSAM Data striping" on page 103

Function	Available since	Section described in
Extended addressability for all VSAM data sets	DFSMS/MVS V1 R5	“Extended addressability (EA)” on page 228
Data striping and multi-layering for all VSAM data set	DFSMS/MVS V2 R10	“VSAM Data striping” on page 103
Ignore DEFINE parameters IMBED, REPLICATE	z/OS V1R3 DFSMS	“Index options” on page 58
Ignore DEFINE parameters KEYRANGE, ORDERED	z/OS V1R3 DFSMS	“Key Range and Ordered” on page 58
SMB retry capability for DO access bias	z/OS V1R3 DFSMS	“System managed buffering (SMB)” on page 82
SMB support for AIX®	z/OS V1R3 DFSMS	“SMB support for AIX” on page 87
RLS cache CI size bigger than 4KB	z/OS V1R3 DFSMS	“RLS CF caching enhancements” on page 280
VSAM data striping support for data sets with REUSE attribute	z/OS V1R3 DFSMS	“VSAM data striping topics” on page 103
RLS caching all or some of the data in a Coupling Facility cache structure	z/OS V1R3 DFSMS	“RLS CF caching enhancements” on page 280
Maximum Volume Count, vols added dynamically to a data set without manual intervention and taking down the application	DFSMS z/OS V1R3	“Implementing VSAM data striping” on page 106
DFSMS Data Set Separation, to allocate data sets in distinct physical DASD controllers	DFSMS z/OS V1R3	“I/O wait time (IOSQ) for VSAM data sets” on page 125
Real addresses greater than 2 GB available for all VSAM data sets	z/OS V1R4 DFSMS	“VSAM and 64 bits” on page 240
Media Manager the unique VSAM I/O driver, exception for Improved Control Interval Processing (ICIP)	DFSMS z/OS V1R4 v	“What is VSAM?” on page 4
RLS system-managed duplexing rebuild process	z/OS V1R4 DFSMS	“System-managed duplexing rebuild” on page 279
RLS Coupling Facility duplexing	z/OS V1R4 DFSMS	“System-managed duplexing rebuild” on page 279

Function	Available since	Section described in
Dynamic Cache Reassignment	z/OS V1R3 DFSMS	"RLS CF caching enhancements" on page 280

1.3 What is VSAM?

VSAM is one of several access methods in z/OS. It only applies to data stored in DASD devices. An access method is re-entrant code contained in DFSMSdfp™, a component of the DFSMS z/OS product. This access method makes it easier for an application to execute an I/O operation (moving data between an I/O device and memory).

1.3.1 VSAM access types

There are three types of access in VSAM:

- Random: This is also referred to as direct access. The logical record needs to be located by the use of a search argument coming from the application. There is no search argument connection between two consecutive logical record accesses.
- Sequential: The entire file is processed (for Read or Write), one logical record after the other. The application does not need to provide any search argument. The access method can implement a Read look ahead technique to load logical records in the buffers not yet required by the application program.
- Skip sequential: A combination of the two previous types of access. The application randomly provides one search argument and from the located logical record on, all records are processed sequentially. An example is sequentially processing all the customers of a bank branch office in a file that has all the customers of the bank.

1.4 Major VSAM parts

There are two major parts to VSAM, catalog management and record management.

1.4.1 Catalog management

VSAM maintains extensive information about data sets and direct access storage space in an integrated catalog facility (ICF) catalog. The catalog's collection of information about a data set defines that data set's characteristics. All VSAM files must be defined in an ICF catalog.

In this book, we will not go into detail on Catalog Management. For more information, refer to the IBM Redbooks *Enhanced Catalog Sharing and Management*, SG24-5594, and *Integrated Catalog Facility Backup and Recovery*, SG24-5644.

1.4.2 Record management

The record management part of VSAM contains the access method code. In this book, when we say VSAM we mean VSAM record management, unless the opposite is stated. VSAM is used to organize records into four types of data sets: key-sequenced, entry-sequenced, linear, or relative record. The primary difference between the types of VSAM data sets is the way their records are stored and accessed.

1.5 VSAM terminology and concepts

Before we discuss VSAM in detail, we need to review some VSAM concepts implemented by VSAM constructs. These concepts will be used throughout the book. We are going to start with the smallest VSAM entity, the logical record, to the largest VSAM entity, the sphere. After that we cover splits, buffering, control blocks and other constructs.

1.5.1 Logical record

A logical record is a unit of information used to store data in a VSAM data set. It is made up of a set of bytes containing a logical description of an item processed by an application program. This item can be a customer with all his or her information, or an employee with all associated data (name, serial number, department).

The logical record is designed by the application programmer from the business model. The logical record is divided in fields, such as: the name of the item, its key, the address, and the account information. The application through a GET request that a specific logical record be moved from the I/O device to memory in order to be processed. Through a PUT the specific logical record is moved from memory to an I/O device.

A logical record can be of a fixed size or a variable size depending on the business requirements. VSAM supports both depending on the specific organization. An example of a variable logical record could be the account information about a bank customer where all debit and credit transactions are individually described at the end of the register.

1.5.2 Key field

An important field in the logical record is the *key.* Its contents can be used to retrieve the specific logical record. It identifies the item associated with the logical record. Keys can be the customer number or the parts number.

To differentiate between keys used in VSAM objects, the key used in a base cluster is called a primary key or base key. The key used in an alternate index is called an alternate key.

In VSAM key sequenced organization, a record must have a unique, imbedded fixed-length primary key located in the same position within each logical record. Primary keys can be a minimum of one byte and a maximum of 255 bytes.

There can be multiple key fields in the same logical record, called alternate or secondary keys. Unlike the primary keys, which must be unique, identical alternate may occur in more than one logical record. This allows the search with a given alternate key to read all base cluster records containing this alternate key.

1.5.3 Ways to identify logical records

In VSAM there are three ways to identify a logical record:

- Key field
- Relative byte address (RBA)

  The RBA of a logical record is the offset of its first byte from the beginning of the file.

  Then, the first logical record in a VSAM file has an RBA of zero; the second logical record has an RBA equal to the length of the first record, and so on. VSAM returns to the application the RBA of the stored logical record. The RBA of a logical record in VSAM includes the free space and control information stored before this logical record.

  RBAs might change when records are added, deleted, or changed in size. With compressed files, the RBAs for compressed records are not predictable. Therefore, access by RBA is not suggested for normal use.

  There are two important RBA values, kept in the Catalog:

  – High used RBA (HURBA) is the RBA of the first available byte for inclusion at end of the VSAM file.
  – High allocated RBA (HARBA) is the RBA of the last byte in the VSAM file

- Relative Record Number (RRN)

  It is the relative number of a logical record in a VSAM specific organization (RRDS). If a logical record has an RRN of 3 means that it is the fourth logical record in the file.

  The terms logical record and record are used interchangeably in this book

1.5.4 Physical record

A physical record is device dependent, its size is calculated at the time the data set is defined. All physical records have the same length. A physical record is also called a physical block or simply a block.

To understand what a physical record on DASD is, we need to review the 3390/3380 count key data (CKD) track format. Every track starts with an index marker, a gap, a home address (describing the cylinder and track number) and a gap. After that, we have the physical records of the track. Refer to Figure 1-1.

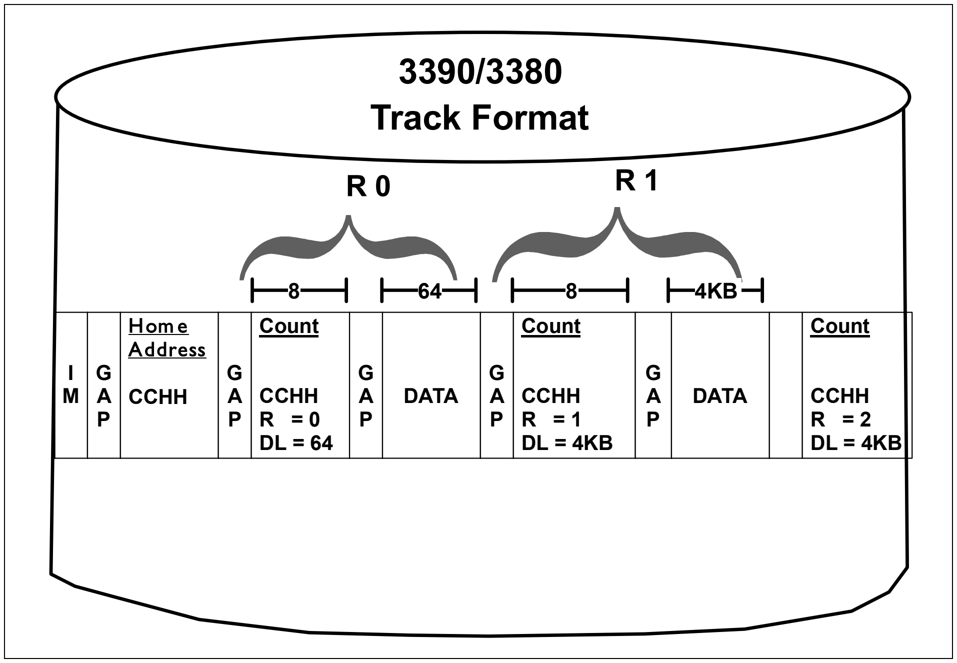


Figure 1-1 3390/3380 Track Format

A physical record is formed by a count, a gap, an optional key, a gap and the data. The key field is only present in non-indexed VTOC and partitioned data sets (PDS) directories. The count part indicates the number of the physical record in the track and its data length, which could be variable. The data is usually a set of logical records (packed together) transferred from or to memory by just one Read or Write CCW. The access method uses the BLKSIZE parameter to determine the length of the physical record. A large block size results in fewer gaps in the track (track capacity is better used) but larger buffer in memory to keep the data.

Today actual 3390/3380 devices are becoming rare. The new controllers logically mimic fix block architecture (FBA) disks. You may ask the question “Do I care about the BLKSIZE to optimize space, if the 3390/3380 track does not really exist?” The answer is yes. The controller simulates the 3390/3380 track. It knows for an specific BLKSIZE how many physical records fit on the 3390/3380 track. So remember the rule-of-thumb (ROT) of a half track BLKSIZE (for sequential processing) to better utilize the 3390/3380 logical track capacity.

There are two techniques for an access method create physical records:

- First format it with zeroes and later fulfill the data portion with real data. This technique is used for random data creation or to create software end of files marks. These EOFs are data parts full of zeroes.
- Format and fulfill with real data at same time. This technique is used for sequential data creation.

VSAM may have control information along with logical records in the data portion of the physical record, as we see in the control interval VSAM concept.

1.5.5 Control interval

Control interval (CI) is a VSAM unique concept. A CI is formed by physical records, usually just one. It is the fundamental building block of every VSAM file. A CI is a contiguous area of direct access storage that VSAM uses to store data records and control information that describes the records. A CI is the unit of information that VSAM transfers between the storage device and the processor during one I/O operation. Whenever a record is retrieved from direct access storage, the entire CI containing the record is read into a VSAM I/O buffer in virtual storage. The desired record is transferred from the VSAM buffer to a user-defined buffer or work area.

Based on the CI size, VSAM calculates the best size of the physical block in order to better use the 3390/3380 logical track. The CI size can be from 512 byes to 32 KB.

The size of the physical record (block) is determined by VSAM based on the CI size, if the data set is in extended format or not (refer to 1.9, “Extended format data set” on page 28) and on the best use of space in the 3390/3380 track. For example in a 3390 without EF, with a CI Size of 22528 bytes, the block size is 5632 bytes; with a CI Size of 24576, the block size is 24576 bytes.

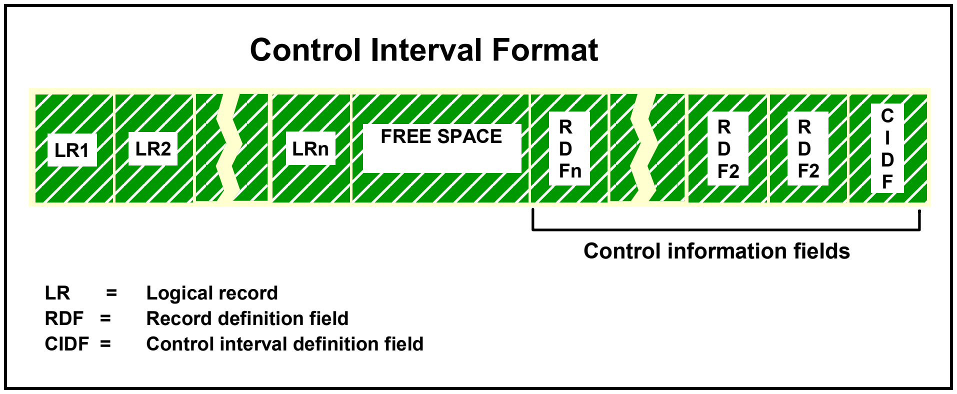
For random access, it is recommended to use small data CIs, to avoid bringing uneeded logical records into memory. For sequential access, it is recommended that you define large CIs to decrease the number of I/O operations and the number of Read/Write CCWs.

A CI consists of:

- Logical records stored from beginning to end
- Free space, for data records to be inserted into or lengthened
- Control information, which is made up of two types of fields; one control interval definition field (CIDF) per CI, and several record definition fields (RDF) describing the logical records.
  - CIDF is a 4-byte field.
  - It contains information about the amount and location of free space.
  - RDF is a 3-byte field.
  - It describes the length of records. For fixed length records there are two RDFs, one with the length and other with how many with the same length.

  For more information on the structure of the control information fields, refer to *DFSMS Using Data Sets*, SC26-7410.

Figure 1-2 shows the general format of a CI. The CI components and properties may vary depending on the data set organization. For example, an LDS does not contain CIDFs and RDFs in its CI. All of the bytes in the LDS CI are data bytes. Refer to 1.6, "VSAM data set organizations" on page 16, for more details.



Control Interval Format
LR1
LR2
LRn
FREE SPACE
R D Fn
R D F2
R D F2
C I D F
Control information fields
LR = Logical record
RDF = Record definition field
CIDF = Control interval definition field

Figure 1-2 General format of a control interval

The size of CIs can vary from one file to another, but all the CIs within the data component of a particular data set must be of the same length. Refer to 1.5.8, "Component" on page 10 for details on the data component.

You can request the CI size using the AMS DEFINE command, you can let VSAM determine the CI size, or you can specify an SMS data class, thereby using the CISIZE defined by your storage administrator.

1.5.6 Control area

Control area (CA) is also a VSAM unique concept. A CA is formed by two or more CIs put together into fixed-length contiguous areas of direct access storage. A VSAM data set is composed of one or more CAs. In most cases, a CA is the size of a 3390/3380 cylinder. The minimum size of a CA is one track. The CA size is implicitly defined when you specify the size of a data set at data set definition.

CAs are needed to implement the concept of splits. The size of a VSAM file is always a multiple of its CA. VSAM files are extended in units of CAs. A spanned record cannot be larger than a CA.

1.5.7 Spanned records

Spanned records are logical records that are larger than the CI size. To have spanned records, the file must be defined with the SPANNED attribute at the time it is created. Spanned records are allowed to extend across or span control interval boundaries. The RDFs describe whether the record is spanned or not.

A spanned record must always begin on a control interval boundary and fills one or more control intervals within a single control area. A spanned record cannot share the CI with any other records. In other words, the free space at the end of the last segment is not filled with the next record. This free space can only be used to extend the spanned record.

Spanned records are needed when the application requires very long logical records. A spanned record may be the data component of an AIX cluster. If spanned records are used for KSDS, the primary key must be within the first control interval. Refer to“Alternate indexes” on page 14.

1.5.8 Component

A component is an individual part of a VSAM data set. Each component has a name, an entry in the catalog and an entry in the VTOC. The KSDS and VRRDS organizations have data and index components. ESDS, RRDS and LDS organizations only have data components. A component can be multi-extent and multi-volume, however VSAM does not allow JCL concatenation between two VSAM components. You receive the IEC161I message with RC 68.

There are two types of components, the data component and the index component.

Data component

The data component is the part of a VSAM data set, alternate index, or catalog that contains the data records.

Index component

Using the index, VSAM is able to randomly retrieve a record from the data component when a request is made for a record with a certain key. The key determines the record's position in the data set. VSAM divides the index CI into sections in order to speed up the search of a key.

A VSAM index can consist of more than one level. Each level contains pointers to the next lower level.

The index component is a collection of logical records containing data keys and the address (RBA) of the data logical records containing these keys. These data keys are taken from a fixed defined field in each data logical record. The keys in the index logical records are compressed (rear and front). The RBA pointers are compacted.

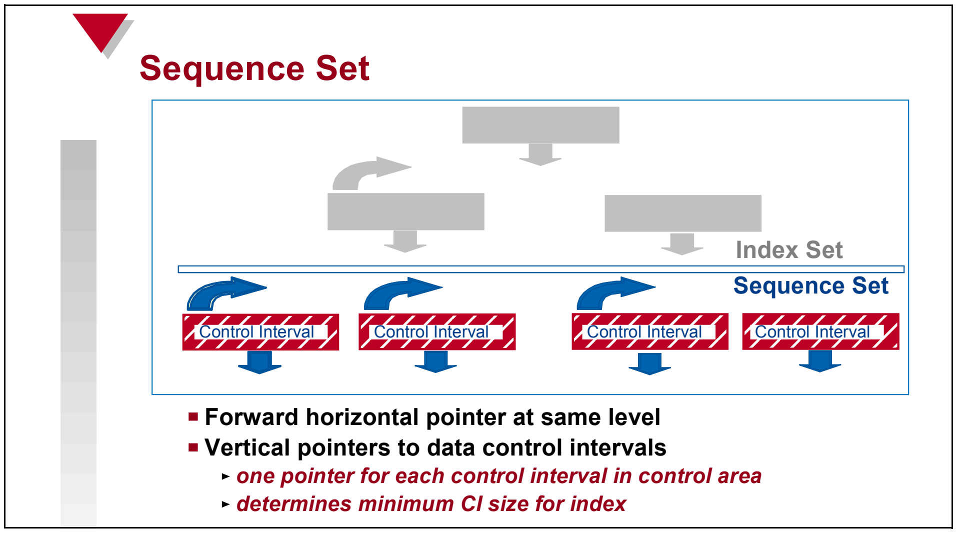
Using the index, VSAM is now able to retrieve a logical record from the data component when a request is made with a certain key. A VSAM index can consist of more than one level (balanced tree). Each level contains pointers to the next lower level. Because there are random and sequential types of access VSAM divides the index into two parts: sequence set and index set.

Sequence set

The sequence set is used for sequential access.

The sequence set is the lowest level of index CIs and directly points (through an RBA) to the data CI in the CA. There is one index entry per CI data and consequently one index CI for each data CA.

Every logical record contains pointers and high key information for each data CI. It also contains horizontal pointers from one sequence set CI to the next higher keyed sequence set CI (see Figure 1-3). These horizontal pointers are needed because the possibility of splits, which make the physical sequence different from the logical collating sequence.



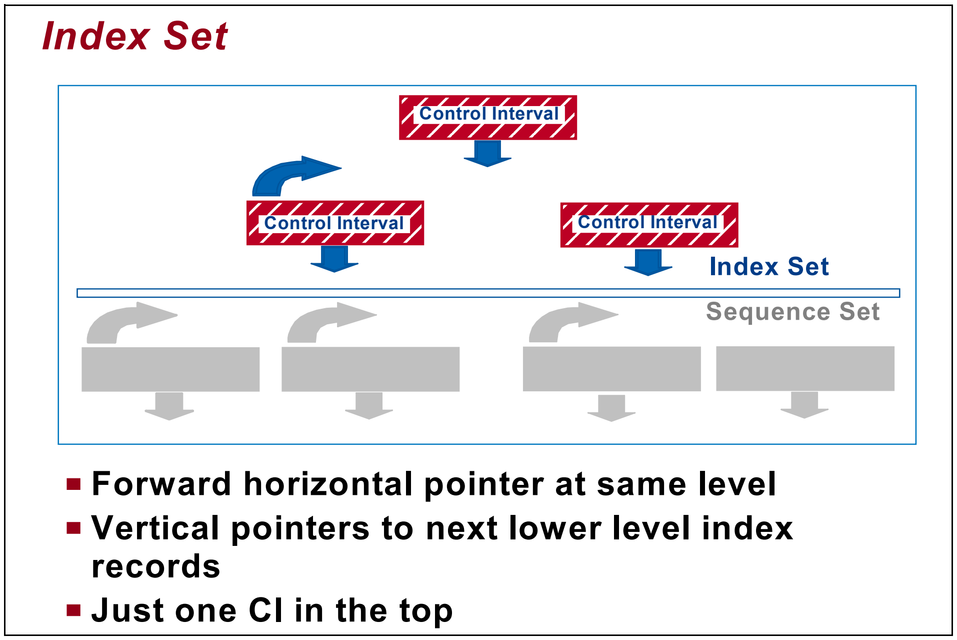
Sequence Set

- Forward horizontal pointer at same level
- Vertical pointers to data control intervals
  - *one pointer for each control interval in control area*
  - *determines minimum CI size for index*

Figure 1-3 General format of a sequence set

Index set

The index set is used for random access. The index set is the remainder of the index component. If there is more than one sequence set CI, VSAM automatically builds another index CI in other level. Each CI in the index set contains pointers and high key information for CIs in the next lower level of the index. See Figure 1-4.



Index Set

- Forward horizontal pointer at same level
- Vertical pointers to next lower level index records
- Just one CI in the top

Figure 1-4 General format of an index set

The highest level of the index always contains a single index CI.

1.5.9 Cluster

A cluster is a set of related components (maximum two). For a KSDS, a cluster is the set of a data component and an index component. The concept of cluster simplifies VSAM processing, providing a way to treat index and data components as a single entity, with its own catalogued name. You can also give each component a name. This allows you to process the data portion separately from the index portion.

RRDS, ESDS, and LDS formats are considered to be clusters without index components. To be consistent, they are given cluster names that are normally used when processing the data set.

Now comes the crucial VSAM question: “What in VSAM corresponds to an MVS data set?” The best answer is *it depends.* If you are referring to the cluster name (in a DD statement for example), then the cluster corresponds to the data set. If you are referring to a component, then it corresponds to the data set.

1.5.10 Sphere

A sphere is a base VSAM cluster and its associated clusters. These associated clusters are the alternate indexes (AIXs) of the base cluster. An AIX is a KSDS cluster containing index entries in the index component organized by the alternate keys of its associated base data records. In the AIX data component there is a set of primary keys associated with the referred secondary key. Then, the concept of an sphere provides another way of locating records (by using other keys) in the data component of a cluster.

An AIX can only be defined over a KSDS or ESDS clusters.

1.5.11 Alternate indexes

Alternate indexes (AIXs) allows logical records of a KSDS or of an ESDS (in this context called a base cluster) to be accessed sequentially and directly by more than one key field. AIXs eliminate the need to store the same data in different sequences in multiple data sets for the purposes of various applications. Each alternate index is a KSDS cluster consisting of an index component and a data component.

Any field in the base cluster record can be used as an alternate key. It may also overlap the primary key (in a KSDS), or with any other alternate key. The same base cluster may have several AIXs varying the alternate key. The alternate key must follow all the key requirements as described in “Logical record” on page 5, with the exception of uniqueness. There may be more than one primary key value per the same alternate key value. As an example the primary key is an employee number and the alternate key is the department name, obviously the same department name may have several employee numbers.

The records in the data component contain the alternate key value and all the primary keys corresponding to the alternate key value (pointers to data in the base cluster). The primary keys in the logical record are in ascending sequence within an alternate index value. If you have a large number of primary keys per alternate key, consider defining the AIX as spanned and compressed.

The AMS program allows you to define, and then to create AIXs when the BLDINDEX command is specified. An AIX is defined only after its associated base cluster has been defined, and it can be built only after its base has been loaded with at least one record.

The BLDINDEX command causes a sequential scan of the specified base cluster, during which alternate key values and primary keys (for a KSDS) or record RBAs (for an ESDS) are extracted and put together to form alternate index records. These records are sorted by ascending alternate keys. The alternate index records are then constructed and written.

Alternate index paths

Before accessing a KSDS or ESDS through an alternate index, a path must be defined. A path is the mean by which a base cluster is accessed through its alternate indices. A path is defined and named using the AMS DEFINE PATH command. At least one path must be defined for each of the alternate indices through which the base cluster is to be accessed. The path name refers to the base cluster and alternate index pair. When a program opens a path for processing, both the base cluster and the alternate index are opened. A base cluster plus all its AIXs is called a sphere.

1.5.12 Splits

CI and CA splits occur as a result of data record insertions (or increasing the length of an already existing record) in KSDS and VRRDS organizations. If a record is to be inserted (in key sequence) and there is not enough free space in the CI, the CI is split. Approximately half of the records in the CI are transferred to a free CI provided in the CA, and the record to be inserted is placed in the original CI.

If there are no free CIs in the CA and a record is to be inserted, a CA split occurs. Half of the CIs are sent to the first available CA at end of the data component. This movement creates free CIs in the original CA, then the record to be inserted causes a CI split.

You should keep in mind that splits are not bad. Splits are how VSAM deals with a lack of space, and this generates free space that will help prevent additional splits.

As a result of the split, the physical sequence of records and CIs is no longer the same as the logical sequence. A new index entry is inserted in the sequence set for the new CI, and the existing index entry is updated.

Splits (mainly CA splits) consume resources when they occur. However, the fact that the data is spread in the 3390/3380 volume is no longer a performance issue. Refer to 2.5, “VSAM rule-of-thumb mode” on page 48.

The number of CI and CA splits is maintained in the catalog.

One example about splits is a real life situation where a customer had a KSDS loaded with just one record with key 0. First, a batch job added a few records with a very high numeric key. After the job randomly started adding lots of records with keys less than the ones inputted by batch. Thousands of splits took place affecting performance. The available solution was to run the job after the online and the number of splits went back to normal.

1.5.13 VSAM buffering

Buffering is one of the key aspects as the I/O performance is concerned. A VSAM buffer is a virtual storage area where the CI is transferred during an I/O operation. In VSAM there are two types of buffers: buffers for data CIs and buffers for index CIs. A buffer pool is a set of buffers with the same size. A resource pool is a buffer pool with some control blocks describing the pool and the clusters with CIs in the resource pool. Refer to “Buffering options” on page 63, for details on VSAM buffering.

1.6 VSAM data set organizations

VSAM arranges records by index key, relative record number, or relative byte address. It is used for direct or sequential processing of fixed-length and variable length records on DASD. VSAM data is cataloged for easy retrieval. VSAM supports five data set organizations:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Relative record data set (RRDS), of which there are two types:
  - Fixed-length relative record data set (RRDS)
  - Variable-length relative record data set (VRRDS)
- Linear data set (LDS)
- HFS files

1.6.1 Key-sequenced data set

In a KSDS organization, records are initially loaded in the data component in ascending collating sequence by key. The key contains a unique value that determines that record's collating position in the data set.

In VSAM KSDS, if a search argument of a logical record is its key field, then:

- In all logical records, the key length must be the same. The length of the key must be from one byte to 255 bytes.
- In all logical records the key offset (in relation to the logical record beginning) must be the same.
- The value of the key field cannot be duplicated among the logical records, it must be unique.
- After it is specified, the value of the key cannot be altered, however the entire record can be deleted.

- In a spanned record, this field must be located totally in the first control interval.

Logical records in a KSDS organization can be fixed or variable length records.

When a new record is added to the data component, it is inserted logically in its collating sequence by key. Refer to Figure 1-5 for the structure of a KSDS.

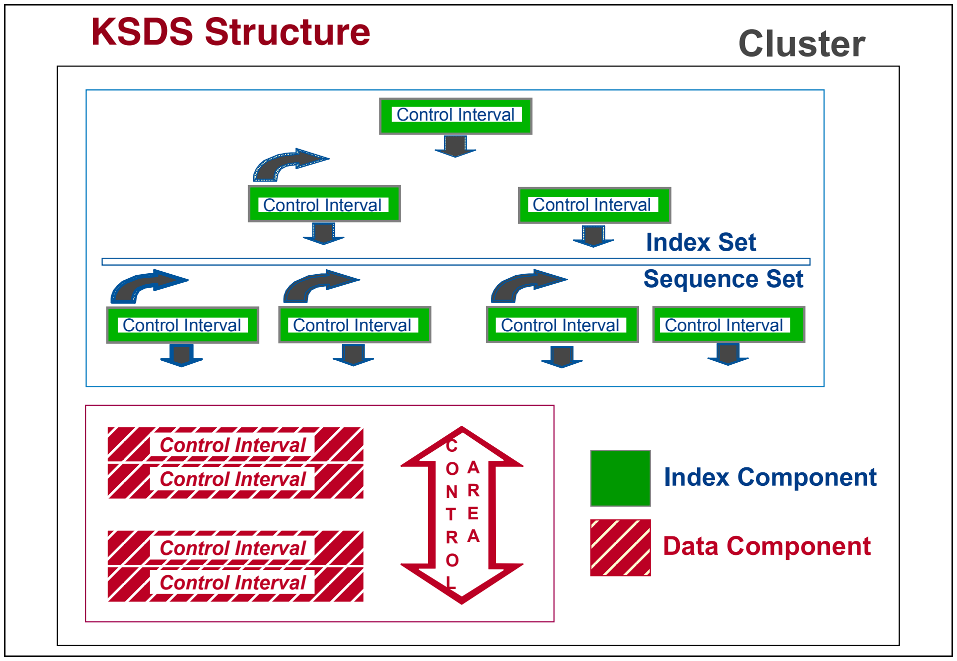


Figure 1-5 Components of a KSDS structure

For VSAM organizations where the key field in a logical record is a search argument (as with the KSDS organization), a “balanced tree” index based in these key values needs to be created. Refer to the Figure 4-1 on page 205. The index has logical records (called index records) which allows a fast random search of a data logical record with a matching key value. In the index component. There is one logical record (with the highest key) for each CI of data, and a CI for each CA of data.

KSDS types of access

There are three methods by which to access a KSDS. These are sequential, direct, and skip-sequential.

- Sequential access is used to load a KSDS, and to retrieve, update, add and delete records in an existing data set.

VSAM uses the index component to access data records in ascending or descending sequence by key. When retrieving records, you do not need to specify key values because VSAM automatically obtains the next logical record in key sequence. The sequence set (a low portion of the index) is used to find the next logical CI, through horizontal pointers.

Sequential access allows you to avoid searching the index more than once. Sequential is faster than direct for accessing multiple data records in ascending key order. Problem is that many times the application logic does not need to process all of them.

- Direct access is used to retrieve, update, add and delete records in an existing data set.

  The application program needs to supply a key value for each record to be processed. You can supply the full key or a generic key. The generic key is the high order portion of a full key. For example, you might want to retrieve all records whose keys begin with XY (where XY is the generic key), regardless of the full key value.

  VSAM searches the index from the highest level index set CI to the sequence set for a record to be accessed. Vertical pointers in the sequence set CI are used to access the data CA containing the record. The number of index CIs accessed is numerically identical to the number of layers.

  Direct access saves you a lot of I/O overhead by not retrieving the entire data set sequentially to process a small percentage of the total number of records.

- Skip-sequential access is used to retrieve, update, add and delete records in an existing data set.

  VSAM retrieves selected records, but in ascending sequence of key values. Skip sequential access avoids:

  - Retrieving the entire data set sequentially in order to process a relatively small percentage of the total number of records in key collating sequence
  - Retrieving the desired records directly, which causes the index to be searched from top to bottom level for each record

For each request the sequence set is used to find the next logical CI and to check if it contains the requested record. If the first skip-sequential search is the first access after opening the data set, a direct search is initiated by VSAM to find the first record. From then on the index sequence set level is used to find the subsequent records. If other operations were performed before (for example, read sequential), either the last position of that operation is used as a starting point to search the sequence set records, or a re-positioning is necessary. If spanned records are used for KSDS, the primary key must be within the first control interval.

You specify the KSDS organization using the IDCAMS DEFINE command with the INDEXED parameter.

1.6.2 Entry sequenced data set (ESDS)

An ESDS is comparable to a sequential non-VSAM data set in the sense that records are sequenced by the order of their entry in the data set, rather than by key field in the logical record, which could be fixed or variable length records.

All new records are placed at the end of the data set. There is no split concept. Existing records can never be deleted. As far as VSAM is concerned, the record is not deleted. It is the responsibility of the application program to identify that record as invalid. If the application wants to delete a record, it must flag that record as inactive.

Records can be updated, but without length change. To change the length of a record, you must either store it at the end of the data set as a new record, or override an existing record of the same length that you have flagged as inactive.

ESDS types of access

A record can be accessed sequentially or directly by its RBA. The specific RBA is converted to CCHHR terms, in order to prepare the locate record CCW for the I/O operation:

- Sequential access

  VSAM automatically retrieves records in stored sequence. Sequential access can be started from the beginning or somewhere in the middle of a data set. If processing is to begin in the middle of a data set, RBA positioning is necessary before sequential access can be performed.

- Direct (or random) access

  When a record is loaded or added, VSAM notes its RBA. To retrieve records directly, you must supply the RBA for the record as a search argument. Although an ESDS does not contain an index component, you can build an alternate index to keep track of these RBAs. Refer to “Alternate indexes” on page 14.

Skip sequential access is not allowed for an ESDS. Refer to 1.6.1, “Key-sequenced data set” on page 16, for more information on skip sequential access.

Figure 1-6 shows the format of an ESDS.

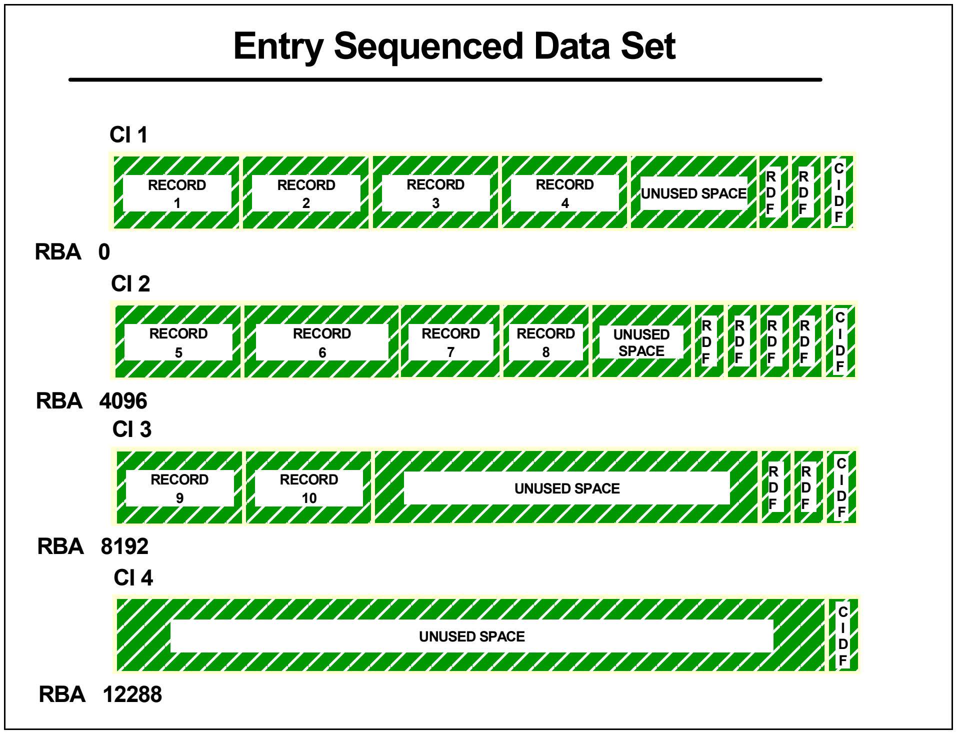


Figure 1-6 Entry sequenced data set (ESDS)

Empty spaces in the CI are referred to as unused space because they can never be used. This is a result of CI internal fragmentation.

You specify ESDS organization using the IDCAMS DEFINE command and specifying the NONINDEXED parameter.

AIX can also be applied to ESDS organization. Instead of radomly accessing an ESDS by RBA, you can ask for an AIX, where a key field in the logical record is used as a random search argument. In this case, the AIX (remember, it is a KSDS cluster) data component automatically relates the key with the RBA in the base cluster. The BLDINDEX command causes a sequential scan of the specified base cluster, during which key values and record RBAs are extracted and put together to form alternate index records. Before accessing an ESDS through an alternate index, a path must be defined.

An example of ESDS with an AIX could be a file used as a time stamp log, describing all the bank transactions in a checking account application.

1.6.3 Relative record data set

An relative record data set consists of a number of pre-formatted fixed-length slots. Each slot has a unique relative record number (RRN), and the slots are sequenced by ascending relative record number. These slots are grouped in CIs.

Each fixed length logical record occupies a slot, and is stored and retrieved by the relative record number of that slot. The position of a logical record is fixed and its relative record number (RRN) cannot change. The logical records are grouped in CIs.

Because the slot can either contain data or be empty, a data record can be inserted or deleted without affecting the position of other data records in the RRDS organization. The RDF shows whether the slot is occupied or empty. Free space is not provided because the entire data set is divided into fixed-length slots. There may be empty spaces in the CI, referred to as unused space, because they can never be used. This is a result of CI internal fragmentation.

Figure 1-7 shows the format of an RRDS.

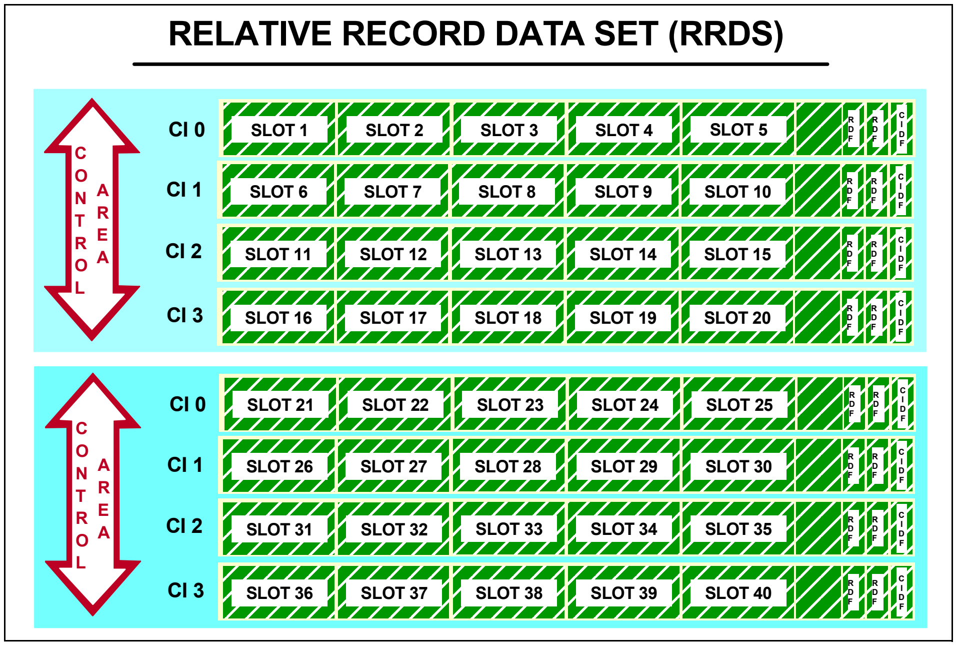


Figure 1-7 General format of a relative record data set

Typical RRDS processing

The application program inputs the RRN of the target record and VSAM is able to find its location (using CCHHR format) quickly using a formula that takes into consideration the geometry of the DASD device. The relative record number is always used as a search argument.

An RRDS can be processed sequentially, directly or skip-sequentially.

- RRDS sequential access is treated the same way as ESDS sequential access. However, empty slots are automatically skipped by VSAM.
- An RRDS can be processed directly by supplying the relative record number as a key. VSAM calculates the RBA and accesses the appropriate record or slot through CCHHR. RRDS direct address processing by supplying the RBA is not supported.
- Skip-sequential access is treated like an RRDS direct processing request, but the position is maintained. After that, records must are processed in ascending relative record number sequence.

You specify the RRDS organization using the IDCAMS DEFINE command with the NUMBERED option.

1.6.4 Variable relative record data set

A VRRDS is a fixed length RRDS, except that it contains variable-length records. Each record has a unique relative record number, and is placed in ascending relative record number order. Each record is stored and retrieved using its relative record number. VRRDS has no slots.

The relative record number of a record cannot change. When that record is erased, the relative record number can be reused for a new record.

Because the logical records have a variable size, it is not possible to use a formula to derive its RBA from its RRN. In a sense a VRRDS organization is a KSDS processed by RRN instead of a key. There is an index and data component forming the cluster. The search argument is the RRN pointing to the RBA of the corresponding logical record in the data component.

You can specify free space for inserting records and increasing the length of a record. The split concept is implemented.

You specify the VRRDS organization with the IDCAMS DEFINE command with the NUMBERED option and variable length record.

1.6.5 Linear data set (LDS)

A linear data set (LDS) contains data that can be accessed as byte-addressable strings in virtual storage. It is a VSAM data set with a control interval size multiple of 4096 bytes. An LDS has no imbedded control information in its CI, that is, no RDFs and CIDFs. All LDS bytes are data bytes. Logical records must be blocked and deblocked by the application program. Logical records are transparent from VSAM's point of view.

In a sense, a LDS is a non-VSAM file with some of the VSAM facilities, such as the use of IDCAMS and VSAM specific information in the catalog. The most common LDS exploiter is DB2®.

Like the ESDS and RRDS, an LDS contains a data component only.

Figure 1-8 shows the format of an LDS.

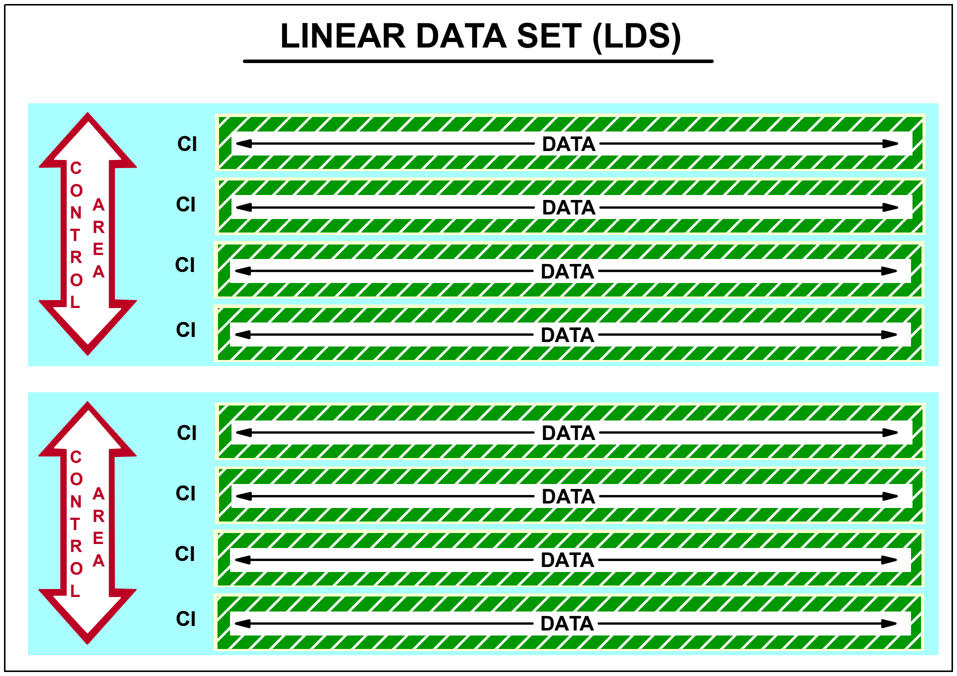


Figure 1-8 Linear data set (LDS)

You specify the LDS organization with the IDCAMS DEFINE command specifying the LINEAR parameter.

Data-in-Virtual

DIV is an optional and unique buffering technique used for LDS data sets. Application programs can use DIV to map an LDS data set or a portion of a data set into an address space, a data space, or a hiperspace. An LDS cluster is sometimes referred to as a DIV object.

Data is read into central storage through the paging algorithms only when that data block is actually referenced. During RSM page steal processing, only changed pages are written to auxiliary storage. Unchanged pages are discarded since they can be retrieved again from the permanent data set.

DIV is designed to improve the performance of applications that process large files non-sequentially in an unpredictable pattern. It reduces the number of I/O operations that are traditionally associated with data retrieval. Likely candidates are large arrays or table files.

Mapping a linear data set

To establish a map from a linear data set to a window (a program provided area in multiples of 4K on a 4K boundary), the program issues:

- *DIV IDENTIFY* to introduce (allocate) a linear data set to DIV services.
- *DIV ACCESS* to cause a VSAM open for the data set and indicate access mode (read or update).
- *DIV MAP* to enable the viewing of the data object by establishing an association between a program provided area and the data object. The area may be in an address space, data space, or hiperspace.

No actual I/O is done until the program references the data in the window. The reference results in a page fault which causes DIV services to read the data from the linear data set into the window.

- *DIV SAVE* can be used to write out changes to the data object.
- *DIV RESET* can be used to discard changes made in the window since the last SAVE operation.

1.7 Comparing VSAM data set organizations

Table 1-2 provides a summary of the characteristics of VSAM data set types described in this chapter.

Table 1-2 Comparison of ESDS, KSDS, RRDS, VRRDS, and linear data sets

ESDS	KSDS	Fixed-length RRDS	Variable-length RRDS	Linear data sets
Records are in the same order as they are entered	Records are in collating sequence by key field after load	Records are in relative record number order	Records are in relative record number order	No processing at record level
Records can be fixed or variable length	Records can be fixed or variable length	Records have fixed length	Records have variable length	No processing at record level
Direct access by RBA	Direct access by key or by RBA	Direct access by relative record number	Direct access by relative record number	Access with Data-In-Virtual (DIV) optionally
Consist of data component only	Consist of data and index components	Consist of data component only	Consist of data and index components	Consist of data component only
Alternate index allowed	Alternate indexes allowed	No alternate index allowed	No alternate index allowed	No alternate index allowed
A record's RBA cannot change	A record's RBA can change	A record's relative record number cannot change	A record's relative record number cannot change	No processing at record level
Space at the end of the data set is used for adding records	Free space is used for inserting and lengthening records	Empty slots is in the data set are used for adding records	Free space is used for inserting and lengthening records	No processing at record level
A record cannot be deleted, but you can reuse its space for a record of the same length	Space given up by a deleted or shortened record becomes free space	A slot given up by a deleted record can be reused	Space given up by a deleted or shortened record becomes free space	No processing at record level
Spanned records allowed	Spanned records allowed	No spanned records	No spanned records	No spanned records
Extended format allowed	Extended format or compression allowed	Extended format allowed	Extended format allowed	Extended format allowed

1.8 Choosing a VSAM data set type

During the application development process, the application programmer needs to make decisions about the data model. Among these decisions, we have data organization, function, type of access, performance, and recovery tools you need. VSAM has several organizations and different ways for accessing your data. How to choose the one for your application is discussed here.

Before you select a particular VSAM organization, you need answer the following questions:

- What is the main purpose of your data set? Does it look more like a log, a database, or an inventory?
- Do you need to access data by a key field in sequential or direct mode?
- Do you need to access the records in sequence, skip sequential, randomly, or all of them?
- Are all the logical records the same length?
- Does the logical record length change?
- Do you need to have insertions and deletions?
- Is the data set going to be extended?
- How often do you need to delete records?
- Do you use spanned records?
- Do you want to keep the data in order by the contents of the key field?
- Do you want to access the data by an alternate index?
- Do you want to exploit DIV?
- Do you want to use data compression?
- Do you need the utility functions provided by IDCAMS?

When you have answered these questions, you can use them to choose the best organization for your data set. Remember that VSAM data sets cannot be processed using non-VSAM applications. Similarly, non-VSAM data sets cannot be processed using the VSAM access method.

Following are our recommendations for choosing a data set organization.

- Use QSAM or BSAM if:
  - You use no direct processing.
  - There are no insertions or deletions, and no change in the logical record length.
  - Record additions are only at the end of the data set.

  - You are not concerned about IDCAMS functions.
  - You want to have data compression.
  - Performance and easy recovery are main issues.
- Use KSDS if:
  - The data access is sequential, skip sequential, or direct access by a key field.
  - You would prefer easy programming for direct data processing.
  - There will be many record insertions, deletions, and logical record length variations.
  - You may optionally access records by an alternate index.
  - Complex recovery (due to index and data components) is not a problem.
  - You want to use data compression.
- Use VRRDS if:
  - You have the same requirements as for a KSDS, but will use record number instead of a key field as argument.
- Use RRDS if:
  - The record processing is sequential, skip sequential, or direct processing.
  - Easy programming for direct processing is not a requirement.
  - The argument for accessing data in direct mode is a relative record number, not the contents of a data field (key). RRDS is suitable for the type of logical records identified by a continuous and dense pattern of numbers (such as 1,2,3,4...).
  - All records are fixed length.
  - There are a small number of record insertions and deletions, and all the space for insertions must be pre-allocated in advance.
  - Performance is an issue. RRDS performance is better than KSDS, but worse than QSAM or BSAM.
- Use ESDS if:
  - You are adding logical records only at the end of the data set and reading them sequentially (in the application control).
  - The logical record is variable length
  - You seldom need direct record processing by key (using AIX).
  - You are using a batch processing application.

- Use LDS if:
  - You want to exploit DIV.
  - Your application manages logical records.
  - Performance is an issue.

Many times, you will be using a specific VSAM organization depending on the software product you are running — for example, DB2 uses LDS data sets. In this case you do not have the option to chose the VSAM organization.

1.9 Extended format data set

Extended format is a technique that affects the way count key data (CKD) is stored in a 3390/3380 logical track. Extended format was first introduced to implement data striping. It increases the performance and the reliability of an I/O operation.

It is recommended that you convert your data sets to extended format for better performance, additional function and improved reliability A good time to convert to extended format is when you reorganize your VSAM data set.

All VSAM data set organizations can be defined in extended format, this includes KSDS, ESDS, RRDS, VRRDS, and LDS. The benefits available to extended format data sets include”

- Data striping
- Data compression
- VSAM extended addressability
- Partial space release
- System-managed buffering

When you create a VSAM component with the extended format option, there are two major differences from those not in extended format.

- If a data set is allocated as an extended format data set, 32 bytes are added to each physical block. This physical block suffix may increase the amount of space actually needed for the data set, depending on the CI size. For example, in a 3390 device with a CI size of 18432 bytes implies in additional space of 12.5%. The 32-bytes suffix contains:
  - A relative record number
  - A 3-byte field to detect controller invalid padding, thus improving the availability of the I/O operation
- The length of the data portion of the physical block is 4KB, a sort of fixed block architecture.

The block format and the suffix are transparent to the application, that is, the application does not require internal or external modifications to create and use the extended format data set.

Extended format also improves VSAM function. The VSAM I/O driver media manager only writes channel programs to exploit extended format. All the new VSAM functions are supported by media manager, therefore, EF is a prerequisite for them.

Extended format data sets must be system-managed. A system-managed data set must have an associated storage class and reside in a system-managed volume. Extended format data sets are described in the catalog as striped data sets with a stripe count of one. When a data set is allocated as an extended format data set, the data and index are extended format. Any alternate indexes related to an extended format cluster are also extended format.

When in extended format, there is no support for the key-range VSAM option, MVS checkpoint restart, or hiperbatch because these options are not supported by media manager. Key-range is not recommended in an SMS environment with the new RAID controllers.

Certain types of key-sequenced data set types cannot be allocated as extended format, including:

- Catalogs
- System data sets (SMS is not active at early IPL stages)
- Temporary data sets

If a data set is allocated as an extended format data set, 32 bytes are added to each physical block. This physical block suffix may increase the amount of space actually needed for the data set, depending on the CI size. For example, in a 3390 device with a CI size of 18432 bytes implies in additional space of 12.5 percent.

To convert a non-extended format data set to extended format, or to allocate an extended format data set, you need to create an SMS data class (DC) with the `DATASETNAMETYPE` field equal to `EXT` and assign the data sets to that data class. Refer to Figure 4-8 on page 229.

1.10 Extended addressability

With extended addressability the 4 GB VSAM architectural limit for data set size imposed by using the 4-byte field for the relative byte address (RBA) was eliminated.

It is important to state up-front that extended addressability and extended format are not the same concept. Extended format is a way of storing data in a 3390/3380 logical volume. Extended addressability provides the ability to have larger VSAM data sets. However, extended format is a prerequisite for extended addressability.

Using extended addressability, the size limit for a VSAM data set is determined by either:

- CI size multiplied by 4 GB
- The volume size multiplied by 59

A 4 K CI size yields a maximum data set size of 16 TB, while a 32 KB CI size yields a maximum data set size of 128 TB. A 4K CI size is preferred by many applications for performance reasons. No increase in processing time is expected for extended format data sets that grow beyond 4 GB.

To use extended addressability, the data set must be:

- SMS-managed
- Defined as extended format

Refer to "Extended addressability (EA)" on page 228 to get more information on the subject.

1.11 Data striping

Usually, in a multi-extent, multi-volume VSAM data set processed in *sequential* access, processing does not allow for any type of parallelism for I/O operations among the volumes. This means that when an I/O operation is executed for an extent in a volume, no other I/O activity from the same task or same data set is scheduled to the other volumes. In a situation where I/O is the major bottleneck, and there are available resources in the channel subsystem and controllers, it is a waste of these resources.

Data striping addresses this sequential access performance problem by adding two modifications to the traditional data organization:

- The records are not placed in *key ranges* along the volumes; instead they are organized in stripes.
- Parallel I/O operations are scheduled to sequential stripes (CIs) in different volumes.

By striping CIs, VSAM can spread simultaneous I/Os across multiple devices. This format allows a single application request for records in multiple tracks and CIs to be satisfied by concurrent I/O requests to multiple volumes.

The result is improved performance by achieving data transfer into the application at a rate greater than any single I/O path.

1.12 Processing a VSAM cluster

Before we cover how to process a VSAM cluster it is recommended that you have an idea about the source of VSAM processing options. Refer to “Major sources of VSAM processing options” on page 233.

To process a VSAM cluster the following actions must be executed:

- Allocate the data set to establish the logical link between a program and a data
- Open the data set, identifying it with a DDNAME.
- Accessing data through GETs and PUTs using an access method.
- Close the data set.
- Unallocate the data set.

Following is a brief discussion of the VSAM processing actions.

1.12.1 Allocating a VSAM cluster

First, we want to differentiate between data set allocation and data set creation. The expression allocate a data set does not imply creating a data set, one can allocate an already existing data set. Allocate a data set to a task, means to create a data set (if it does not exist) and establish a connection between a program running under such task with the data set. However, after allocation the data set must first be opened in order to be accessed. This connection is represented by an entry in the task’s TIOT (a control block) linking the DDNAME with the UCB representing the device containing the data set.

There are several ways to allocate a data set, through a DD card, through TSO ALLOC command, IDCAMS or by issuing the DYNALLOC macro within a program.

The functions used at allocation time depend if the data set already exists or not:

- If it exists, its device is located through a catalog search, a serialization ENQ is issued, and a VTOC search is executed.
- If it does not exist, a device needs to be selected by SMS and SRM. The data set is created, an entry made in the VTOC, and catalogued (all VSAM components must be catalogued).

Defining VSAM Clusters

To define a cluster is an VSAM expression and it means to create and catalog the cluster, it does not imply allocating the data set. You can define a VSAM data set using any of the following methods:

- **IDCAMS DEFINE or ALLOCATE** commands. Here, clusters are defined but not allocated to a task
- **ALLOCATE TSO command** allocates (defining is included) a cluster VSAM from a TSO terminal. The TSO commands are described in **MHL** *OS/390 TSO/E Command Reference,* SC28-1969.
- **JCL DD statements** define and allocate a VSAM cluster. For information on using JCL, see *z/OS MVS JCL Reference,* GC28-1757, and *z/OS MVS JCL User's Guide,* GC28-1758.
- **Dynamic allocation** allocates and defines a VSAM cluster. The DYNALLOC macro is described in *z/OS MVS Authorized Assembler Services Guide,* GC28-1763.

Using IDCAMS

When using IDCAMS to define a VSAM data set, you specify:

- Component name or cluster name.
- Data set type. The default is INDEXED (KSDS).
- Space allocation, both primary and secondary allocations, and the volumes on which the cluster’s components are to have space.
- Data set attributes, such as record size and CI size. For a KSDS, you specify key information and free space.
- Catalog information.

Figure 1-9 shows the syntax of the DEFINE command and required parameters.

```
DEFINE CLUSTER -
      (NAME(entryname))-
      CYLINDERS(primary secondary)|
      KILOBYTES(primary secondary)|
      MEGABYTES(primary secondary)|
      RECORDS (primary secondary)|
      TRACKS(primary secondary) -
      VOLUMES(volser[volser...]) -
  DATA (parameters) -
  INDEX (parameters) -
  CATALOG(subparameters)
```

Figure 1-9 DEFINE command required parameters

1.12.2 Accessing VSAM cluster

VSAM clusters can be accessed by different types of programs, such as batch program, CICS® application programs, IDCAMS, and DITTO. TSO REXX and CLISTs cannot be used to access VSAM entities.

Batch and CICS application programs

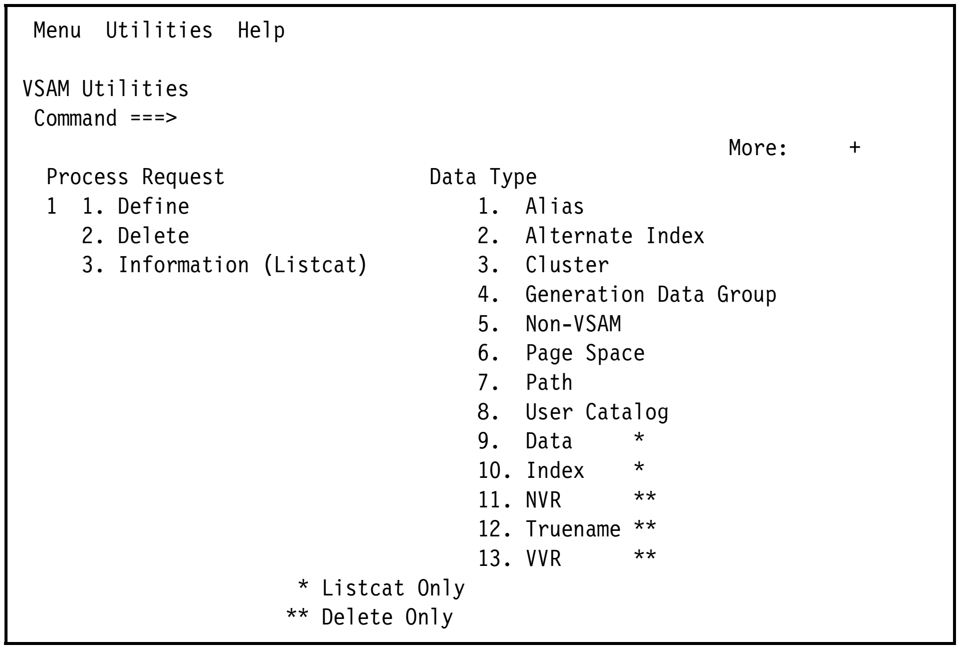
They can be written using languages that support VSAM, such as: COBOL, PL/I, JAVA, and Assembler. To obtain VSAM services, these application programs use VSAM macros, mainly the GET/PUT. For details, refer to *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913.

IDCAMS

IDCAMS is a program included in DFSMSdfp used to establish and maintain catalogs and VSAM clusters.

IDCAMS can be invoked:

- As a job or job step by specifying PGM=IDCAMS on the EXEC card.
- From a TSO terminal executing commands. There is also under ISPF an user friendly option 3.2.V (VSAM utilities), that produces a panel where certain functions may be asked. Refer to Figure 1-9.
- From an application program.



```
 Menu  Utilities  Help

VSAM Utilities
 Command ===>
                                                           More:     +
  Process Request                Data Type
  1  1. Define                       1.  Alias
     2. Delete                       2.  Alternate Index
     3. Information (Listcat)        3.  Cluster
                                     4.  Generation Data Group
                                     5.  Non-VSAM
                                     6.  Page Space
                                     7.  Path
                                     8.  User Catalog
                                     9.  Data     *
                                     10. Index    *
                                     11. NVR      **
                                     12. Truename **
                                     13. VVR      **
                        * Listcat Only
                       ** Delete Only
```

Figure 1-10 ISPF under TSO

You can also call the IDCAMS program from within another program and pass the Access Method Services command and parameters to the IDCAMS program.

There are two types of Access Method Services commands: functional commands and modal commands.

- Functional commands are used to request the actual work, for example, defining a data set or listing a catalog, see Figure 1-11.
- Modal commands allow the conditional execution of functional commands. TSO users can use functional commands only. See *DFSMS Access Method Services for Catalogs*, SC26-7394 for more information.

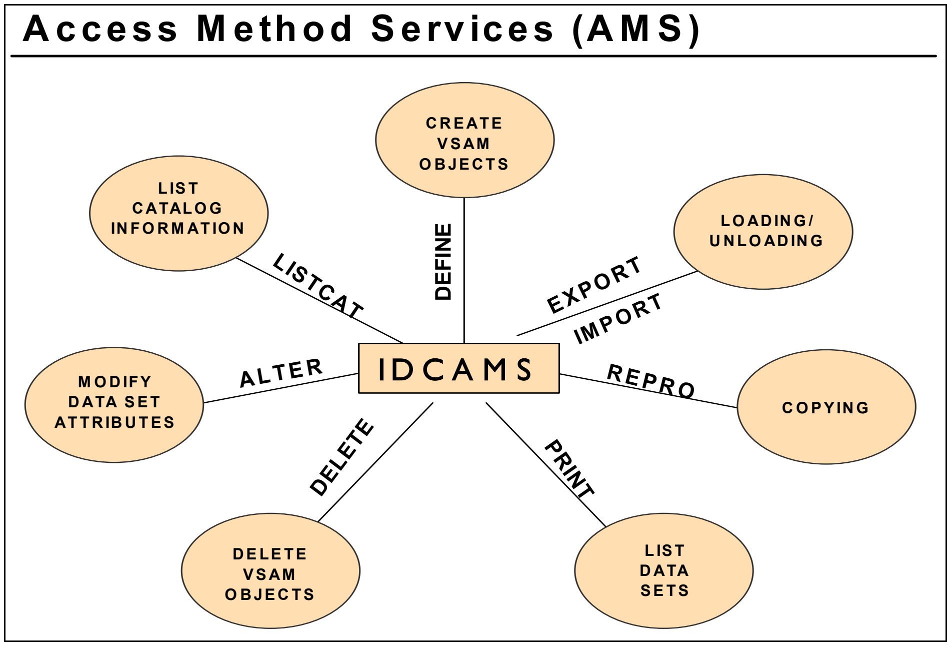


Figure 1-11 AMS commands

DITTO/ESA

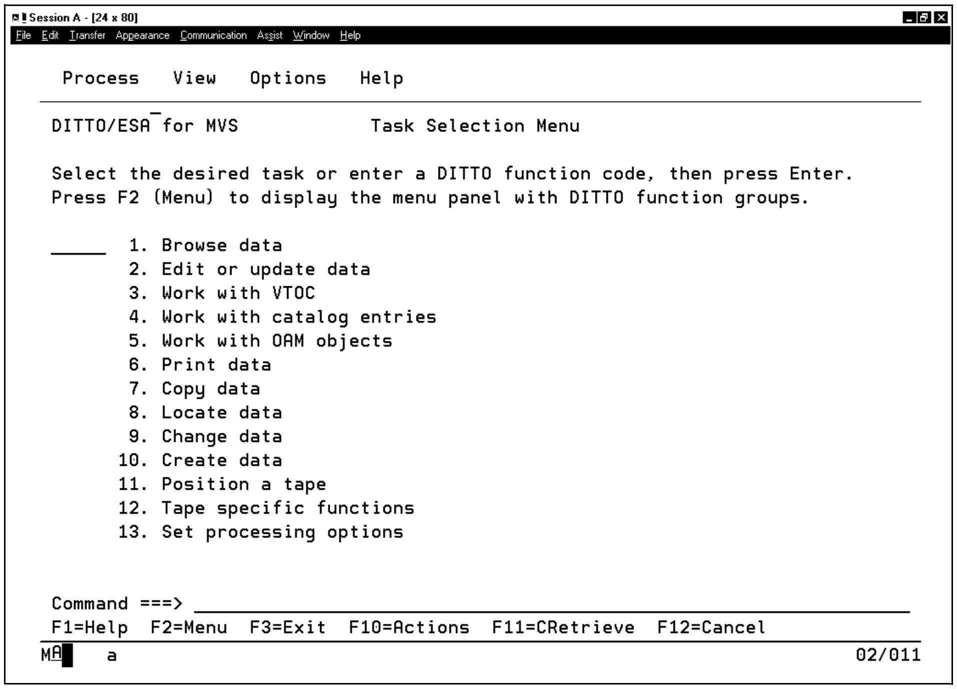
DITTO is a very powerful utility product that you can use to browse, edit, and delete VSAM records.

You can start DITTO in full-screen mode from a TSO terminal. Check with your system programmer how to invoke DITTO as start procedures may vary with the installation.

In full-screen mode, you can use menus, online help, and interactive browse and update functions. You will probably find full-screen mode the most convenient way to run DITTO, especially if you are a new DITTO user.

You can also run DITTO in line, command, or batch modes. Refer to *DITTO/ESA V1R3 User's Guide,* SH19-8221, for more information on using DITTO.

Figure 1-12 shows the DITTO Task Selection Menu. Choose option 1 and then option 3 to browse a VSAM data set.



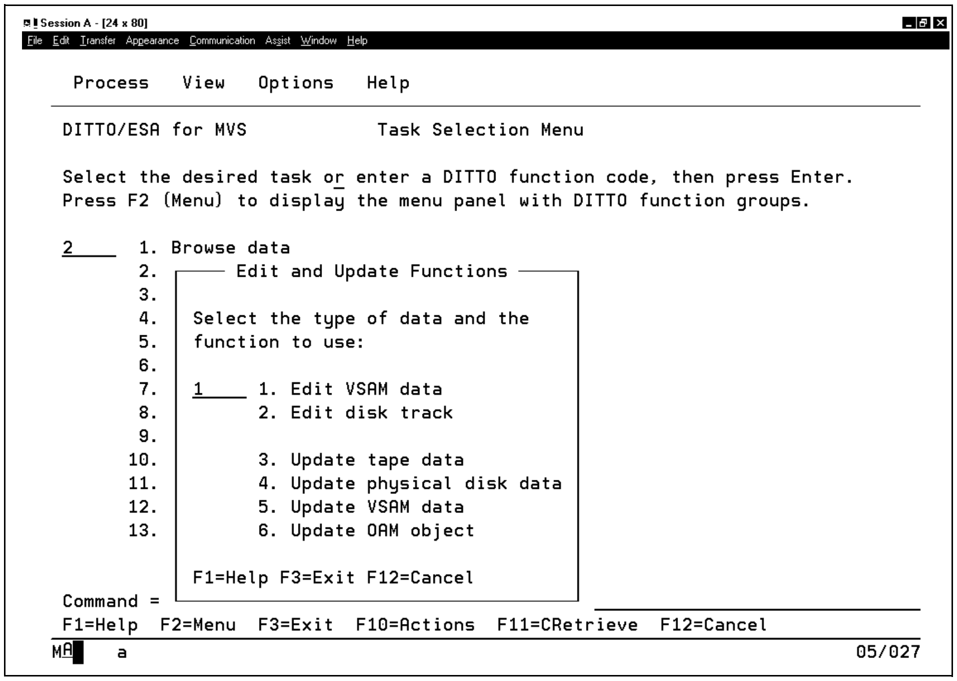
Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

```
  Process   View   Options   Help
 ------------------------------------------------------------------------------
 DITTO/ESA for MVS            Task Selection Menu

 Select the desired task or enter a DITTO function code, then press Enter.
 Press F2 (Menu) to display the menu panel with DITTO function groups.

 _____  1. Browse data
        2. Edit or update data
        3. Work with VTOC
        4. Work with catalog entries
        5. Work with OAM objects
        6. Print data
        7. Copy data
        8. Locate data
        9. Change data
       10. Create data
       11. Position a tape
       12. Tape specific functions
       13. Set processing options


 Command ===> _________________________________________________________________
 F1=Help  F2=Menu  F3=Exit  F10=Actions  F11=CRetrieve  F12=Cancel
MA    a                                                                 02/011
```

Figure 1-12   DITTO selection panel

Choose option 2 on the Task Selection Menu and then option 1 to edit a VSAM data set.

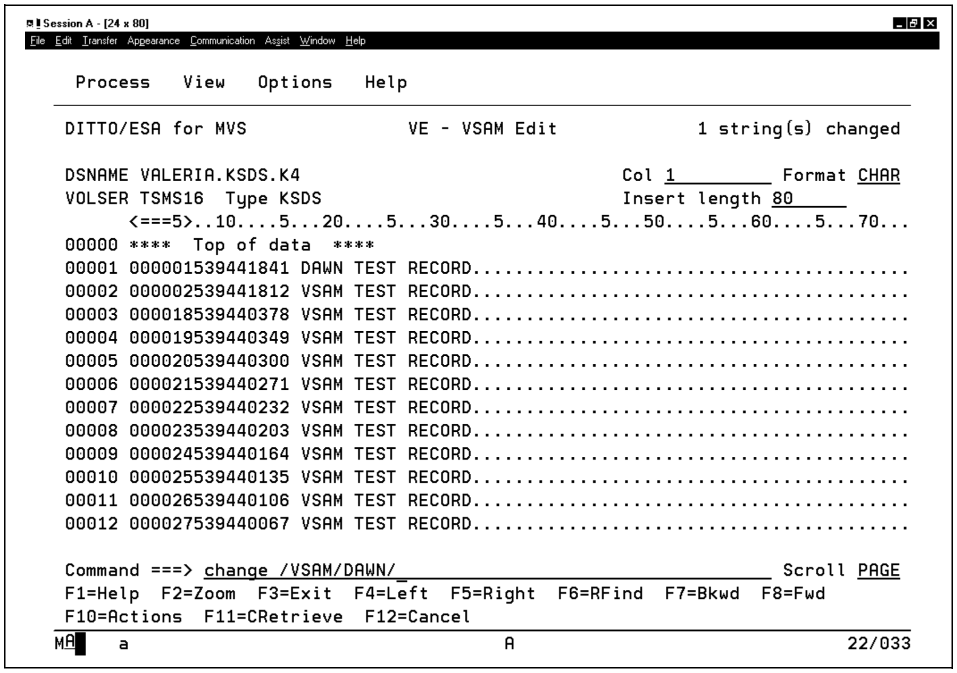
Figure 1-13 shows the DITTO edit menu.


Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

```
  Process   View   Options   Help
 ------------------------------------------------------------------------------
 DITTO/ESA for MVS            Task Selection Menu

 Select the desired task or enter a DITTO function code, then press Enter.
 Press F2 (Menu) to display the menu panel with DITTO function groups.

 2____   1. Browse data
         2. ┌──── Edit and Update Functions ────┐
         3. │                                   │
         4. │ Select the type of data and the   │
         5. │ function to use:                  │
         6. │                                   │
         7. │ 1____  1. Edit VSAM data          │
         8. │        2. Edit disk track         │
         9. │                                   │
        10. │        3. Update tape data        │
        11. │        4. Update physical disk data│
        12. │        5. Update VSAM data        │
        13. │        6. Update OAM object       │
            │                                   │
            │ F1=Help F3=Exit F12=Cancel        │
 Command =  └───────────────────────────────────┘ ______________________________
 F1=Help  F2=Menu  F3=Exit  F10=Actions  F11=CRetrieve  F12=Cancel
MA     a                                                                05/027
```

Figure 1-13 DITTO edit function

Figure 1-14 shows an example of editing a record using DITTO.


Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

```
   Process   View   Options   Help
 ------------------------------------------------------------------------------
  DITTO/ESA for MVS              VE - VSAM Edit           1 string(s) changed

  DSNAME VALERIA.KSDS.K4                               Col 1_________ Format CHAR
  VOLSER TSMS16  Type KSDS                             Insert length 80_____
        <===5>..10....5...20....5...30....5...40....5...50....5...60....5...70...
  00000 ****  Top of data  ****
  00001 000001539441841 DAWN TEST RECORD.......................................
  00002 000002539441812 VSAM TEST RECORD.......................................
  00003 000018539440378 VSAM TEST RECORD.......................................
  00004 000019539440349 VSAM TEST RECORD.......................................
  00005 000020539440300 VSAM TEST RECORD.......................................
  00006 000021539440271 VSAM TEST RECORD.......................................
  00007 000022539440232 VSAM TEST RECORD.......................................
  00008 000023539440203 VSAM TEST RECORD.......................................
  00009 000024539440164 VSAM TEST RECORD.......................................
  00010 000025539440135 VSAM TEST RECORD.......................................
  00011 000026539440106 VSAM TEST RECORD.......................................
  00012 000027539440067 VSAM TEST RECORD.......................................

  Command ===> change /VSAM/DAWN/_________________________________ Scroll PAGE
  F1=Help  F2=Zoom  F3=Exit  F4=Left  F5=Right  F6=RFind  F7=Bkwd  F8=Fwd
  F10=Actions  F11=CRetrieve  F12=Cancel
 ------------------------------------------------------------------------------
 MA     a                                 A                              22/033
```

Figure 1-14 VSAM edit panel

1.12.3 Unallocation

Unallocation undoes what allocation did. There are three ways to unallocate a VSAM cluster.

- Closing the data set can perform this function if FREE=CLOSE was specified for the allocation.
- Your program can call dynamic unallocation.
- During the step termination process, the initiator/terminator program automatically unallocates any remaining allocated data sets, not to be accessed in the following steps.

1.13 VSAM exploiters

The major applications that exploit VSAM functions are described in this section.

1.13.1 DB2

DB2 uses linear (LDS) VSAM data sets for its table spaces, without implementing Data-in-Virtual. All the control (including buffer pool) is done by DB2. For example, DB2 implements data striping in LDS data sets.

1.13.2 Hierarchical file system (HFS)

HFS is a UNIX® Services data organization with no determined logical records — it is a byte string — made of embedded directories and data. It can be accessed by Posix code (a UNIX standard for operating system interfaces, APIs) running under z/OS or by z/OS applications. In the second case an HFS looks like a VSAM ESDS organization and is accessed in the same way.

Accessing HFS files through VSAM

You can access an HFS file through VSAM in one of the following ways:

- JCL DD statement specifying PATH=pathname
- SVC99
- TSO ALLOCATE command

HFS files are simulated as an ESDS. However, since HFS files are not actually stored as ESDSs, VSAM cannot simulate all the characteristics of a sequential data set. As a result, there are certain macros and services which have incompatibilities or restrictions when dealing with HFS files. Refer to *DFSMS Using Data Sets*, SC26-7410 for more information.

1.13.3 zSeries File System (zFS)

zSeries® File System (zFS) is a z/OS UNIX file system that can be used in addition to the Hierarchical File System (HFS). zFS provides significant performance gains in accessing files approaching 8K in size that are frequently accessed and updated. The access performance of smaller files is equivalent to that of HFS. zFS provides reduced exposure to loss of updates by writing data blocks asynchronously and not waiting for a sync interval. zFS is a logging file system. It logs metadata updates. If a system failure occurs, zFS replays the log when it comes back up to ensure that the file system is consistent.

Application programming interfaces

zFS file systems contain files and directories that can be accessed with the z/OS hierarchical file system application programming interfaces on the z/OS operating system as follows:

- An application interface composed of C interfaces, some of which are managed within the C Run-Time Library (RTL), while others access kernel interfaces to perform authorized system functions on behalf of the unauthorized caller

- An interactive z/OS shell interface by shell users

A zFS aggregate

A zFS aggregate is a data set that contains zFS file systems. The aggregate is a VSAM Linear Data Set (VSAM LDS) and is a container that can contain one or more zFS file systems. An aggregate can only have one VSAM LDS, but it can contain an unlimited number of file systems. The name of the aggregate is the same as the VSAM LDS name. Sufficient space must be available on the volume or volumes, as multiple volumes may be specified on the DEFINE of the VSAM LDS. DFSMS decides when to allocate on these volumes during any extension of a primary allocation. VSAM LDSs greater than 4 GB may be specified by using the extended format and extended addressability capability in the data class of the data set.

1.13.4 CICS

CICS allows the exploiting of VSAM clusters by CICS transactions. CICS uses the MVS system logger data set for logging.

1.13.5 DFSMShsm

DFSMShsm™ has three control data sets, respectively migration control data set (MCDS), backup control data set (BCDS) and offline control data set (OCDS). All of them are VSAM KSDSs. In a multi-MVS image environment these data sets can be shared between distinct DFSMShsm instances. This environment is called HSMplex.

1.13.6 DFSMSrmm

DFSMSrmm™ also uses VSAM data sets for their control data sets. They are defined with SHAREOPTIONS(3 3) and they use the VSI control block to ensure that they have the most recent HURBA on each system.

1.13.7 Java Record I/O (JRIO)

JRIO is a set of APIs under OS/390 2.6 (JDK 1.1.8) level allowing the access of Java™ code to I/O record oriented data organizations. It is an additional to java.io APIs which only supports HFS type of access.

JRIO lets Java applications access traditional z/OS file systems in addition to the Hierarchical File System (HFS). JRIO makes it easier for Java applications to access records within files and to access file systems through native methods when java.io Application Programming Interfaces (APIs) do not support those file systems.

JRIO is a class library, similar to java.io. While java.io provides byte-oriented or field-oriented access to files, JRIO provides record-oriented access, which is much more natural. The major differences are:

- Read/write sequential and random access for a byte string data sets is provided by java.io.
- JRIO allows:
  - Read/Write, append, update-in-place (changing length), insert, delete
  - Different types of records format as: fixed, variable, spanned, undefined
  - Different types of access as: sequential, key direct, RBA direct, skip sequential

JRIO lets record-oriented applications (supporting multiple file systems) run using files on different file systems. It also provides a set of z/OS native code drivers to access:

- Virtual Sequential Access Method (VSAM) data sets (KSDS only)
- Non-VSAM record-oriented data sets (sequential or random access)
- The system catalog (listing the High Level Qualifiers (HLQ) from the system catalog and data sets for a given HLQ)
- Partitioned data set (PDS) directory
- HFS for sequential and random I/O access to *records*. The HFS support uses pure Java code to provide a set of concrete classes and directory classes that use the underlying java.io. JRIO also provides navigational support for HFS directories.

To run a JRIO application, Java commands implicitly set the CLASSPATH for the JRIO classes, which reside in the same subdirectory as the Java for z/OS classes.

Then, you should update your CLASSPATH to include the application classes by using the following Shell command:

```
export CLASSPATH=.:/u/joe/java/myclasses:$CLASSPATH
```

In this example, the class loader first scans the current directory for the application classes. If that fails, the class loader then scans the /u/joe/java/myclasses directory.

To run the JRIO sample programs, update CLASSPATH to include the JRIO sample classes by using the following Shell command:

```
export CLASSPATH=$CLASSPATH:$JAVA_HOME/recordio:
                    $JAVA_HOME/recsamp.jar/
```

JRIO and VSAM

JRIO provides indexed I/O access to records within a VSAM KSDS. The VSAM support uses z/OS native code to provide a set of concrete classes that implement the KeyedAccessRecordFile class. This lets you access records:

- In entry sequence order
- By primary unique key
- By alternate unique or non-unique key

In “JRIO API examples” on page 364, you’ll find a set of APIs that you may use to access KSDS VSAM data sets.

2

Performance

How can you get most out of your VSAM clusters? How can you improve data storage and retrieval? What parameters can be used when you define a VSAM data set to enhance data access?

This chapter answers these questions and describes the VSAM functions that can enhance performance. Hints and tips are provided to help you implement these VSAM functions. However the specific performance aspects of VSAM RLS and Transactional VSAM are discussed in their respective chapters.

Before we discuss VSAM performance, we first review some performance basics.

2.1 Service level agreement

To make the subject of performance more objective and related to specific business needs, the Service Level Agreement or SLA was introduced.

The SLA is a agreement between Information Systems organizations and user departments that should describe:

- Average transaction *response time* (Tr) for accessing:
  - Network
  - Input/output (I/O)
  - CPU
- The distribution of the response times, a measurement about how erratic they are
- The throughput, also called external throughput rate (ETR), measured in ended transactions per second of elapsed time (not CPU time)
- System availability, which is the percentage of time that the system is available to the end user

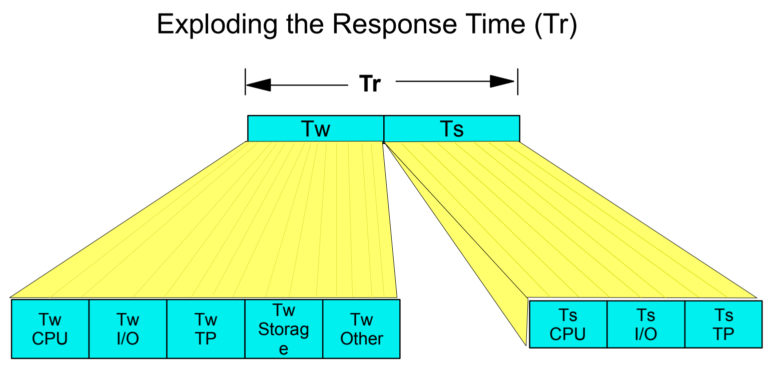
2.2 Transaction performance

A transaction is a business unit of work produced by an online or batch interaction with an end user or a department with the system. It can be a CICS, TSO, a WEB, an APPC, distributed DB2, or even a batch interaction. If your MVS is in Workload Manager (WLM) goal mode, all these transactions are monitored and accounted for by z/OS.

To characterize the performance of a transaction, we need to understand its different response time components. Figure 2-1 shows general response time components, where:

- `Tr = Ts + Tw`

  Ts is Service Time, Tw is Waiting Time (also called queue time), and Tr is Response Time



Tw (CPU):	Time with ready dispatchable units, but not enough dispatching priority.
Tw (I/O):	Time with I/Os delayed in UCB or channel subsystem.
Tw (TP):	Time in the VTAM or TCP/IP queues.
Tw (Storage):	Time suffering a page fault or being swapped-out.
Tw (Other):	Time delayed by operator, ENQ, data set recall, server AS service.
Ts (CPU):	Time with dispatchable units executing CPU.
Ts (I/O):	Time with I/Os being executed (connected or disconnected).
Ts (TP):	Time being transported in the network.

Figure 2-1   Response time components

- Exploding the above formula, we have:

  ```
  Ts = Ts(CPU) + Ts(IO) + Ts(TP)
  Tw = Tw(CPU) + Tw(IO) + Tw(TP) + Tw(Storage) + Tw(Other)
  ```

- **`ETR = Ne / T`**

  ETR is the External Throughput Rate, Ne is the number of ended transactions, T is the elapsed time.

- There is also one formula relating the Tr with ETR, derived by Little’s law:

  `ETR` `= N / (Tt + Tr)`

  In this formula, N is the average number of users sending transactions (logged on), and Tt is the average thinking time of these users. Following are some considerations regarding this formula:

  – The variables that more intensively affect the ETR are N and Tt due to their usual numeric values. Therefore, you should never accept an SLA specifying ETR, because the only variable that the IS department can directly control is Tr.

  – However, experiences show that when Tr is in a sub-second value, the value of Tt drops dramatically. This fact has to do with the human behavior in front of a terminal. If the machine responds fast (in a sub-second value), we also respond fast.

Transaction response time is the best way to characterize the performance of a transaction. Here, the target is to reduce its value and consequently to increase the ETR figures (when the value is below one second). Remember that in RMF reports the Ts(TP) and Tw(TP) are not included in the response time pictured.

2.3 Performance management

Performance management is the activity in an installation that monitors and allocates data processing resources to transactions according to a service level agreement (SLA).

There are three main ways to solve performance problems:

- **Buy:** You can simply buy more resources.
- **Tune:** Tuning your system makes more effective and efficient use of those resources.
- **Steal:** You can “steal” the resources from a less critical transaction.

2.3.1 I/O performance

As CPU speed increases, the I/O response time (I/O Tr) is the determinant factor in the average transaction response time, as shown in the formula:

I/O Tr = I/O Ts + I/O Tw

So obviously, you can get excellent response time returns by reducing I/O wait time and I/O service time.

Generally speaking, you can reduce the average I/O response time (I/O Tr) using software or hardware techniques. The software techniques aim to decrease the number of I/O operations, by implementing:

- Virtual address space (above and below the bar) buffer and data space buffers
- Hiperspace™ buffers
- Data compression
- Data striping

Examples of hardware techniques are:

- Faster channels (as FICON™, FICON Express)
- Faster device paths (adapters) to the controllers
- Larger controller cache
- More DASD subsystem concurrency, for example, parallel access volumes (PAV) in Enterprise Storage Server® (ESS).
- Faster disks (RPMs), small size, RAID-10

However, experience has shown that when you succeed fighting an I/O bottleneck, increasing the I/O rate (for example, implementing data striping), suddenly the bottleneck is moved from the I/O subsystem to the processor.

2.4 VSAM performance management

The goal of VSAM performance management is to decrease the values of I/O wait time (Tw(IO)) and I/O service time (Ts(IO)) — that is, the I/O response time (Tr(IO)), for the transactions accessing VSAM data. It can be done by avoiding I/O or doing it faster and playing with I/O priorities. These priorities must be closely linked to your business needs.

In this performance chapter, we use two approaches to help you to improve VSAM performance. First, we discuss all the VSAM external parameters whose values may affect performance. We give recommendations based on experience, on how those parameters should be set. This is known as “rule-of-thumb” (ROT) mode. At each topic end you have a set of *recommendations,* which summarize the explanations.

In the second approach, we simulate a scenario (as real as possible), where one or several VSAM data sets are causing performance problems to key transactions in a real installation. We suggest a methodology to fix the situation. Here some of the recommendations covered in the rules of thumb are presented again, now connected with an specific VSAM behavior.

Also, other factors not connected to VSAM parameters, such as I/O configuration delays, use of FICON channels, SMS storage class attributes, use of Hiperbatch™, and workload CPU bound, are introduced and discussed.

Throughout this chapter, you see a set of screens containing real RMF reports covering VSAM I/O measurements related to the theories presented. These measurements were obtained in our lab setup described in “General lab description” on page 396.

2.5 VSAM rule-of-thumb mode

Before we start, let us make a general statement about VSAM cluster parameters affecting performance and their defaults. You should note that these defaults were established a long time ago, when virtual storage was all below the 16 MB line; central storage size restricted; the DASD was slow, removable, and expensive; and channels were a scarce resource.

Therefore, the defaults are outdated in many cases. The reason IBM does not change the defaults is to maintain compatibility with your existing workload. Compatibility is an important feature protecting your investment. You do not need to throw out your code and hardware when z/OS changes a release. However, keep in mind that these VSAM defaults can be easily overwritten through JCL, the ACB macro, data class constructs, ACS routines, or IDCAMS.

2.5.1 Invalid rules-of-thumb

Another important point to mention is that you may have heard many recommendations about data set placement and the effect on I/O performance. We call these recommendations “invalid rules-of-thumb” (IROTs). Generally speaking, they are out of date due to the introduction of the RAID controllers and enhancements to channel programs, such as these:

- The physical 3390/3380 volume concept no longer exists, and we do not know where their logical tracks are located on the disks, so we should not care about seek arm movement considerations, in such devices.

  Also, in the past we had several changes in the device geometry as new products were introduced. Now the logical 3390/3380 geometry will still be valid for a long time because it is not affected by enhancements to the disk controllers. For example, the device independence recommendation (records instead of tracks for units of allocation), due to changes in future track or cylinder geometry, is not a consideration anymore.
- In the case of a cache miss, the channel always reads (or writes) in cache, asynchronously in relation to the disk, so there are no extra revolutions. caused by 3390 RPS misses.
- The existence of the Define Extent Format CCW, the Prefix CCW, and Parallel Access Volume (PAV) for the Enterprise Storage Server (ESS).

The following recommendations (IROTS), are *no longer valid*:

- Place your VTOC around the middle of the volume, or at 1/3 for the purists.
- If you need to place two active data sets in the same volume, place them close to each other, to avoid arm stealing along seeks.

- Do not allow much secondary allocation in the same volume because of the embedded long seeks in the same data set.
- Use VSAM KSDS embedded sequence set index records to minimize seeks.
- Use VSAM KSDS replication to avoid unnecessary revolutions.
- When allocating a data set on device dependent geometry, use cylinders, not tracks, for better performance.
- It is better to use a device independent geometry for allocation units (for example, records) to avoid modifications when the 3390/3380 geometry changes.
- Reorganize your KSDS data set after some CA splits in order to avoid long embedded seeks. (For information on VSAM KSDS data set reorganization refer to 4.1, “Reorganization considerations” on page 204.)
- Avoid channel utilization above 30% to inhibit RPS misses and another DASD revolution.
- Use VSAM keyrange to have control over the allocation of the key ranges of your data set.
- Avoid many active data sets in the same ESS volume.

2.6 Parameters affecting performance

The following VSAM parameters and options can affect performance.

2.6.1 Allocation units

When you define your *new* data set, either the end user or SMS administrator through defining a data class (DC) must specify the amount of space to be allocated for it. One of the allocation functions for a new data sets is the creation of the DSCB in the VTOC, is done by the DADSM routine. Refer to “Allocating a VSAM cluster” on page 31, for more information about allocation. If SMS is active, you can specify a data class and take advantage of the space allocation set by your storage administrator. If you choose to specify space explicitly, you can specify it for VSAM data sets in units of records, kilobytes, megabytes, tracks, or cylinders.

DADSM only accepts allocation requests in tracks or cylinders. The units specified (records, kilobytes, or megabytes) are converted by IDCAMS before the DADSM request.

If you specify records as the allocation unit, the number of records you declare is multiplied by the value of the keyword RECORDSIZE(AVERAGE) value, to derive the space in bytes. This keyword can also indicate the maximum value allowed for the record length. Later during the access, If the maximum record length is exceeded, VSAM rejects the new record.

When the primary amount on the first volume is used up, a secondary amount is allocated on that volume by the end-of-volume (EOV) routine. VSAM acquires space in increments of control areas (CAs). Each time a new record does not fit in the allocated space, EOV allocates more space in the secondary space amount. This can be repeated until the volume is out of space or the extent limit is reached. Depending on the type of data set allocation request, a new volume may be used.

Do not define a small amount of secondary space allocation value, especially for a KSDS or a VRRDS data set. There are a large number of I/O operations involved when the secondary allocation takes place.

We do not state any recommendations on allocation because these depend on the use, or not, of the Guaranteed Space option.

2.6.2 Guaranteed Space

The allocation function depends on whether the data set has the Guaranteed Space attribute.

The Guaranteed Space, a storage class SMS attribute lets you reserve space on specific volumes (VOLSER) for clusters that require special placement to meet performance or availability requirements. For example, IMS™ logs that are duplexed by IMS to improve availability should be on separate volumes. However, the specified volumes must be described in the storage group associated with the cluster.

Typically, you use storage class performance and availability attributes and storage group assignments to determine where to place your data set. You assign storage groups in your storage group ACS routine. SMS then selects volumes by evaluating each candidate's ability to satisfy the performance, availability, and space requirements for the data set. To place data sets on specific volumes, assign a storage class to the data set that supports Guaranteed Space and maps to the correct storage group. If the resulting storage group does not contain the specified volume, or all the volumes are not in the same storage group, the allocation will fail.

For non-guaranteed space data set allocation, when you allocate space, it is possible for the user to specify whether to use primary or secondary allocation amounts when extending to a new volume. This is done with an SMS

DATACLASS parameter for VSAM attributes, as shown in Figure 2-2, of the ISMF application.

```
 ADDITIONAL  The ADDITIONAL VOLUME AMT field shows the type of allocation
 VOLUME AMT  amount when a VSAM data set in extended format begins
allocation
 ---(15)---  on subsequent new volumes.

   Possible values:

       PRIMARY    Primary allocation amount has been requested.

       SECONDARY  Secondary allocation amount has been requested.

       ---------  If the value has not been specified.  The system will use
                  the default value of Primary.
```

Figure 2-2 Data Class panel

For a Guaranteed Space data set allocation, the following conditions must met:

- All volumes specifically specified in VOLSER belong to the same storage group.
- The storage group to which these volumes belong is in the list of storage groups selected by the ACS routines for this allocation.

We recommend that you allocate space at the data component level, because this is the component that you are able to size. VSAM allocates space as follows:

- If space is specified at the cluster or alternate index level (refer to 1.5.11, “Alternate indexes” on page 14), the amount needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data.
- If space is specified at the data component level only, the specified amount is assigned to data. The amount needed for the index is in addition to the specified amount.
- If allocation is specified at both the data and index levels, the specified data amount is assigned to data and the specified index amount is assigned to the index.
- If secondary allocation is specified at the data level, secondary allocation must be specified at the index level or the cluster level.

Recommendations

- For KSDS and VRRDS formulate space for data component only.

- Use the most you can space information from data class definition.
- Avoid Guaranteed Space.
- Avoid to use tracks as unit to get rid of the contiguous requirement. Use records, this unit is closer to the application needs.
- Do not use a very small secondary allocation figure.
- Consider using “Allocation constraint relief” on page 54.

2.6.3 Optimizing control area (CA) size

Before we discuss this topic, we want to clarify that when we say cylinder and track, we are referring to the logical 3390/3380 cylinder and track, not the cylinder and track of the disks in the RAID DASD controllers.

Generally, the primary and secondary space allocation amounts determine the CA size, as follows:

- If either the primary or secondary allocation is smaller than one cylinder, the smaller value is used as the CA size.
  - TRACKS(100,3): This results in a 3-track CA size.
  - TRACKS(3,100): This results in a 3-track CA size.
  - KILOBYTES(100,50): The system determines the control area based on 50 KB, resulting in a 1-track CA size.
  - RECORDS(2000,5): Assuming that 10 records would fit on a track, this results in a 1-track CA size.
- If both the primary and secondary allocations are equal to or larger than one cylinder, the CA size is one cylinder. An exception is for data striping, where the CA size can be 16 tracks (one more than the cylinder) — the maximum size for a CA.
  - CYLINDERS(5,10): This results in a 1-cylinder CA size.

For KSDS and VRRDS organizations the index CI size and buffer space can also affect the CA size. The previous examples assume the index CI is large enough to handle all the data CIs in the CA, and the buffer space is large enough not to affect the CI size.

A spanned record cannot be larger than a CA minus the control information size (10 bytes per CI for fixed logical records length). Therefore, do not specify large spanned records and a small primary or secondary allocation which results in a CA not large enough to contain the largest spanned record.

CA size has significant performance implications. One-cylinder CAs have the following advantages:

- There is a smaller probability of CA splits.
- The index is more consolidated. One index CI addresses all the CIs in a CA. If the CA is large, fewer index records and index levels are required. For sequential access, a large CA decreases the number of reads of index records.
- There are fewer sequence set index CIs because there are less CAs of one cylinder size. In sequential access all those records need to be read. Fewer index CIs means more effective channel programs.
- If you have allocated enough buffers, a large CA allows you to read more buffers into storage at one time. A large CA is useful if you are accessing records sequentially.
- The overlap between I/O and CPU for sequential processing is done in a CA boundary. When reached the application must wait until the last input/output to the CA is done before proceeding to the next CA. The I/O operations are always scheduled within CA boundaries.

The following disadvantages of a one-cylinder CA must also be considered:

- If there is a CA split, more data is moved.
- During sequential I/O, a large CA ties up more real storage and more buffers.

2.6.4 Partial release

Partial release is used to release data (not index) unused space from the end of an *extended format* data set. Refer to 1.9, “Extended format data set” on page 28. Partial release is specified through the SMS management class or by the JCL RLSE subparameter.

All space after the high used RBA (HURBA) is released on a CA boundary up to the high allocated RBA (HARBA). Refer to 3.8, “IDCAMS LISTCAT output fields” on page 187, to get more information on HURBA and HARBA. If the high used RBA is not on a CA boundary, the high used amount is rounded to the next CA boundary. Partial release restrictions are:

- Alternate indexes (AIXs) opened for path or upgrade processing are not eligible for partial release. The data component of an AIX when opened as cluster could be eligible for partial release.
- Partial release processing is not supported for temporary close. Temporary close (TYPE=T) means that the ACB is still opened, so there is no need of an Open to restart processing. However, the data set looks closed with EOF marks at the end.
- Partial release processing is not supported for data sets defined with guaranteed space.

- Extended format is a requirement.

2.6.5 Allocation constraint relief

Users occasionally encounter data set allocation or extension failures (X37 abends) because there is not enough space available on a volume to satisfy the request. SMS alleviates this situation to by performing volume selection, checking all candidate volumes before failing an allocation.

You can also use SMS data class the Space Constraint Relief and Reduce Space Up To (%) attributes to request that an allocation be retried, if it fails due to space constraints.

Recommendations

- Do not define a small amount of secondary space allocation for a KSDS or VRRDS data set.
- Allocate space at the cluster or data levels.
- Do not use tracks as an allocation unit; this forces the CONTIG attribute for secondary allocations.
- Avoid either primary or secondary allocation smaller than one cylinder, because it makes the CA size less than one cylinder.
- Consider partial release, if applicable.
- Consider constraint relief, if applicable.

2.6.6 Control interval size

There are two types of control intervals (CIs) in a KSDS and VRRDS: the index and the data. The other VSAM organizations only have data CIs. The data CI and index CI sizes are declared in the IDCAMS DEFINE command and kept in the catalog. If you define CI size for the cluster, this value is used for both data CI and index CI. In the following sections we offer recommendations about how to size both.

Data control interval size

There are arguments favoring both large and small CI sizes. Let us look at each of these possibilities:

- For sequential processing, larger data CI sizes are desirable. For example, given a 16 KB data buffer space to be fulfilled, it is faster to read two 8 KB CIs with one I/O operation than four 4 KB CIs with one operation.
- A large CI is a more centralized management of the free space CI, and consequently causing fewer CI splits. One component with one 16 KB CI with

20% of free space has less splits than a two 8 KB CIs with the same 20% of free space.

- For direct processing, smaller data CIs are desirable because the application only needs to retrieve one logical record at a time, then avoiding useless data transfer.
- For data set shared access within an address space, refer to 4.3.3, “Sharing data in a single VSAM control block structure” on page 218. The size of the CI affects the amount of data locked by VSAM; the smaller, the better.

In conclusion, for data sets accessed both randomly and sequentially, a small data CI with multiple buffers to help for sequential processing can improve performance.

Index control interval size

Usually you do not specify index CI size for KSDS and VRRDS. Let VSAM calculates it. Recall that VSAM uses one index record per each data CI and one index CI per each data CA. The size of the index CI affects:

- The number of levels in the index set. However, the number of indexes will effect the performance only for very large clusters that are greater than four levels.
- The usable capacity of a data CA. Sometimes a small index CI size may cause some loss in the usable capacity of the associated data CA. For the calculation of the CI index size, VSAM assumes a compressed key of 5 bytes. Sometimes the key does not compress well and this assumption is underestimated. Pay attention that at z/OS 1.3 the formula used by VSAM to calculate the index CI size changed. To see this modification and also to determine if your CI size is too small and because of this you are losing data CA space, refer to 4.2, “New Index CI size calculation algorithm” on page 208.

Recommendations

- For sequential access, define data CIs with 16 KB or larger.
- For random access, define data CIs with 4 KB.
- For mixed random and sequential access, define data CIs with 4 KB (for favoring random access) and plenty of buffer space in the buffer pool (allowing CCW chaining for sequential access).
- Let VSAM derive the size of the index CI. Define it yourself only when you are losing data CA capacity due to small CI index size.

2.6.7 FREESPACE definition for KSDS and ESDS

Freespace option specifies, through IDCAMS DEFINE (kept in the catalog), the percentage of each data CI and each data CA is to be set aside as free space when the cluster KSDS or VRRDS is initially loaded or when a mass insertion (writes skip sequential) is done. You can change the amount of free space using the IDCAMS ALTER command. Free space is specified as a percentage. If the FREESPACE amount is altered after the data set is initially loaded, and sequential insert processing is used, the allocation of free space is not honored.

The syntax of the FREESPACE parameter is:

```
FREESPACE(CI-percent CA-percent)
```

There is no free space external specification for index CIs or index CAs. Freespace is used to reduce the number of CI and CA splits along insertions or updating in-place records with a length increase.

When you specify free space in CI (first value in FREESPACE CI percentage), ensure that the CI free space percentage does not result in lots of unused free space. You can ensure this by taking into account the logical record length, the size of the CI, and the length of CIDF and RDFs. Following is a numeric example:

- Assuming a fixed length record data set, where each CI is 4096 bytes, 10 bytes are reserved for control information (2 RDFs and 1 CIDF). Each logical record has 1000 bytes and you want to reserve room for 10% inclusion.
- If you specify 10% of CI free space VSAM reserves 410 bytes (4096 * 0.10) for free space. The logical record space is 3676 (4096 - 420) bytes.
- Because the records loaded in the data set are 1000-byte records, there is only space for three records, leaving 1086 (410 + 676) for insertions. In this free space you can fit another logical 1000 byte record, so your free space setting is correct. You are loosing only 86 bytes of unused space. All this calculation is done for a non-compressed data set. For compression information, refer to 2.6.15, "Data compression" on page 94.

For CA free space, VSAM ensures that at least one CI per CA remains empty during loading when you declare a FREESPACE CA percentage amount other than zero.

Too much free space in a CI or CA can result in:

- Increased number of index levels be cause less data per data CI/CA, which affects run times for direct processing slightly.
- More DASD storage required to contain the data set.

- More I/O operations required to sequentially process the same number of records. Note that these extra I/O operations are only affected by an excess of CI free space. CA free space does not increase the number of I/O operations in a sequential read because totally free CIs are not moved to storage.

Too little free space can result in an excessive number of CI and CA splits (depending on the key pattern of the records to be inserted), with consequences such as:

- The CA splits are resource consuming, due to the overhead (during the split), since approximately half of the CIs from the CA are moved to the end of the data set.
- CI and CA splits may also affect the sequential processing because the DASD controller, to better use the cache, only detects 3390/3380 tracks physical sequence. Splits make the logical sequence different from the physical sequence.

Determine the amount of CI free space based on the percentage of record additions expected, and their distribution.

If you know in advance the pattern of the keys being inserted, you can take advantage of it by choosing the technique best suited for the application.

Recommendations

- Determine the amount of CI free space based on the percentage of record additions expected, and their distribution:

  No additions. If no records will be added and if record sizes will not be changed, there is no need for free space.

  Few additions. If few records will be added to the data set, consider a free space specification of (0 0). When records are added, new CAs are created to provide room for additional insertions.

  If the few records to be added are fairly evenly distributed, CI free space should be equal to the percentage of records to be added (FSPC (nn 0), where nn equals the percentage of records to be added.)

  Evenly distributed additions. If new records will be evenly distributed throughout the data set, CA free space should equal the percentage of records to be added to the data set after the data set is loaded. (FSPC (0 nn), where nn equals the percentage of records to be added.)

  Unevenly distributed additions. If new records will be unevenly distributed throughout the data set, specify a small amount of free space. Additional splits, after the first, in that part of the data set with the most growth will produce CIs with only a small amount of unneeded free space.

Mass insertion. If you are inserting a group of sequential records, you can take full advantage of mass insertion by using the ALTER command to change free space to (0 0) after the data set is loaded.

Additions to a specific part of the data set. If new records will be added to only a specific part of the data set, load those parts where additions will not occur with a free space of (0 0). Then, alter the specification to (n n) and load those specific parts of the data set.

2.6.8 Index options

There are two index options associated with a cluster defined in the IDCAMS DEFINE command and stored in the catalog respectively REPLICATE and IMBED. They exploit 3390/3380 characteristics.

Beginning with DFSMS 1.5 this parameter is no longer valid. No warning message is issued. Because of that they are not discussed here.

Recommendations

- Do not use REPLICATE.
- Do not use IMBED.

2.6.9 Key Range and Ordered

Key Range, for example, allows the user to select a volume to a set of keys in a collating sequence (another volume to another set). The major reason of that is to increase the parallelism of access, because you use multiple volumes. Ordered implies that the order of declared volumes must be followed along secondary allocation.

These options are not recommended anymore, because they place a burden in the user. It is recommended that they not be used on z/OS 1.3 systems and up, because they are not supported.

2.6.10 Share options

There are several mechanisms to implement VSAM data set integrity (read and write integrity). They are:

- Intra address space locks for serializing VSAM data sets among tasks of the same address space.
- VSAM SHAREOPTIONS play an important role in VSAM integrity, specifying whether and to what extent VSAM data sets are to be shared among tasks in one or multiple z/OS address spaces. This feature uses ENQ on SYSVSAM.data-set-name in order to achieve the required serialization.

- ENQ/Reserve serialization functions issued by the application.
- JCL disposition (OLD or SHR), which also implies in an ENQ.
- Record level sharing (RLS) locking mechanism, which is covered in “Define CF lock structures” on page 258.

The reason that we are covering these mechanisms in the performance chapter is because usually integrity and performance vary inversely. Total integrity may result in bad performance.

When you define VSAM data sets, you can specify how the data is to be serialized (if shared) within a single system or among multiple systems that can have access to your data. Before you define the level of sharing and the serialization mechanism for a data set, you must evaluate the consequences of reading incorrect data (a loss of read integrity) and writing incorrect data (a loss of write integrity). And the situations that can result when one or more do not adhere to guidelines recommended for accessing such shared data sets. On the other hand, it is important to avoid the unnecessary use of certain serialization functions which may cause a performance degradation.

Refer to 4.3, “Sharing VSAM data sets” on page 212, for more information.

Recommendations

- If you are sure that no application updates or deletes the VSAM data set, then do not use SHAREOPTIONS 4. It causes bufferpool refresh for direct reads or writes. The bufferpool is useless and no I/Os are saved.
- If you are doing your own serialization use ENQ SHARE instead of ENQ EXCLUSIVE for reads.
- In a cross-systems environment, avoid the use of the RESERVE macro, which locks the full 3390/3380 logical volume. Instead use the ENQ macro for a global resource with GRS. However, for applications already using the Reserve macro, it can be transformed (without changing the code) in a Systems ENQ through the GRS Conversion RNL.

2.6.11 Initial load option

Initial load mode occurs when you load *(write)* your data *sequentially* in a VSAM data set which has a High Used RBA (HURBA) equal to zero. The data set to be loaded is already IDCAMS defined (cataloged and described through the DSCB in VTOC). Initial load can be done through by IDCAMS REPRO, IMPORT or application program. There are two cases:

- After the DEFINE indicating that this is the first time you are writing into the data sets its initial contents, or

- When a data set, defined with the REUSE attribute, is opened with MACRF=RST (reset) in the ACB, meaning that you reusing the data set with new data.

You need to load a data set first, because in VSAM non-RLS mode, your application program cannot open for input to an empty data set.

Initial load uses a NSR buffering, refer to “Non-shared resources (NSR)” on page 68.

RECOVERY and SPEED options

There are two options where performance of an initial load process may be affected, respectively RECOVERY and SPEED. Those options have to do with the techniques used by an access method for creating physical records:

- RECOVERY. This option first formats a 390/3380 track (erases the previous contents) by executing the Write Count Data CCW (also called Write Format) creating physical records full of zeroes in the Data part (of a CKD physical record). These zeroes mean a VSAM end-of-file indicator.
- The data portion of tracks are filled with real data by executing the Write Data CCW (Write Modified). The I/O response time for the load almost doubles. Read “DASD cache highlights” on page 126, for more information on Write Format and Write Modified CCWs. The tracks beyond HURBA are not formatted.
- The Recovery option allows the restart of the load operation from the last written record (just before the first end-of-file physical record), however other difficulties such as tape repositioning (in the input file) have inhibited customers from using it.
- SPEED: The data CIs/CAs are not pre-formatted with zeroes. The Write Count Data CCW is executed (in just one pass) creating the physical record with real data in the data part. An end-of-file indicator is written only after the last record is loaded. As in the Write Modified type of channel program (in RECOVERY), many writes are assembled in the same channel program to improve CPU and I/O usage (depending on the number of NSR data buffers). However free CIs (if you declare CAs with free CIs) are not written together with occupied CIs in the same channel program.

Table 2-1 this table compares the RECOVERY and SPEED options

Initial Load	Extended Format	EXCP	Total Connect Time (msec)	CPU TIME (sec)	Elapsed time (sec)
Recovery	Yes	22276	36881	7.8	64

Initial Load	Extended Format	EXCP	Total Connect Time (msec)	CPU TIME (sec)	Elapsed time (sec)
Recovery	No	20569	33156	7.7	60
Speed	No	8604	16993	7.5	35
Speed	Yes	8605	20079	7.6	38

Because it has just one pass, with SPEED you get better performance for initial load mode. Refer to Table 2-1.

Be aware that during load mode processing, you cannot share data sets. Share options are overridden during load mode processing to (1 3). When a shared data set is opened for create or reset processing, your program has exclusive control of the data set within your operating system by the use of the ENQ Systems exclusive in the SYSVSAM resource. If data sets are shared between systems, never place this resource name in the exclusion list.

There are other VSAM options affecting the initial load performance:

- System managed buffering (SMB) due to better buffer management.
- Extended format; refer to 1.9, “Extended format data set” on page 28.

Recommendations

- Do not use the RECOVERY option.
- Use SMB.
- Use extended format.
- Remember that any specific SHAREOPTIONS assignment in an initial load is forced to (1 3).

2.6.12 Region size

The key to reducing the number of I/O operations is to keep more data in virtual storage. The recommendations given here go in this direction. So, before you implement them, you should review the region size parameter to avoid an S878 type of abend, as shown in the following VSAM message:

```
IDC3351I ** VSAM CLOSE RETURN CODE IS 136
Not enough virtual storage was available in the program's address space for
a work area for Close
```

Figure 2-3 VSAM message indicating too small a region size

The portion of the user's private area within each virtual address space that is available to the user's programs is called the *user region* (located from the bottom to top of the private area). The *region size* is the amount of storage in the *user region* available to the job, started task, or TSO/E user. Figure 2-4 shows the virtual storage layout. For a description of each area, refer to the *OS/390 MVS Initialization and Tuning Guide*, SC28-1751.

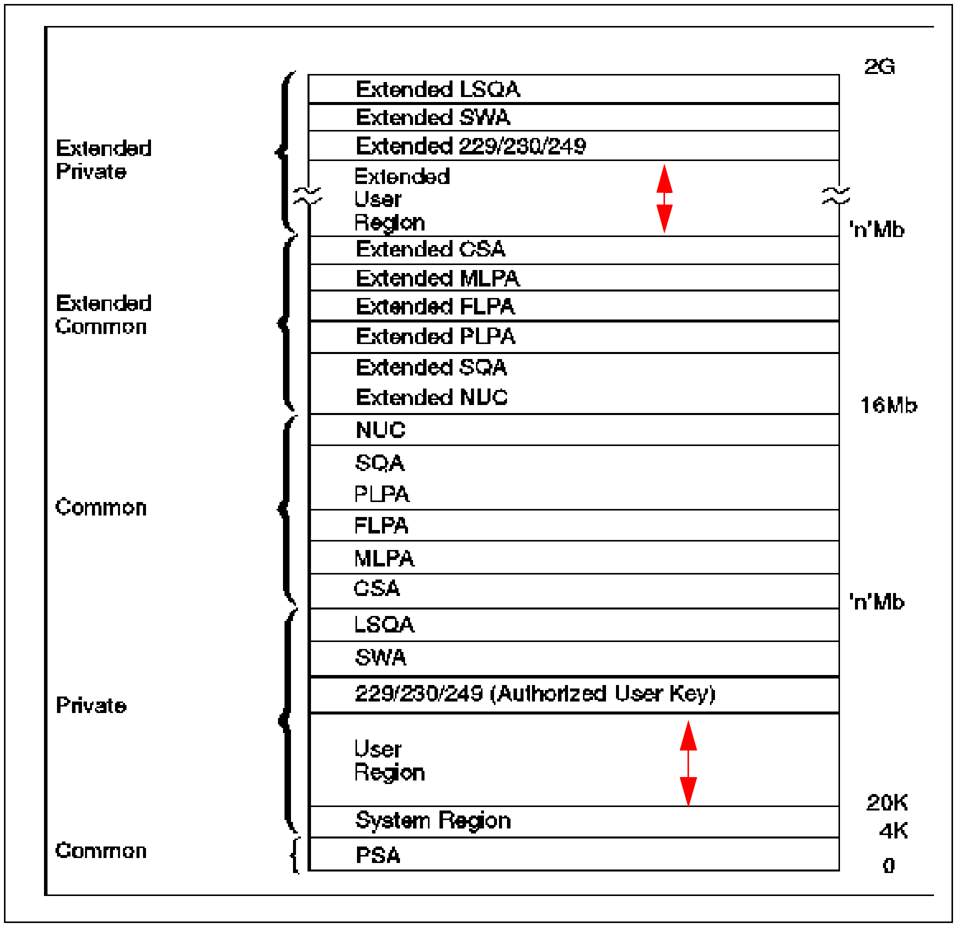


Figure 2-4 Address space layout

You specify a job's region size by coding the REGION parameter in the JOB or EXEC statement. The system rounds all region sizes to a 4K multiple. Some jobs run out of virtual space and abend when:

- The REGION specified is greater that the available private area.
- A private area GETMAIN reaches the limit determined by the IEFUSI exit (the default is the specified REGION plus 64K) even if there is virtual free space available.
- A private area GETMAIN cannot be served because no contiguous virtual free space exists with the required size. This means a collision between the top-to-bottom and bottom-to-top subpools.

Refer toTable 2-2 to understand how the JCL REGION parameter is interpreted and how much virtual storage is available, below and above 16 MB, for each step of a job.

Do not hesitate to increase your job region size, or even avoid it by specifying REGION=0. Good buffering can reduce the number of I/Os, job elapsed time, CPU time and device connect, and disconnect time. If your job is I/O bound, then by giving it enough resources, it executes more quickly. That is a benefit for the user and for total system performance.

Table 2-2 Region JCL parameter

REGION value	Region available below 16 MB	Region available above 16 MB
0k > REGION < 16 MB	Establishes the private area size below	32 MB
16 MB > REGION =< 32M	All private area below is available	32 MB
32 MB > REGION =< 2G	All private area below is available	Establishes the private area size available above 16 MB
0K or 0M	All private area below is available	All private area above is available

If your job is experiencing constraints in virtual storage below 16 MB when VSAM data sets are opened, you can relieve storage usage below 16 MB by specifying that VSAM allocate the buffers and/or control blocks above 16 MB. For details, refer to “Locating VSAM buffers above 16 MB” on page 81.

2.6.13 Buffering options

Buffering in virtual storage is an important technique in VSAM to improve performance. You have the possibility of using this technique through the buffering options.

A VSAM resource pool is a set of VSAM control blocks plus a buffer pool. These control blocks are not enough to allow the data set to be processed. At open time more control blocks are created, which together with the buffer pool form a structure control blocks. Here are several key aspects for managing a buffer pool:

- The algorithm to keep CIs in the buffers. For random it is LRU, the most accessed CIs are kept in buffer. For sequential access, it uses the sequential algorithm, which has two pieces: look ahead for reads and to dispose of any CI already processed (reads or writes) to make room for the new ones.
- What to do with the CIs updated in the buffers. Should they be stored through in DASD immediately or later? For random, there is an option called Defer Write, where the installation decides what to do. For sequential, VSAM defers the write until half of the buffers are ready for the write. VSAM Shareoptions may change this behavior. Refer to "Share options" on page 58.
- The amount of buffers can degrade the performance. For random it is recommended to have all the index CIs and lots of data CIs in the buffer pool. For sequential just one index CI buffer (sequence set) and many data CIs buffers for the look ahead.
- When the same cluster is shared between two applications (for read/write) in different MVS systems, the coherency of the buffer pool must be guaranteed. That is, if a CI is altered in a local buffer pool, the other local buffer pool should reflect such change.

When a buffer's contents are written, using direct access, the buffer's space is not released. The control interval remains in storage until overwritten with a new control interval. If your program refers to that control interval, VSAM does not have to reread it. VSAM checks to see if the desired control interval is in storage. This is not valid for share option 4, where buffers used for direct processing are refreshed for each request.

Buffer space is released when all data sets that are using the buffer pool are closed, and for the LSR/GSR, the DLVRP macro is issued. If an abend occurs before closing VSAM data sets, the buffers are not flushed by VSAM or MVS Recovery Termination Manager routines. It is left to the application decision throughout the use of an (E)SPIE/(E)STAE routine to close the data sets and consequently flush the buffers. For details about ESTAE and ESPIE macros, refer to *OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-IXGWRITE)*, GC28-1765. Pay attention that such routines do not gain the control when the abend is caused by an operator CANCEL command. Also there are options (TRAP) in Language Environment® to control the flush or not flush of those buffers.

Choosing the adequate VSAM buffer length and buffering technique is the key to reducing the number of I/Os and reducing the I/O response time (Tr). More buffers (either data or index) than necessary might cause excessive paging or excessive internal processing. There is an optimum point at which more buffers do not decrease the job elapsed time and device connect time. You can see that in Table 2-4 on page 71. You should attempt to have data available just before it is to be used. If data is read into buffers too far ahead of its use in the program, it can be paged out.

For more efficient use of virtual storage, buffer pools can be shared among data sets using locally or globally shared buffer pools. There are four types of resource pools management, called modes, according to the technique used to manage them:

- Non-Shared Resource (NSR)
- Local Shared Resource (LSR)
- Global Shared Resource (GSR)
- Record Level Shared (RLS)

Those modes are not an attribute of a cluster, but an access option as interpreted by the Open routine. The same cluster can be opened in NSR by a task and in LSR by another task, or the same task can open a cluster initially as LSR and later as NSR. You see details about each one in this chapter, except for RLS. The exploiter of RLS is mainly CICS.

Buffers are acquired dynamically usually when the data set is opened. However with LSR/GSR they are acquired by the application program before the Open through the BLDVRP macro. Also they can be allocated after the Open through the Dynamic String Addition function. The amount of space for buffers is based on parameters in effect when the program opens the data set.

One of the major sources for determining how much space should be allocated in a buffer pool is the Access Control Block (ACB). The system obtains buffer pool information for VSAM data sets as follows:

- DD statements overrides SMS data class information.
- SMS data class
- The program's ACB
- The catalog entry
- For LSR buffering, the BUFND, BUFNI, BUFFERSPACE and STRNO parameters do not apply and cannot be overridden by JCL.

Table 2-3 has the parameters that influence buffer pool allocation.

Table 2-3 Parameters affecting buffer allocation

Source	**Parameter**
DD Statement	BUFSP, BUFND, BUFNI, and ACCBIAS

SMS data class (ACDS)	RECORD ACCESS BIAS (ACCBIAS)
Program’s ACB	MACRF=(IN|OUT, SEQ|SKP, DIR)
	STRNO=n, BUFSP=n, BUFND=n, BUFNI=n **(1)**
	SHRPOOL=n **(2)**
The catalog entry for the data set	BUFFERSPACE
Note: (1) Applies only to NSR. (2) Only for LSR/GSR, connect the ACB to an existent resource pool	

You can see how specifications in the MACRF, ACB’s parameter, affects buffering and data set processing:

- Buffering management: NUB for management of I/O buffers to be done by VSAM or UBF when management of I/O buffers is left up to the user, or a VSAM exploiter as DB2. NUB is the default value.
- VSAM buffering modes to be used: NSR (default), LSR, GSR or RLS. These modes assigning buffers to strings, so the number of strings also affects the buffering
- The manner in which the records are intended to be accessed: Direct (DIR), sequential (SEQ), skip sequential (SKP).
- Type of argument used to access records: By key (KEY option), by RBA (ADR option), by RRN, or access is to the entire contents of a control interval rather than to an individual data record (CNV). KEY is the default.
- What kind of processing is done in the data set: Input (IN, default) or output (OUT); this is just an intention.
- How writes are to be managed: defer writes (DFR) or not (NDF). For details about deferring writes; refer to “Deferring write requests” on page 80.
- Using LSR, how VSAM deals with conflict for an exclusive control in buffers: VSAM defers the request until the resource becomes available (LEW, default) or VSAM returns the exclusive control return code X’14’ to the application program (NLW). The application program is then able to determine the next action.
- Insert record strategy: Split CIs and CAs at the insert point (SIS) or at the midpoint (NIS, default) when doing direct PUTs.

In MACRF at ACB, you code the types of access you intend to use during the processing. They are used mainly for buffering management and at open time. To process the data set, you use request parameter lists (RPL), where you specify only the processing options appropriate to that particular request.

If you open a data set whose ACB includes MACRF=(SEQ,DIR), buffers are allocated according to the rules for sequential processing, NSR buffering management.

BUFFERSPACE, BUFSP, BUFND, BUFNI affects buffering

The BUFFERSPACE value in the catalog entry for the data set is the *minimum* amount of buffer space while the value assigned to BUFSP, in JCL or ACB, is the *maximum* amount of buffer space. The BUFFERSPACE value applies to the whole cluster. Additional buffer space can be assigned to any data set by:

- Modifying the data set's BUFFERSPACE value
- Specifying a larger BUFSP value with the AMP parameter in the data set's DD statement

If you do not specify BUFSP, the amount of virtual storage used for buffers is the *largest* of these values:

- The amount specified in the catalog (BUFFERSPACE)
- The amount determined from BUFND and BUFNI
- The minimum storage required to process the data set with its specified accessing options, as sequential, direct

The BUFSP value takes precedence over BUFNI and BUFND as follows:

- If the number of buffers specified in the BUFND and BUFNI subparameters exceed the virtual storage specified in the BUFSP space, the number of buffers is decreased to fit in the BUFSP space as follows:
  - If the ACB indicates direct access only, first the number of data buffers is decreased until it reaches the BUFSP value, but it never becomes less than the minimum required. If the BUFSP value is not reached, then the number of index buffers is decreased until BUFSP is reached.
  - For sequential access, BUFNI is decreased to reach BUFSP up to the minimum plus one. If not reached, BUFND is decreased. When the minimum is reached, but BUFSP is not reached, than one buffer is subtracted from the number of index buffers.
- If BUFSP specifies more space than is required by BUFND and BUFNI, the number of buffers is increased to fill the BUFSP space as follows:
  - For direct access only, additional index buffers are allocated.
  - For sequential access, one additional index is allocated and as many data buffers as possible are allocated.

If BUFSP is specified and the amount is smaller than the minimum amount of storage required to process the data set, VSAM cannot open the data set.

When processing a data set using a path, the number of needed buffers increase, since buffers are needed for the alternate index, the base cluster, and any alternate indexes in the upgrade set.

When a base cluster is opened for processing with its alternate index, the BUFSP, BUFND, BUFNI, and STRNO parameters apply only to the path's alternate index. The minimum number of buffers are allocated to the base cluster unless the cluster's BUFFERSPACE value (specified in the DEFINE command) or BSTRNO value (specified in the ACB macro) allows for more buffers. VSAM assumes direct processing, and extra buffers are allocated between data and index components accordingly.

The default for the number of VSAM allocation buffers is as follows:

- For the index component: BUFNI=STRNO
- For the data buffers: BUFND=STRNO+1

One of the data buffers (BUFND) is used only for formatting CAs and splitting CIs and CAs. Only data buffers are needed for ESDS, RRDS, or LDS.

This default is also the *minimum* number of buffers required by VSAM.

Recommendations

Do not specify BUFSP or BUFFERSPACE, take the default value. This avoids mistakes in calculation leading to different results from those expected. For NSR use SMB and for LSR it is an exploiter decision.

2.6.14 Buffering techniques

Here is a look at various buffering techniques.

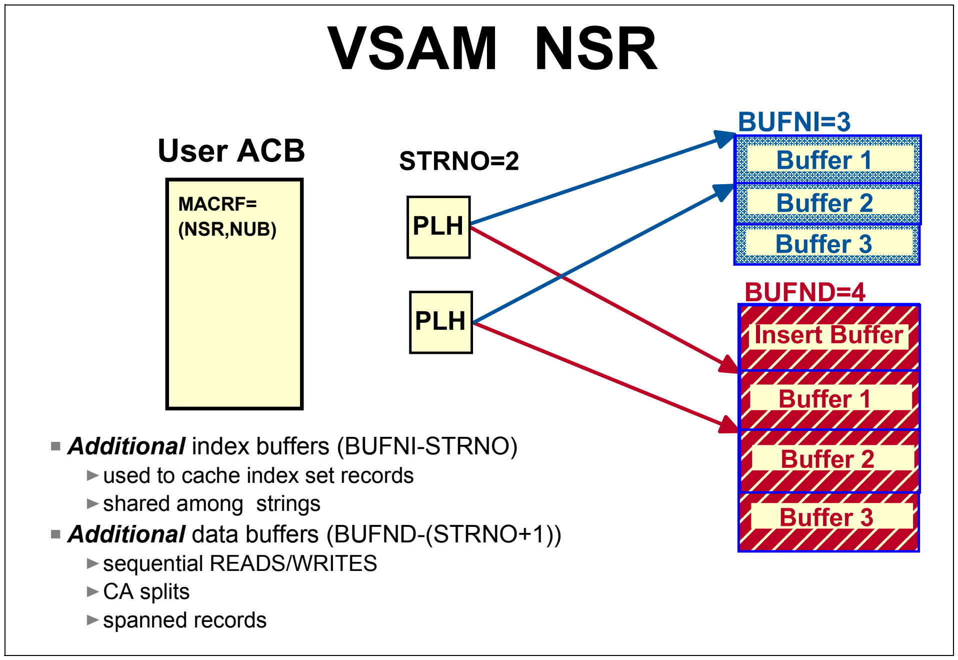
Non-shared resources (NSR)

Non-shared resources (NSR) is the default VSAM buffering technique. It has the following characteristics:

- The buffers are *not* shared among VSAM data sets.
- The buffers are located in the private area.
- It is suited for sequential processing, because the buffers are managed via a sequential algorithm:
  - For sequential processing, there is look-ahead (time permitting CIs not yet requested are brought to buffer pools), and CIs in the buffer pool are not managed by LRU (meaning after use by the application program, they are strong candidates to leave the buffer pool).

  - For direct processing, there is no look-ahead (which is good), but CIs are not managed by LRU (which is bad).
- It is used by high level languages.
- The resource pool (buffer pool plus control blocks) is built automatically by OPEN accordingly to BUFND, BUFNI, BUFFERSPACE, BUFSPC.
- NSR has specific properties when assigning buffers to strings (or place holders).
  - Each string has its own buffer for the index sequence set component. Additional index buffers provided (allowed by the buffer parameters) are used to cache index set records, shared among strings
  - Each string has its own buffer for the data component. If additional data buffers are provided, they are for:
    - Sequential reads and writes
    - CA splits
    - Spanned records
  - There is also a buffer for insert records is used only along CI splits.
- It dynamically extends the number of strings as they are needed by concurrent requests for the ACB.
- For subtask sharing, when a CI is not available for the type of task processing requested, VSAM under NSR buffering has a proper way of managing the contention. For details, refer to 2.6.10, “Share options” on page 58.

Figure 2-5 shows how NSR buffers are constructed.



- ***Additional*** index buffers (BUFNI-STRNO)
  - used to cache index set records
  - shared among strings
- ***Additional*** data buffers (BUFND-(STRNO+1))
  - sequential READS/WRITES
  - CA splits
  - spanned records

Figure 2-5 NSR buffering

NSR with sequential access

NSR is best used for applications that use sequential or skip sequential as their primary access mode.

At initial buffer pool loading for reads on the behalf of a string, VSAM NSR loads the buffers using just one channel program with chained CCWs. The number of CIs in the channel program is equal to the number string’s assigned buffer plus *all* additional buffers to a *maximum* of CIs/CA, since I/Os are scheduled on CA boundaries.

When another string, at the same time, needs buffers, VSAM uses its assigned buffer and the remaining buffers to a maximum of CIs/CA, and so forth. So, having more buffers than CI/CA plus one is useful only when having more than one string. When a string finishes using its buffers (processed by the application program associated with the string), additional buffers are available to other strings.

For overlap between CPU and I/O after initial buffer pool load:

- For sequential reads, once one-half of the buffers is being processed by the application program, an I/O operation is scheduled for this half. This continues until a CA boundary is encountered and the application must wait until the last

I/O to the CA is done before proceeding to the next CA. The I/O operations are always scheduled within CA boundaries.

- For sequential writes, once one-half of the buffers are filled by the application program with logical records, an I/O operation is scheduled for this half. Then for sequential PUT processing, VSAM NSR does not immediately write the updated CI from the buffer unless a CI split is required. VSAM saves I/O operations by deferring sequential writes. For details about deferred write (sequential and direct), refer to “Deferring write requests” on page 80.

When you are accessing data sequentially, you can increase performance by increasing the number of data buffers, up to a certain limit. When there are multiple data buffers, VSAM uses a read-ahead function to read the next data control intervals into buffers as buffers become available as described above.Table 2-4 shows results of tests varying the number of buffers.

Table 2-4 NSR: Read sequential varying the number of buffers.

Data buffers	**Index buffers**	**EXCPs**	**Device connect time**	**SRB time**	**TCB time**	**Elapsed time**
Default=2	Default=1	167828	23.50	2.12	7.52	120
10	1	33568	14.48	0.53	3.99	36
30	1	11577	11.82	0.26	3.44	22
50	1	6948	11.40	0.21	3.35	19
50	2	6948	11.40	0.21	3.37	19
90	1	4633	11.26	0.18	3.44	18
181	1	3476	11.15	0.19	3.86	14
181	3	3476	11.19	0.19	3.91	19
10000	1	3476	11.16	0.19	3.89	19
SMB 01 Stripe		4633	15.75	0.23	3.45	22
SMB - 02 Stripes		7580	17.11	0.17	3.61	15
SMB - 04 Stripes		14073	19.3	0.17	3.65	11
SMB 08 Stripes		27061	24.30	0.17	3.77	10
Note: All times are in seconds						

As more buffers are used, the number of Execution Channel Program (EXCPs) is reduced. This means there are fewer I/O interrupts. That is why SRB time consumption drops so much compared with TCB time. For a discussion about what is a VSAM EXCP in numerical values, refer to Appendix B, “Miscellaneous performance items” on page 395. With VSAM, the number of EXCPs is equal to the number of SSCH instructions, that is the number of I/O operations, and not the number of transferred CIs or transferred physical blocks.

- The TCB time drops due to decrease in I/O preparation. After the optimum point, the TCB time increase due to excessive buffering management processing. Remember that our program is just a *read and forget,* so all the TCB time is consumed in the I/O process. We recommend you read the in section “Our test environment” on page 396, for a better understanding of the test results shown in this book.

With SHAREOPTIONS 4, buffers are refreshed at each request. Also, the read-ahead (a look-ahead synonym) function has no effect and defer write is not used. Therefore, for SHAREOPTIONS 4, keeping data buffers at a minimum can actually improve performance.

The POINT macro does not cause read-ahead processing unless RPL OPTCD=SEQ is specified. POINT positions the data set for subsequent sequential retrieval.

Having only one index I/O buffer per string does not hinder performance in a read sequential access because to access a logical record, VSAM gets to the next index sequence set and uses it for the complete data CA. At end of the CA by using the horizontal pointers, rather than the vertical pointers in the index set, the next index sequence set CI is retrieved. Extra index buffers have little effect during sequential processing. You can see that in Table 2-4.

When reading sequential, buffers are used to balance the ability of the application to process data with the capacity of the storage device to deliver the data to the application. One of the factors, that affects the amount of data CI buffers needed is the number of logical records per data CI, and how heavy is its processing. In a data CI with many records, the read data buffers are saturated before a data CI with few records. Saturated means you do not get better performance by increasing the number of buffers. In this case the CPU processing become the slower factor.

By specifying enough data buffers, you can access the same amount of data per I/O operation with small data CIs as with large data CIs.

Table 2-4 contains the results from our lab tests. Remember that, in our lab tests, the data set record processing is minimal (very I/O-bound). Also notice that we did not run in a controlled environment, so the elapsed time value can vary according to priority and system workload. The numeric values are total and not an average per I/O operation.

If you experience a sequential performance problem waiting for input from the device, you should specify more data buffers to improve your job's run time. More data buffers allow you to do more read-ahead processing and less I/O operation for the same amount of data, which decreases the disconnect time.

If your data set is SMS managed, NSR utilizes extended format, therefore you do not need to worry about how many buffers to specify for the index or data component. This means that you can take advantage of system managed buffering. For details, refer to “System managed buffering (SMB)” on page 82.

Buffering in NSR initial load mode

Initial load mode occurs when you load your data *sequentially* in a VSAM data set which has a High Used RBA (HURBA) equal to zero. Initial Load uses a NSR buffering option.

Pay attention to performance aspects of using RECOVERY or SPEED attributes, Refer to 2.6.11, “Initial load option” on page 59.

There is a difference with index buffers when you compare reading and initial loading (or extending) sequentially a VSAM KSDS cluster. In the first case VSAM never refers to the index set, then there is no need to set aside buffers for such index CIs. However, along Initial loads the index set CIs are built, then there is a little gain in performance if you provide some buffering for them.

With extended format data sets and SPEED (writing each CA with just one I/O pass) you can get much better performance using SMB. For details, refer to “System managed buffering (SMB)” on page 82.

Table 2-5 shows the results of loading an empty data set, with 2,000,000 records, using IDCAMS REPRO. The best result is obtained when:

Data buffers = `181 = CIs/CA + 1`

Index Buffers = **`3`** (the number of index levels after data set loading)

Table 2-5 NSR - Initial Load mode varying the number of buffers

Data buffers	Index buffers	Extended Format / stripes	EXCPs	Device connect time (sec)	CPU Time (sec)	Elapsed time (sec)
Default	Default	No / 1	27856	41.2	8.65	68
90	Default	Yes / 1	13970	24.9	8.08	41
181	Default	Yes / 1	13969	24.5	8.08	42
181	3	Yes / 1	12810	24.6	8.08	44

Data buffers	Index buffers	Extended Format / stripes	EXCPs	Device connect time (sec)	CPU Time (sec)	Elapsed time (sec)
360	Default	Yes / 1	13969	25.5	8.07	44
10000	3	Yes / 1	13971	24.3	8.06	42
SMB	SMB	yes / 1	12813	28.4	8.05	48
SMB	SMB	yes / 2	14153	28.6	8.18	30
SMB	SMB	yes / 4	11991	20.2	8.2	28
SMB	SMB	yes / 8	22815	34	8.24	22

NSR with direct access

NSR is not intended for direct or random access. However, many of your applications may use NSR for direct processing because it is simple. In this topic we look at how NSR works for direct access. Remember that today you have solutions available to avoid NSR direct processing without changing your code: system managed buffering, and batch local shared resources. We cover their functions later in this chapter.

In direct access, records are randomly accessed by key, RBA or RRN depending on the data set organization. Increasing data buffers do not boost performance, because VSAM NSR does not implement an LRU algorithm to manage the buffer pool.

Two good aspects of NSR read direct processing are:

- Does not use look-ahead buffers
- Only reads one CI per GET request

For direct output processing (PUT for update), VSAM defers write *only* when OPTCD=NSP option is specified in the RPL macro; otherwise, VSAM immediately writes the updated CI.

Within a NSR buffer pool, data and index sequence set buffers are not shared among strings, refer to “Non-shared resources (NSR)” on page 68. Each string is associated with one request parameter list (RPL). For example, when an application uses an RPL to issue a direct GET for a record with a key of 1234 using NSR, VSAM manages buffers as follows:

1. VSAM locates the correct CI, though a top-down search through the index set.
2. VSAM then reads the data CI into storage and gives the user the requested record.

3. If the user then issues another direct GET for a record with a key of 6789, which is in a different CI from record 1234, the same data buffer gets used for this request, and the CI containing 6789 overlays the CI containing 1234.
4. If the user issues another direct GET for a record with key 1235 (which is in the same CI as 1234), that CI must be read in again because the intervening GET for 6789 causes the previous buffer contents to be lost.

This problem cannot be solved by using multiple strings. If, for example, the requests for 1234, 6789, and 1235 use three different RPLs, both CIs are in storage when the request for 1235 occurs, but VSAM looks *only* in the buffer assigned to the string related to RPL requesting 1235 and does not look in another string's buffers to satisfy the request. The CI has be read in again anyway.

This NSR characteristic leads to performance complaints in cases where the processing is random, but the same CI is requested multiple times (re-visiting) with intervening requests to other CIs.

When the number of I/O buffers provided for index records is greater than the number of strings, the surplus is shared among the strings:

- One buffer is used for the highest-level index record.
- Additional buffers are used, as required, for other index set index records, as shown in Figure 2-5 on page 70.

For a KSDS or VRRDS, with NSR direct access, you should provide at least enough index buffers. It needs to be at least to the value of the STRNO parameter of the ACB plus one, if you want VSAM to always keep the highest-level index record always resident. You can increase performance by going beyond such number. Unused index buffers do not degrade performance. Direct processing always requires a top-down search through the index.

Then, for optimum performance, the number of index buffers should at least equal the number of high-level index set CIs plus one per string. This makes the entire high-level index set and one sequence set CI per string in virtual storage. Table 2-6 shows how adding index buffers improves performance. The elapsed time can vary according to the system workload. Note that additional index buffers are not use for more than one sequence set buffer per string.

VSAM NSR reads index buffers one at a time (per I/O operation). Index buffers are loaded when the index is referred to. When many index buffers are provided, index buffers are not reused for another index CI, until a requested index CI is not in storage.

VSAM NSR keeps as many index set records as the buffer space allows in virtual storage. Ideally, the index would be small enough to allow the entire index set to remain in virtual storage. Then, you should be aware of how index I/O buffers are used so you can determine how many to provide.

Many data buffers do not increase performance, because only one data buffer is used for each access, as you can see in Table 2-6. The elapsed time can vary according to the system workload.

Table 2-6 NSR buffering with direct access; STRNO=1

Data Buffers	Index Buffers	EXCPs	Device Connect time (sec)	CPU time (sec)	Elapsed time (sec)
Default	Default	794950	103.34	31.76	506
90	1	794950	103.34	32.42	506
Default	3	505171	76.67	21.11	326
Default	5	445160	62.32	18.87	291
Default	50	368823	51.64	16.36	242
Default	1200	368823	51.64	17.13	243

If your job is having performance problems randomly accessing VSAM data sets in NSR mode, you can improve performance with no changes in your applications, as follows:

- If your data set is SMS managed, has extended format, and your installation has DFSMS V1R4 or later, you can use system managed buffering. For details, refer to “System managed buffering (SMB)” on page 82.
- If the above conditions are not met, you can use BLSR. For details, refer to “Batch local shared resources (BLSR)” on page 92.

Refer to “Sample programs extract from SMF record type 64” on page 367 for information that can help you find data sets which are candidates to use SMB or BSLR. Both offer real gains for direct access.

If your data set meets both requirements for SMB and BLSR, use SMB, for better results.

Recommendations

For NSR, we recommend:

- For sequential processing, use SMB to get an optimum buffering, or:

  BUFNI = Number of levels

  BUFND = Number of CI/CA plus one

Additional data buffers, when STRNO higher than one

- For direct access, use SMB or BLSR to convert to LRU algorithm, or in a worst case keep NSR with:

  BUFNI = STRNO plus number of levels

  BUFND = Let the default value (STRNO plus one)

NSR with mixed access

The best advice here is still to suggest SMB or BLSR. If you need to keep NSR then, for mixed access situations (sequential and direct), you can improve performance by doing the following:

- Increase the number of index component buffers to the number of index levels, to favor direct access. Table 2-6 shows how the number of index component buffers can improve performance for direct access.
- Increase the number of data component buffers, to favor sequential access. Table 2-4 on page 71 shows how this can improve performance, based on tests in our lab.

Local Shared Resources (LSR)

This is another mode of managing VSAM buffers. In this mode, the buffers in a LSR pool:

- Are shared among VSAM data sets accessed by tasks in the same address space. It is a centralized way for buffer management. Instead of having individual buffer pools (some very much utilized, some not), we have a large one shared by several VSAM data sets, improving the utilization of the resource.
- Are explicitly constructed through BLDVRP macro, before the OPEN for the first data set that uses it.
- Are explicitly deleted by the DLTVRP macro.
- Are located in the private area and ESO hiperspace (if specified in BLDVRP macro).
- Its CIs are replaced based on the least recently used (LRU) algorithm, which is designed for random processing.
- Have no look-ahead, even for sequential processing.
- Usually implemented only by CICS.
- For subtask sharing, when a CI is not available (locked) to the task for the type of processing requested, VSAM under LSR and GSR buffering has a proper way of managing the contention. For details, refer to 2.6.10, “Share options” on page 58.

- Applications using LSR can invalidate BP contents through MRKBFR macro and force them to be written immediately through WRTBFR macro.

LSR relieves virtual storage constraint and reduces I/O for applications that access the same data multiple times. This technique is best used for truly random access, with multiple references to the same data. Subsequent access to data does not have to go through DASD.

With LSR (and GSR), the number and size of buffers are specified in BLDVRP macro (see Figure 2-6) and are not overridden by ACB or JCL. The buffer pool is identified by a number and a data set is connected to it through the ACB macro, where the buffer pool ID is specified. After all data sets using a resource pool are closed, the resource pool can be delete issuing the DLVRP (delete VSAM resource pool) macro. For more details about macros, refer to *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913.

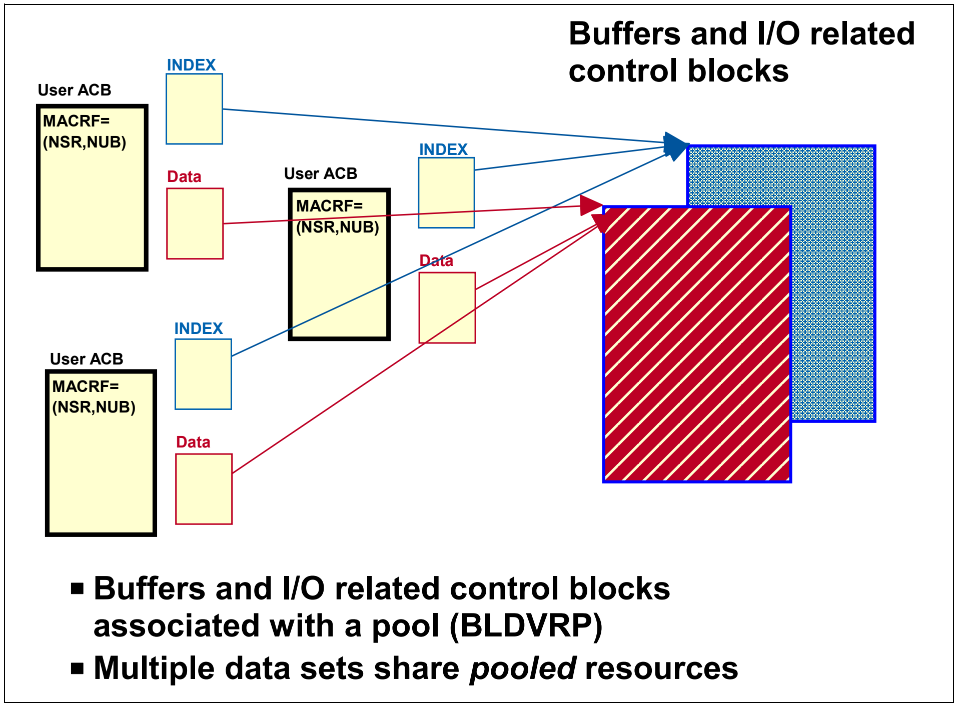

Buffers and I/O related control blocks
User ACB
MACRF=(NSR,NUB)
INDEX
Data
User ACB
MACRF=(NSR,NUB)
INDEX
Data
User ACB
MACRF=(NSR,NUB)
INDEX
Data
■ Buffers and I/O related control blocks associated with a pool (BLDVRP)
■ Multiple data sets share *pooled* resources

Figure 2-6 VSAM shared resources (LSR/GSR)

Online transaction program (OLTP) applications like CICS and IMS are the biggest users of LSR shared resources because they typically need to have hundreds of data sets open in one address space at any given time. Having an individual buffer pool for each data set would be a waste of virtual storage. With

CICS and IMS, the BLDVRP is issued by them, not directly by the user application. Information for building the shared buffer pool is specified in the product FCT table (for CICS). Refer to the product manuals.

With LSR and GSR, writes can be deferred until VSAM needs a buffer to satisfy a GET request. Deferring writes saves I/O requests in cases where subsequent requests can be satisfied by the data already in the buffer pool. For more details, refer to “Deferring write requests” on page 80.

The search for a CI in the buffer pool is not affected by the pool size once the search is by hashing, so there is no overhead. Also, with LSR buffering, your application can use hiperspace as a second level of buffering. If you intend to build an application using LSR buffering techniques:

- Note that LSR is suited for direct access; for sequential access, use NSR.
- Build resource pools before any open to the data sets that use them.
- Build a separate resource pool for indexes to avoid index data being flushed by data being read.
- Use defer write if possible.

For details on how to build an LSR pool, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

LSR with direct access

LSR buffering mode is designed for direct access. If your applications use NSR buffering and direct access, and you are having performance problems, you can take advantage of LSR buffering techniques using SMB or BLSR.

Refer to “System managed buffering (SMB)” on page 82, and “Batch local shared resources (BLSR)” on page 92.

LSR with sequential access

As we said, LSR buffering mode is designed for random access, not sequential. If you are relying on sequential look-ahead (also called read-ahead) for good performance LSR buffering could degrade, rather than improve performance. LSR does not do read-ahead. Your sequential request is read in just one data CI per I/O operation. You should use NSR buffering, where VSAM can read up to a CA's worth of data CIs in a single I/O, conditions permitting. For details about NSR, refer to “NSR with sequential access” on page 70.

Recommendations

If you intend for your batch application to use the LSR buffering technique, we recommend you:

1. Write the application using the default buffering NSR. It is easier and you can use a high level language, such as COBOL.
2. Use SMB to convert to LSR when processing or BLSR if there is no SMS.

Global Shared Resources (GSR)

GSR is similar to LSR buffering technique. The GSR characteristics that differ from LSR are:

- The buffer pool is shared among VSAM data sets accessed by tasks in *multiple* address spaces in the same system image.
- Buffers are located in CSA.
- The code using this must be in supervisor state.
- Buffers cannot use hiperspace.
- The separate index resource pools are not supported for GSR.

With these differences in mind, refer to “Local Shared Resources (LSR)” on page 77, for details on how GSR buffering technique works.

GSR is not used by any of the main VSAM exploiters.

The *disadvantages of* GSR are:

- Use of common area, a limited resource in the system
- The address space that opened the data set for the first time must remain started to allow acquire new space to the data set.
- The allowed number of resource pools is one by storage key number (0-7).
- All VSAM requests related to the global resource pool may be issued only by a program in supervisor state with the same protection key as the resource pool.
- Workload balancing: you *cannot* distribute the workload between system images and keep a single control block structure.

Recommendations

Do not use GSR mode for buffering, use RLS.

Deferring write requests

Specify defer write, if possible in the ACB. With defer write, VSAM uses the buffer pool in a *store in* mode. That is, the updates are not propagated immediately to

DASD. They are deferred until the WRTBFR macro is issued or until VSAM needs a buffer to satisfy a GET request.

The performance advantages of deferring writes are these:

- If the same record is updated *n* times and is then written asynchronously to DASD, you save *n-1* I/O operations. It is clear that, if there are no further updates, there are no saves.
- The application issuing the write does not wait for the I/O operation.

The negative aspect of deferring writes is that VSAM does not have a log (at present). If the system fails before the buffer destage, you lose some updates in your data set. So it is up to you to make the decision based on the type of data you are processing.

The Defer Write option is only valid for LSR and GSR. The defer write option is bypassed in LSR and GSR if SHAREOPTIONS 4 is specified. Refer to 4.3, "Sharing VSAM data sets" on page 212.

When NSR defer write is used, with the exception of SHAREOPTIONS 4, the buffers are immediately refreshed.

You specify that writes are to be deferred by coding MACRF=DFR in the ACB, along with MACRF=LSR or GSR:

```
ACB      MACRF=({LSR|GSR},{DFR| NDF},...),...
```

In RLS mode, deferred writes are ignored and direct request modified buffers are immediately written to disk and the coupling facility.

Locating VSAM buffers above 16 MB

The *default* VSAM allocation for a resource pool is below 16 MB line. The exception is SMB, where the default is above 16 MB line. The location of the buffers and I/O control blocks can be controlled using:

- In an assembler application program:
  - For NSR buffering, in the RMODE31 keyword in ACB
  - For LSR/GSR buffering, in BLDVRP macro
- In the JCL, through RMODE31 parameter on the DD statement

The values you may specify for RMODE31 follow:

- ALL is used to allocate I/O relate control blocks and buffers above 16 MB. IDCAMS does not accept this value because it only accesses control blocks below the 16 MB line.
- BUFF is used only to allocate buffers above 16 MB.

- CB is used only for I/O relate control blocks above 16 M. IDCAMS does not accept this value because it only accesses control blocks below the 16 M line
- NONE is the default. VSAM allocates I/O related control blocks and buffers below 16 M.

With the flexibility of JCL, you can avoid the necessity of changing an existing application. But be aware that programs with AMODE=24 can abend with S0C4 when *locate* mode is used and RMODE31=BUFF is specified.

Locate mode is used when OPTCD=LOC is specified in the RPL, and indicates to VSAM to return to the application the address of the record, which is located in the buffer. Using locate mode, programs addressing 24 bits cannot access data above 16 MB (addressing 31 bits).

There is also a *move* mode, where VSAM moves your logical record from the buffer to a user area your program specifies.

COBOL for OS/390 always:

- Uses MVE mode; VSAM moves the record to an area whose address is indicated in the RPL, built by COBOL.
- Requires VSAM buffers above 16 MB.

You can relieve the storage constraint below 16 MB for application programs using VSAM data sets by specifying RMODE31 in the AMP parameter. You can increase the number of buffers with no effect on virtual storage below 16 MB. Do not forget to specify enough private area above. Refer to 2.6.12, “Region size” on page 61.

Recommendation
We recommend you use VSAM buffers above 16 MB.

System managed buffering (SMB)
SMB was introduced in DFSMS V1R4 and is available to all VSAM data sets open for NSR processing. SMB enables VSAM to determine the optimum number of index and data buffers, as well as the type of buffer management: LSR (LRU, lots of index buffers, lost of data buffers containing 20% of data, no look-ahead) or NSR (discard the ones already processed, just one index buffer, look-ahead).

SMB allocates, at Open time, the optimum number of data and index buffers based on the type of access declared in the ACB’s MACRF parameter. Usually SMB allocates much more buffers than without SMB. Then, the installation should allow a larger virtual storage, as making Region equal to zero. If the

Region is not enough, you may receive the following message: **IEC161I 001(8,36**).

Performance improvements can be dramatic with random access, particularly if you have little buffering, due to defaults, because SMB enlarges the number of buffers and switches from NSR mode to LSR mode as bufferpool managing is concerned. This can be seen in Table 2-7 on page 86.

When processing a VSAM data set, using SMB, and SMB switches to LSR mode, you can receive the message IEC161I 001(8,36)-087. It is issued due an error building the VSAM resource pool (BLDVRP macro) and indicates that there was not enough virtual storage to satisfy the request done by SMB. SMB gets the available storage and processing goes on. To get optimum SMB buffering, you should provide enough virtual storage. Refer to 2.6.12, "Region size" on page 61. You can relieve the use of storage below 16 MB by specifying that VSAM allocates buffers above16 MB. Refer to "Locating VSAM buffers above 16 MB" on page 81.

SMB is available under the following conditions:

- ▶ The data set must be in extended format. To be in extended format, the data set must be system managed (SMS) and use a data class defined with DSNTYPE=EXT. For details about extended format; refer to 1.9, "Extended format data set" on page 28.
- ▶ In the application program, ACB *must* be NSR and MACRF keyword cannot contain:
  - – ICI data list: Improved control interval processing.
  - – UBF: Management of I/O buffers left up to the VSAM user, as in DB2.
  - – For releases *prior* to z/OS 1.3 DFSMS, processing the data set through the alternate index of the path specified in the DDname is not supported.

When the conditions above are not satisfied, the job does not abend, but the SMB services are not used and no messages are issued.

You can invoke SMB services through one of the following methods:

- ▶ Using a data class defined with RECORD_ACCESS_BIAS
- ▶ Specifying ACCBIAS in the AMP parameter of JCL DD statement

The order of precedence for specifying values are shown in the *SOURCE* column in Table 2-3 on page 65.

Both parameters can be used to specify access bias with a subparameter ACCBIAS, where you specify the type of buffering management technique. You can specify ACCBIAS equal to one of the following values:

- **SYSTEM**: Let the system determine the buffering technique, based on:
  - The ACB's MACRF type of access intention: sequential, direct, or skip sequential, or a combination, and
  - The Storage Class specifications: read or write bias for sequential direct access and MSR.

  One of the four techniques is used (DO, DW, SO or SW). Refer to Table 2-8 on page 86. Those are explained in sequence.

  The MACRF type of access is just an *intention*. The real type of access is declared per I/O operation in the RPL If you know in advance that the MACRF is not correct (by talking to the programmer, for example). You must give a hand by telling it the real type of access.

  Also in MACRF all the accessing types maybe specified, not helping SMB in discovering the best buffer management method. This situation also happens in COBOL when *ACCESS MODE IS DYNAMIC* is used. It means all of them (direct, sequential and skip sequential) are declared in the MACRF. To help SMB there are the following parameters:

  - **DO**: SMB optimizes buffers management to direct access. SMB changes the buffering management to LSR and allocates the adequate number of buffers for direct processing.
  - **DW**: To indicate to SMB that the processing is mixed direct and sequential, but with dominance of direct access. So, more index buffers are allocated to support direct processing but some buffers will be reserved for data to help any sequential processing that might occur. The buffering management technique is changed to LSR.
  - **SO**: Indicates to SMB to optimize buffer allocations for sequential processing. More data buffers (less index buffers) are allocated to support sequential access.
  - **SW**: Specifies mixed processing, but with dominance of sequential access. SMB optimizes buffer handling for sequential processing allocating more data buffers, but buffers will be reserved for index to help direct access. SMB is faster than NSR for sequential process, as indicated in Table 2-9 on page 87.
- **USER**: Bypass SMB. This is the default if you code no specification for the ACCBIAS subparameter. This default is not used when the data class specifies RECORD_ACCESS_BIAS.

For direct optimization (DO), you can use the following DD AMP parameters (not present in the data class) to tell the LSR buffer manager how to handle the processing of the buffers:

- **SMBVSP**: Specifies the amount of virtual storage to obtain for buffers when opening the data set. Used to override the default buffer space to be obtained, which is calculated assuming that 20% of the data accounts for 80% of the accesses. The buffer space acquired is split across two LSR pools: one for the index and one for the data. The specification can be done in either of two formats:
  - SMBVSP=nnK
  - SMBVSP=nnM
- **SMBHWT**: Used to allocate hiperspace buffers based on a multiple of the number of address space virtual buffers that have been allocated. It can be an integer from 0 to 99. The value specified is not a direct multiple of the number of virtual buffers that are allocated to the resource pool, but act as a weighting factor for the number of hiperspace buffers to be established. The hiperspace size buffer will be a multiple of 4K. These buffers may be allocated for the base data component of the sphere. If the CI size of the data component is not a multiple of 4K, both virtual space and hiperspace is wasted. The default is 0 and means that hiperspace is not used.
- **SMBDFR**: Allows the user to specify whether writing the data from a buffer to DASD can be deferred until the buffer is required for another request or the data set is closed. By the way, a CLOSE macro invoked with TYPE=T (temporary, does not need an OPEN to restart processing) option does not write the buffers to DASD when LSR processing is used for direct optimization. The format is:

  SMBDFR=Y *or N*

  Y is the default for SHAREOPTIONS (1,3) and (2,3)

  N is the default for SHAREOPTIONS (3,3), (4,3) and (x,4)

Table 2-7 shows the benefits of using SMB, compared with the use of BLSR, refer to “Batch local shared resources (BLSR)” on page 92 and buffering default. When we ran our test, we had 61 CA splits. Some CPU time was due to managing these splits; allocating extents, and data set catalog entry update. For considerations on CA and CI splits, see 1.5.12, “Splits” on page 15.

If you specify RECORD_ACCESS_BIAS=SYSTEM in the data class, you should:

- Rely on the MACRF accessing options (in MACRF we trust)
- Make sure that MACFR only has one access type
- Ensure that storage class bias parameters is correctly set

You can also see that for extended format data sets, you can get performance enhancements, even with default buffering. In our lab tests (refer to Table 2-7), the number of EXCPs, CPU time, and connect time is less than non-extended format data sets. For details on why this happens, see “Use of extended format” on page 134.

Table 2-7 Direct access: benefits of using SMB: Updates and insertions

Ext format	Buffering	EXCPs	connect time (sec)	CPU time (sec)	Elapsed time (min)
No	Default	861744	123.51	51.37	843
Yes	Default	764332	120	50.32	809
No	BLSR	330852	73.13	24.42	245
Yes	SMB (System)	280276	62.5	19.22	189
Gain using SMB (%)		**67**	**50**	**62.5**	**78**

When SYSTEM is specified, the parameters used to determine how buffers are allocated and managed are:

- ACB’s MACRF values of SEQ, DIR, and SKP
- The storage class values for BIAS and MSR

Table 2-8 shows how ACB’s MACRF and storage class BIAS parameters affect the buffer allocation and management when SYSTEM is specified.

Table 2-8 Some effects of ACB’s MACRF and storage class BIAS parameters

ACB MACRF parameters	***Storage class BIAS***			
	SEQ	**DIR**	**Both**	**None**
MACRF=DIR	DW	DO	DO	DO
MACRF=SEQ (default)	SO	SW	SO	SO
MACRF=(SEQ,SKP)	SO	SW	SW	SW
MACRF=SKP	DW	DW	DW	DW
MACRF=(DIR,SEQ) or (DIR,SKP) or (DIR,SEQ,SKP)	SW	DW	DW	DW
Note: DO=Direct Optimized; DW=Direct Weighted; SO=Sequential Optimized; SW = Sequential Weighted				

Table 2-9 shows the results when loading our laboratory data set using default buffering and SMB. You can see, the results are the same using sequential optimization (SO) or SYSTEM. For details about our lab, refer to “General lab description” on page 396.

Table 2-9 Initial load mode comparing SMB with no-SMB buffering

Extended format	Buffering	EXCPs	Connect time	CPU time	Elapsed time
No	NSR - Default	27856	41.2	8.65	68
Yes	ACCBIAS=SO	12813	28.4	8.04	48
Yes	ACCBIAS=SYSTEM	12813	28.4	8.05	48
Gain using SMB (%)		**54**	**30**	**7**	**30**
Note: All times are shown in seconds					

Retry capability for Direct Optimized technique (DO)

In z/OS 1.3 DFSMS, retry capability was added to DO technique. SMB tries to:

1. Obtain storage buffers to about 20% of data set records plus to buffer the entire index component, if the VSAM organization is keyed.
2. Reduce the numbers of buffers by 50% from the optimum amount for the data components and retry.
3. Reduce the number of *data buffers* to the minimum and retry. The minimum size is 1 MB.
4. Reduce the number of *index buffers* to the minimum and retry. The minimum is the amount of virtual storage to contain the entire index set plus 20% of the sequence set records.

If none of the retries above is successful, change to Direct Weighted (DW).

Messages related with SMB

Messages related to SMB are listed below:

- IEC161I 001(sfi)-032:

  sfi is 101, 102 or BLDVRP return and reason code. For BLDVRP return codes, refer to “Return Codes from Macros Used to Share Resources Among Data Sets” in *z/OS DFSMS Macro Instructions for Data Sets,* SC26-7408.
- IEC161I 001(sfi)-033: means that SMB did not return ACCBIAS.
- IEC161I 001(sfi)-034: sfi is for CATALOG LOCATE ERROR.

SMB support for AIX

With z/OS DFSMS R1.3, VSAM data sets using alternate indexes and SMB have a better performance. Now SMB supports data sets with alternate indexes for the DO technique and better size the resources needed for processing with non-DO technique.

Using SMB, the buffer allocation and buffering technique used depends on:

- The order in which the components are opened:
  - Base Cluster
  - Path Cluster
  - The ACCBIAS informed by the user or detected by SMB
  - Where VSAM control blocks exist, in such a case, there are already buffers:
    - When connecting to the existing structure, just add more buffers to existing pool.
    - When connecting is not possible, build new control blocks and new buffers, using SHAREOPTIONS, to enforce the sharing rules.

During OPEN time, the decision to share a control block structure is based on DSN and DDN sharing criteria. For sharing criteria refer to “Using a single VSAM control block structure” on page 221.

Table 2-10 presents how SMB allocates buffers when a VSAM control block structure exists for the components, when sharing by data set name (MACRF=DSN).

Table 2-10 SMB: AIX support with Data Set Name Sharing (MACRF=DSN)

OPEN BASE CLUSTER:				
Current ACCBIAS	**User ACCBIAS** [a]	**COMPONENT**	**SHARE OPTIONS**	**BUFFERS**
DO	DO or blank	All	Does not matter	Connect in the existing structure and LSR pool
DO	Non DO	Base Cluster	Apply	See Note [b]
		Upgrade AIXes [c]		BUFNI=# levels BUFND=2
Non-DO	DO	All	Apply	See note b

OPEN BASE CLUSTER:				
Current ACCBIAS	**User ACCBIAS** [a]	**COMPONENT**	**SHARE OPTIONS**	**BUFFERS**
Non-DO	Non-DO or blank	Base Cluster	Does not matter	More BUFND and BUFNI added to current pool [d]
		UPGRADE AIXes [e]	Does not matter	BUFNI=# levels BUFND=2
OPEN PATH CLUSTER:				
DO	DO or blank	All	Does not matter	Connect in the existing structure and LSR pool
DO	Non DO	Base Cluster	Apply	See note b
		UPGRADE AIXes c	Apply	BUFNI=# levels BUFND=2
Non DO	DO	All	Apply	See note b
Non DO	Non-DO or blank	Path AIX	Does not matter	More BUFND and BUFNI added to current pool d
		UPGRADE AIXes	Does not matter	BUFNI=# levels BUFND=2

a. ACCBIAS specified by the user in the DDname being opened
b. A new VSAM control block structure is used and SMB allocates buffers according to user ACCBIAS or ACCBIAS detected by SMB.
c. When open intention for UPGRADE components is for OUTPUT
d. When SMB detects ACCBIAS=DO, SMB treats as DW.
e. For UPGRADE components, If this is the first OPEN for OUTPUT.

Table 2-11 presents how SMB allocates buffers when sharing by DDname, that is MACRF=DDN in ACB.

Table 2-11 SMB: AIX support with DDname Sharing (MACRF=DDN)

OPEN BASE CLUSTER:				
Current ACCBIAS	**User ACCBIAS** [a]	**COMPONEN;**	**SHAREOPTIONS**	**BUFFERS**
DO	DO or blank	All	Does not matter	Connect in the existing structure and LSR pool

OPEN BASE CLUSTER:				
Current ACCBIAS	**User ACCBIAS [a]**	**COMPONEN;**	**SHAREOPTIONS**	**BUFFERS**
DO	Non DO	Base Cluster	Apply	See Note [b]
		Upgrade AIXes [c]	Apply	BUFNI=# of levels BUFND=2
Non-DO	DO	All	Apply	See note b
Non-DO	Non-DO or blank	Base Cluster	Does not matter	More BUFND and BUFNI added to current pool [d]
		UPGRADE AIXes [e]	Does not matter	BUFNI=# of levels BUFND=2
OPEN PATH CLUSTER:				
DO	DO or blank	All	Does not matter	Connect in the existing structure and LSR pool
DO	Non DO	Base Cluster	Apply	See note B
		UPGRADE AIXes [c]	Apply	BUFNI=# of levels BUFND=2
Non-DO	DO	All	Apply	See Note [f]
Non-DO	Non-DO or blank	Path AIX	Does not matter	More BUFND and BUFNI added to current pool d
		UPGRADE c	Does not matter	BUFNI=# of levels BUFND=2

a. ACCBIAS informed by the user in the DDname being opened
b. A new VSAM control block structure is used and SMB allocates buffers according to user ACCBIAS or ACCBIAS detected by SMB.
c. When open intention for UPGRADE components is for OUTPUT
d. When SMB detects ACCBIAS=DO, SMB treats as DW.
e. For UPGRADE components, If this is the first OPEN for OUTPUT.
f. A new VSAM control block structure is used and SMB builds an LSR pool.

Table 2-12 shows how SMB allocates buffers when there is no control block structure. This means, at the first OPEN.

Table 2-12 SMB: AIX support for DO, no VSAM control block structures

OPEN	ACCBIAS[a]	COMPONENT	BUFFERS
Base Cluster	DO	All	LSR Pool
	Non DO	Base Cluster	SMB [b]
		Upgrade AIXes [c]	BUFNI=# of levels BUFND=2
Path Cluster	DO	All	LSR Pool
	Non DO	Base Cluster	SMB allocates buffers for ACCBIAS=DW
		Upgrade AIXes [d] c	BUFNI=# of levels BUFND=2
		Path AIX	According to ACCBIAS or processing intention

a. User informed or detected by SMB
b. SMB allocates buffers according to user ACCBIAS or detected by SMB.
c. When open intention is for output and is the first open
d. Add to the Path AIXes if it is defined as UPGRADE

Table 2-13 shows the amount of processing per compressed read or write operations that we found in our lab environment.

Table 2-13 SMB: AIX support for non DO

	OPEN	COMPONENT	BUFFERS
MACRF=DDN	Base Cluster	Base Cluster	according to ACCBIAS or processing intention
		Upgrade AIXes	BUFNI=# of levels BUFND=2
	Pathname	Base Cluster	SMB allocates buffers for ACCBIAS=DW
		Path AIX	according to ACCBIAS or processing intention
		Upgrade AIXes [a]	BUFNI=# of levels BUFND=2

	OPEN	COMPONENT	BUFFERS
MACRF=DDN	Pathname	Base Cluster	SMB allocates buffers for ACCBIAS=DW
		Path AIX	according to ACCBIAS or processing intention
		Upgrade AIXes[a]	BUFNI=# of levels BUFND=2
	Base Cluster	Base Cluster	according to ACCBIAS or processing intention
		Upgrade AIXes[a]	BUFNI=# of levels BUFND=2
		Path AIX	according to ACCBIAS or processing intention

a. When OPEN for output for the first time.

Batch local shared resources (BLSR)

BLSR is a subsystem that provides advantages in an application using VSAM NSR buffering techniques to switch to LSR without changing the application source code or link-editing the application again. Only a JCL change is required.

Your application performance will improve using BLSR when direct access is used and the same CI is referenced more than once in the processing. Using the BLSR subsystem with sequential access could degrade performance rather than improve it. For information about how LSR works, see “Local Shared Resources (LSR)” on page 77.

For mixed processing (some direct, some sequential), you may benefit from using BLSR. If the amount of data to be processed sequentially is not very large, you can compensate for the lack of read-ahead by using a large data CI size.

BLSR supports the VSAM data set types KSDS, ESDS, RRDS and VRRDS. Using BSLR, you can force VSAM buffers and control blocks to be located above 16 MB without having to use hiperspace. You can also use hiperspace, and you can restrict its use with RACF or an equivalent security software.

The BLSR subsystem has the following restrictions:

- The ACB cannot be above 16 MB. Otherwise, the system fails the OPEN request with error message IEC190I.

- If the application closes and then reopens the ACB without refreshing the DDNAME, the request is bypassed, and the data set is opened using the same options as the last time it was opened.

  That is, if the data set was previously opened for LSR processing, then it is reopened for LSR. Similarly, if the data set was not eligible for LSR processing the first time, then it is reopened for NSR processing, even if LSR is now applicable.

High-level languages refresh the DDNAME for each open. Consequently, the subsystem is always called for:

- COBOL/VS programs using the ISAM interface to access VSAM data sets.
- PL/1 programs written for ISAM accessing VSAM data sets.
- Other programs using the RDJFCB macro to try to identify the file type, for example, IDCAMS.

Before using BSLR, the subsystem must be installed. Contact your installation system programmer, or refer to the manual MVS Batch Local Shared Resources, GC28-1469.

When the subsystem is active, to invoke its services, add the SUBSYS parameter to the JCL. The following example illustrates how to do this.

If, for example, the application opens the following data set for NSR processing:

```
//VSAMDD  DD DISP=SHR,DSN=VSAMDSN
```

You can convert to the BLSR subsystem as follows:

1. Change the DDNAME on the previous JCL command statement, for example, from `VSAMDD` to `NEWBUFF`:

   ```
   //NEWBUFF DD DISP=SHR,DSN=VSAMDSN
   ```

2. Add the following DD statement, where the SUBSYS subsystem-name subparameter is BLSR and the SUBSYS DDNAME subparameter is the DD name selected in Step 1:

   ```
   //VSAMDD  DD SUBSYS=(BLSR,'DDNAME=NEWBUFF')
   ```

When SUBSYS is specified in JCL, the job's INITIATOR issues the IEFSSREQ macro, invoking BLSR subsystem services, passing the information specified in the SUBSYS parameter.

Then, the BSLR subsystem:

1. Includes an EXIT to call BLSR at OPEN.
2. Dynamically allocates the data set to the correct DDNAME (`NEWBUFF` in the example).

When the application opens the `VSAMDD` ACB, the BLSR subsystem completes the conversion to LSR processing.

The following system parameters are not allowed with the SUBSYS parameter:

```
*, AMP, BURST, CHARS, COPIES, DATA, DDNAME, DYNAM, FLASH, MODIFY, QNAME, SPACE,
SYSOUT.
```

You must nullify any of these parameters if they are specified on a DD statement you are overriding.

You specify the SUBSYS parameter as:

```
SUBSYS=(subsys-name,'DDNAME=value','subparm1=value',....,'subparmn=value')
```

Where:

- *subsys-name* is the name given to BLSR subsystem, usually, BLSR.
- *DDNAME=value* is the name of the DDNAME to be open by the application program.

The following subparameters are allowed on the BLSR SUBSYS parameter: BUFND, BUFNI, HBUFND, HBUFNI, RMODE31, STRNO, DEFERW, SHRPOOL, BUFSD, BUFSI and MSG.

BLSR can be used with SMS and non-SMS-managed data sets.

If your data set is SMS-managed and is in extended format, you will get better performance using SMB. For details, see “System managed buffering (SMB)” on page 82.

We recommend:

- If possible, use SMB. The results are better, as shown in Table 2-7 on page 86.
- If possible, locate the buffers above 16 MB.

2.6.15 Data compression

To compress means to store data in a format that requires less space than the original data. There are quite a few methods (algorithms) to compress data, such as:

- Character-based methods: Huffman, Run-length encoding, Ziv-Lempel
- Bit-level methods: Image data compression, IDRC (tape controllers)

Some of these methods use the concept of a dictionary, which is a mapping from one vocabulary to another. There are two types:

- Compression dictionary
- Expansion dictionary

The same method may use many dictionaries.

S/390 uses the Ziv-Lempel (ZL) method in software and hardware options. ZL-based schemes work by entering phrases into a dictionary and then, when a repeat occurrence of that particular phrase is found, outputting the dictionary index instead of the phrase. Several compression algorithms are based on this principle. They differ mainly in the manner in which they manage the dictionary, and all of them tend to perform much better in decoding than in encoding. S/390 compression uses the ZL1 version of the ZL implementation, and the RVA family of products uses ZL2.

Where to use compression

Compression can be executed in the processor or outbound in an I/O controller (for example, tape). With CPU compression, you save:

- I/O buffer pool space
- Channel cycles
- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- Transmission line cycles (for TP)

The advantages of compressing in the controller are to save:

- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- CPU cycles

z/OS compression interface

The interface to data compression in z/OS is through the Compression Management Facility (CMF), which consists of two parts.

Compression service activation

Compression service activation covers checking and setting up the required compression environment.

It determines if the Compression Call instruction is available; if not, the compression is done by software.It verifies if the SYS1.DBBLIB data set is available. It contains dictionary building blocks (DBB) used for compression.

Compression management services (CMS)

Compression management services covers the actual process a data set goes through when compression is requested. CMS provides and carries out the following services used by VSAM.

VSAM Candidate data set verification

VSAM Candidate data set verification has the following requirements:

- KSDS organization only. Only data is compressed, indexes are not compressed (AIXs are not compressed).
- SMS functions must be active and the data set must be SMS managed and in extended format.
- Data class assigned has to specify the DSNTYPE=EXT with a required COMPACTION=Y (blank defaults to N).
- Have a primary allocation of at least 5 MB (data component only) due to the amount of sampling needed to develop a dictionary token. If no secondary allocation is specified, then the primary allocation must be at least 8 MB.
- Have a minimum record length of 40 bytes (not including key length).
- CI Mode processing is not allowed.
- BCS catalog, system data sets, and temporary data sets cannot be compressed because extended format is not supported.

For VSAM extended format and compression restrictions and incompatibilities see *DFSMS Using Data Sets*, SC26-4922-01.

Dictionary selection

There are two forms of compression: generic and tailored (refer to Figure 2-7 on page 98.)

- **Generic compression**:

  This involves the sampling and interrogation of the compression eligible data set by CMF. Next, a sample from the data set is taken, at load time, and compared to the DBBs for similar content and compression efficiency. Up to 64 KB of the data set can be sampled and written before the data set starts to be compressed. The first time a compressed format data set is written to disk, the first bytes are written in non-compressed form.

  Because the dictionary is assembled using DBBs during the sampling and interrogation, the dictionary does not exist when the first bytes of the data set are written. Once the dictionary is built, the data can use this dictionary to compress the rest of the data set. Information about the mix of DBBs selected during the sampling and interrogation process is kept in the catalog. This information enables the decompression of the data set when required but does not require a dictionary to be stored with the data set.

- **Tailored compression**:

  Tailored compression introduces a new form of compression for sequential extended format data sets. By the way, VSAM data sets do not support Tailored compression.

  With tailored compression, the system attempts to derive a compression dictionary that is tailored specifically to the initial data written to a data set. Once a tailored dictionary is derived, it is imbedded in the compressed data set. This technique is expected to provide improved compression ratios, thereby reducing DASD usage and channel traffic.

  Because the dictionary is tailored to the user data, significantly more data is sampled than was required by generic compression. The process of sampling the data and building the dictionary during creation of a new data set takes more CPU cycles and is, therefore, most noticeable when compressing small data sets. For larger data sets, the cost of sampling is amortized significantly. Reuse of a tailored compressed data set saves the cost of sampling and dictionary creation.

  An installation has the option of either using the new tailored compression or continuing to use generic DBB-based compression first introduced with DFSMS V1R2. Tailored compression allows for more types of data to be compressed, for example, where non-English languages are used.

  Because the original generic dictionary was developed with standard American-English scripts, files containing sequences of characters that do not appear in the American scripts do not compress very well. The tailored dictionary avoids this problem, as it is generated dynamically from the data itself, for each data set.

  Dynamically generating a tailored dictionary from the data itself should make tailored compression more useful to the non-English-speaking world. Tailored compression support does not apply to VSAM KSDSs, which can continue to be compressed with generic DBB dictionary compression.

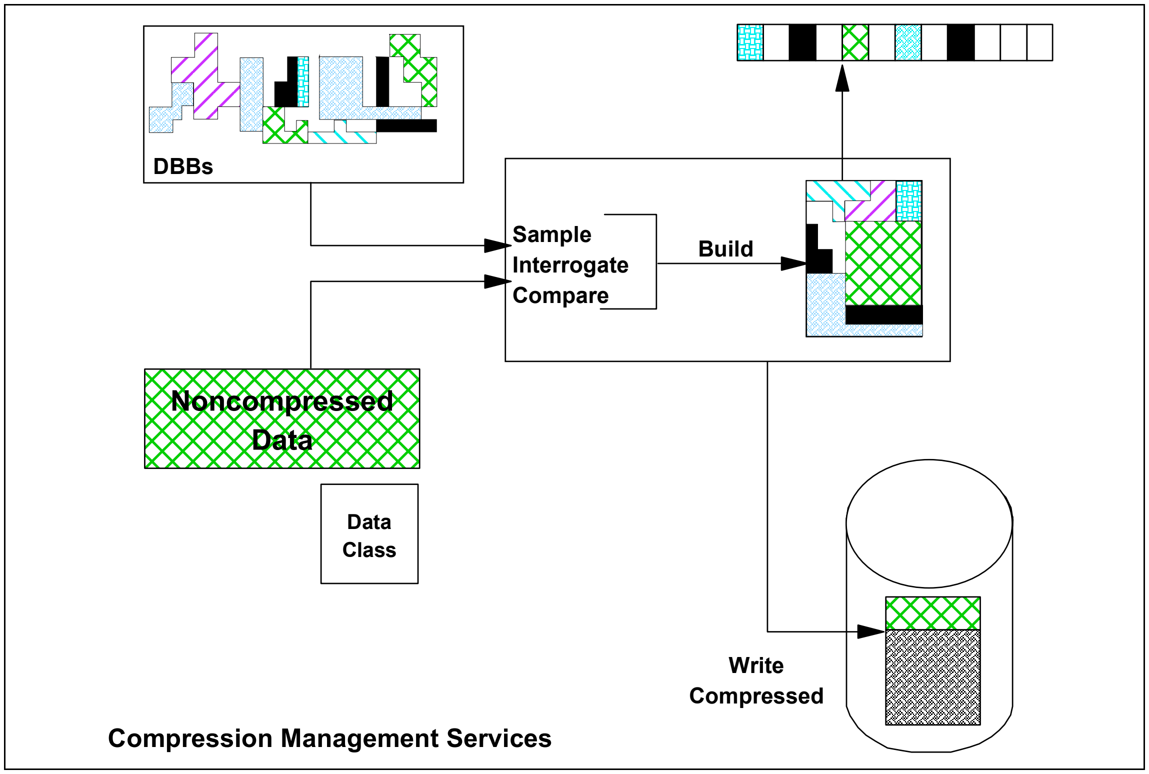


Figure 2-7 CMS dictionary selection

Forms of candidate data sets

A compression-eligible data set can exist in one of three forms, depending on the moment:

- *Dictionary selection:* The data set is eligible, but its makeup is yet to be determined. The dictionary tokens are compared with the data set contents during interrogation and sampling. Interrogation maps bytes into alpha, numeric, upper- or lower-case, and sampling evaluates DBB compression efficiency.
- *Mated*: The data set is mated with the appropriate dictionary; this concludes the sampling and interrogation processes.
- *Rejected*: A suitable dictionary match was not found during sampling and interrogation, so compression is bypassed, or for a sequential data set, the data set was closed before a token could be selected. Note that the data set is in compressed format.

There are some types of data sets that are not suitable for ZL compression, resulting in rejection, such as these:

- Small data sets.
- Data sets in which the pattern of the data does not produce a good compression rate, such as image data.
- Small records. Compression is performed on a logical record basis. When logical records are very short, the cost of additional work may become excessive compared to the reduction in size that compression achieves.

Data compression and decompression

Data compression and decompression are invoked whenever read and/or write or get and/or put is done on a mated data set. A mated data set is one that has acquired a suitable dictionary token through successful interrogation and sampling processes. In other words, suitable building blocks have been found and selected from the DBB distribution library, and their combination constitutes the dictionary token associated with the data set and held in the catalog entry as well. Compression and decompression of a mated data set use the dictionary built from the dictionary token associated with the data set entry in the extended format cell of the catalog.

The dictionary is customized to the data set through the dictionary selection process. However, the dictionary is not stored with the data. Only the token information is stored, because the dictionary is a table dynamically built in storage from the token information that enables the compression and decompression of a data set when it is opened.

Using this approach, DFSMS retains responsibility for dictionary management and shields the user and application from the physical representation of compressed data on disk.

Compression concepts

Some basic concepts regarding compression include:

- Import into an empty data set does not propagate extended format and compression information.
- When data is compressed, the length of a stored record may change after an update without any logical record length change.
- Locate mode processing is allowed but requires a larger number of internal work areas to process.
- ISMF adds a *%user data reduction* (compression factor) field in data set application.

- IDCAMS REPRO copies data sets without decompressing:
  - If the target data set is eligible for compression, and
  - If the target device is a like device, or CI sizes are equal for VSAM
- Otherwise, REPRO decompress the data when reading and optionally compresses the data when writing.
- Our SMF sample report shows the compression factor.
- LISTCAT lists the compression related information.
- IEHLIST indicates that the data set is compressed.
- DFSMSdss™ can be used for:
  - *Logical dump*: This does not change format from compressed to non-compressed or vice versa; and includes cataloging information with dump.
  - *Logical restore*: This does not change format from non-compressed to compressed.
  - *Logical copy*: This never changes the format from non-compressed to compressed.
  - DFSMShsm avoids double compression.

VSAM compression

VSAM compression is done transparently to the application, through the data class (DC) parameter in SMS data sets. This DC assigned to the data set has to specify the following DSNTYPE=EXT with a required COMPACTION=Y (blank defaults to N). Figure 2-8 shows the ISMF list of the DC:

```
DATACLAS                     EXTENDED                   MEDIA
NAME     DATA SET NAME TYPE  ADDRESSABILITY COMPACTION TYPE
--(2)--- -------(26)-------  -----(27)-----  ---(28)---  -(29)-
DCSMB    EXTENDED REQUIRED   NO              ----        ------
DCSTRIPE EXTENDED REQUIRED   NO              ----        ------
DCXXXX   EXTENDED REQUIRED   NO              ----        ------
DIRECT   ------------------  NO              ----        ------
ECCST    ------------------  NO              YES         MEDIA2
EHPCT    ------------------  NO              YES         MEDIA3
```

Figure 2-8 ISMF Data Class display

VSAM compression only applies to KSDS in extended format. All the fields to the left of the key in the logical record are not compressed. In your next data model, you can define the key field with offset equal to zero.

Compression affects the catalog, VTOC, and SMF information about VSAM data sets. Refer to “Compression information sources” on page 101 for information on how to get this data.

Compression performance

The relative CPU compression cost depends on:

- Dictionary size:

  Usually, you do not have much control over this.

- Ratio of reads to writes of compressed records:

  Ziv-Lempel expands faster (less CPU cycles) than compresses. This means that it is better suited for data sets with a high read-to-write ratio.

Table 2-14 shows the amount of processing per compressed read or write operation that we found in our lab environment.

Table 2-14 Comparing compression

Type of access [a]	Compression	EXCPs	Device connect time (sec)	Step elapsed time (sec)
Initial load	No	12813	28.38	48
Initial load	Yes	10243	23.09	57
Direct access (updates/inserts)	No	280276	62.5	189
Direct access (updates/inserts)	Yes	288461	74.23	112
Direct access (reads)	Yes	99347	15.90	76
Direct access (reads)	No	102697	16.43	77

a. SMB was used in all experiments

We recommend that you:

- Do not use compression for data sets with low read-to-write ratio (less than 60%).
- Set key offset equal to zero.

Compression information sources

Compressed data sets have specific information about the compression process in different sources.

- Catalogs:

  ICF catalogs are not eligible for compression, but are enhanced to contain additional information about compressed format data sets in the extended format cell of the catalog entry. The extended format cell is part of the VSAM volume data set (VVDS) component of the ICF catalog.

  The extended format cell holds:

  – Number of stripes (STRIPE-COUNT)
  – Compression flags (COMP-FORMAT)
  – Physical block size
  – Non-compressed user data set size in bytes (USER-DATA-SIZE)
  – Compressed user data set size in bytes (COMP-USER-DATA-SIZE)
  – Active dictionary token (ACT-DIC-TOKEN — Active Dictionary Token or NULL)
  – Whether user data sizes are valid (SIZES-VALID)
  – Compression characteristic record.

In the appendix, there is a REXX routine to search in a catalog for all compressed data sets or an specific one. It shows for each of them the percent compression ratio. This tool allows you to determine if your CPU's cycles are efficiently utilized, in terms of saving storage media, channel cycles, and buffer pools. Refer to "SMFLSR sample program" on page 380.

- SMF:

  New fields have also been added to SMF type 64 records for VSAM compression. These new fields record the size of the data before and after compression, flags for extended format and compression, and dictionary tokens used for compression. You can find more detailed information on the SMF record layout in *MVS/ESA SP V5 System Management Facilities (SMF)*, GC28-1457.

  QSAM compression information in the type 14 and 15 records is updated when CLOSE is performed on a sequential compressed format data set. Information on the dictionary token selected and the compressed and non-compressed data set size is added to the SMF type 14 and 15 records.

  This information is also maintained in the extended format cell in the catalog.

- VTOC:

  There is now a compression indicator, DS1COMPR, and an extended format data set indicator, DS1STRP, in the VTOC. Because a compressed format data set must be an extended format data set, both indicators are on for compressed format data sets:

  VTOC Format 1 DSCB

DS1STRP — extended format VSAM data set. Contained within the DS1SMSFG offset x’ 4 E’

DS1COMPR — Compressed data set. Contained within the DS1FLAG1 offset x’ 3 D’

2.6.16 VSAM Data striping

Usually, in a multi-extent, multi-volume VSAM data set processed in sequential access mode, processing does not present any type of parallelism for I/O operations among the volumes. This means that when an I/O operation is executed for an extent in a volume, no other I/O activity from the same task/same data set is scheduled to the other volumes. In a situation where I/O is the major bottleneck, and there are available resources in the channel subsystem and controllers, it is a waste of these resources.

Data striping addresses this performance problem by imposing two modifications to the traditional data organization:

- The records are not placed in *key ranges* along the volumes; instead they are organized in stripes.
- Parallel I/O operations are scheduled to sequential stripes in different volumes.

By striping data, the tracks in the case of SAM, and the control intervals (CIs) for VSAM, are spread across multiple devices. This format allows a single application request for records in multiple tracks and CIs to be satisfied by concurrent I/O requests to multiple volumes.

The result is improved performance by achieving data transfer into the application at a rate greater than any single I/O path. The scheduling of I/O to multiple devices in order to satisfy a single application request is referred to as an *I/O packet*.

VSAM data striping topics

Data striping support for VSAM was initially provided with DFSMS 2.10. Support is provided for all VSAM organization, including KSDS, ESDS, RRDS, VRRDS, and LDS.

The idea is to spread sequenced data CIs in different data CAs located in different volumes. Following are the characteristics of a VSAM striped data set:

- Striping is done by CI, as opposed to a track for SAM.
- A stripe is associated with a single volume or a set of volumes (if you have multi-layer).

- Only the data component of a base cluster may be striped.
- A data set may have up to a maximum of 16 stripes.
- A stripe on a single volume has a maximum of 123 extents per stripe.
- A multi-layered stripe (a stripe allocated on several volumes) has a maximum of 255 extents per stripe.
- The following features are supported in a striped VSAM data set:
  - Extended addressability (>4GB):
  - Compressed format
  - Partial release
- A data set is system-managed.
- A data set is allocated in extended format.
- Prior to DFSMS z/OS 1.3 a striped data set cannot be reused.
- A data set may be assigned either GUARANTEED SPACE (does not support multi-layered) or NON-GUARANTEED SPACE.
- For *guaranteed space*, the number of stripes is equal to the number of volumes (VOLUME COUNT) you specify in the data class or the VOLUMES parameter in JCL, up to a maximum of 16 stripes. The JCL specification overrides the data class.

  For *non-guaranteed space*, SMS determines the number of stripes to use based on the value of the SUSTAINED DATA RATE(SDR) in the storage class.

Figure 2-9 shows an example of a four-stripe VSAM data set.

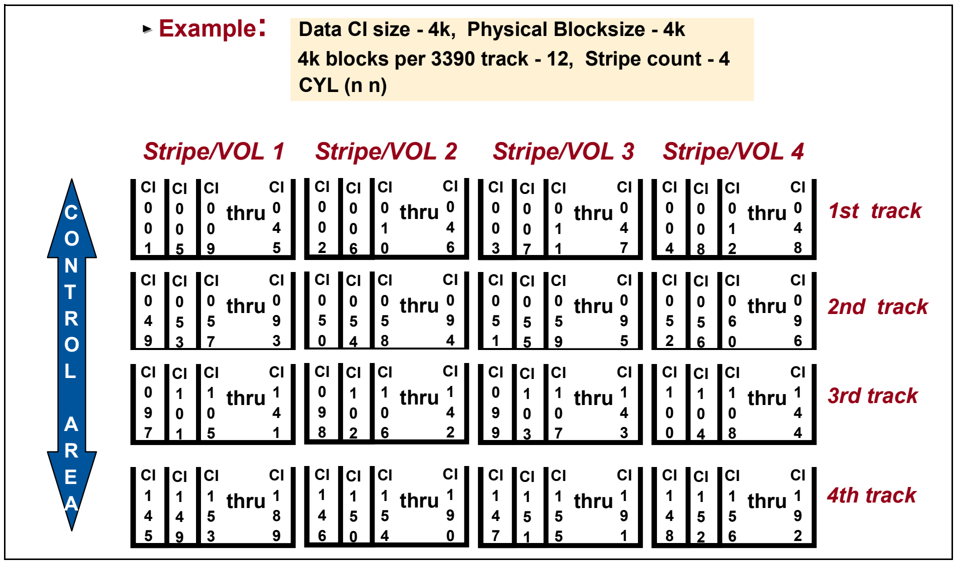

Example:
Data CI size - 4k, Physical Blocksize - 4k
4k blocks per 3390 track - 12, Stripe count - 4
CYL (n n)
Stripe/VOL 1
Stripe/VOL 2
Stripe/VOL 3
Stripe/VOL 4
CONTROL AREA
1st track
2nd track
3rd track
4th track
CI 001, CI 005, CI 009 thru CI 045
CI 002, CI 006, CI 010 thru CI 046
CI 003, CI 007, CI 011 thru CI 047
CI 004, CI 008, CI 012 thru CI 048
CI 049, CI 053, CI 057 thru CI 093
CI 050, CI 054, CI 058 thru CI 094
CI 051, CI 055, CI 059 thru CI 095
CI 052, CI 056, CI 060 thru CI 096
CI 097, CI 101, CI 105 thru CI 141
CI 098, CI 102, CI 106 thru CI 142
CI 099, CI 103, CI 107 thru CI 143
CI 100, CI 104, CI 108 thru CI 144
CI 145, CI 149, CI 153 thru CI 189
CI 146, CI 150, CI 154 thru CI 190
CI 147, CI 151, CI 155 thru CI 191
CI 148, CI 152, CI 156 thru CI 192

Figure 2-9 Striped VSAM data set

As you can see, a data control area is formed by CIs not within sequenced key range. Also for a KSDS, index records in the sequence set contained in the same CI index point to different CAs.

Layering and striped data

VSAM supports multi-layering. A layer in a striped environment is defined as the relationship of the volumes that make up the total number of stripes. That is, those volumes that participate as part of an I/O packet.

Once the stripe extends to a new volume, and the I/O packet changes, this constitutes another layer. The Sequential Access Method (SAM) is restricted to extending only to the current stripe volume and does not support the concept of multi-layering.

Figure 2-10 shows an example of the concept of layering with a three-stripe data set.

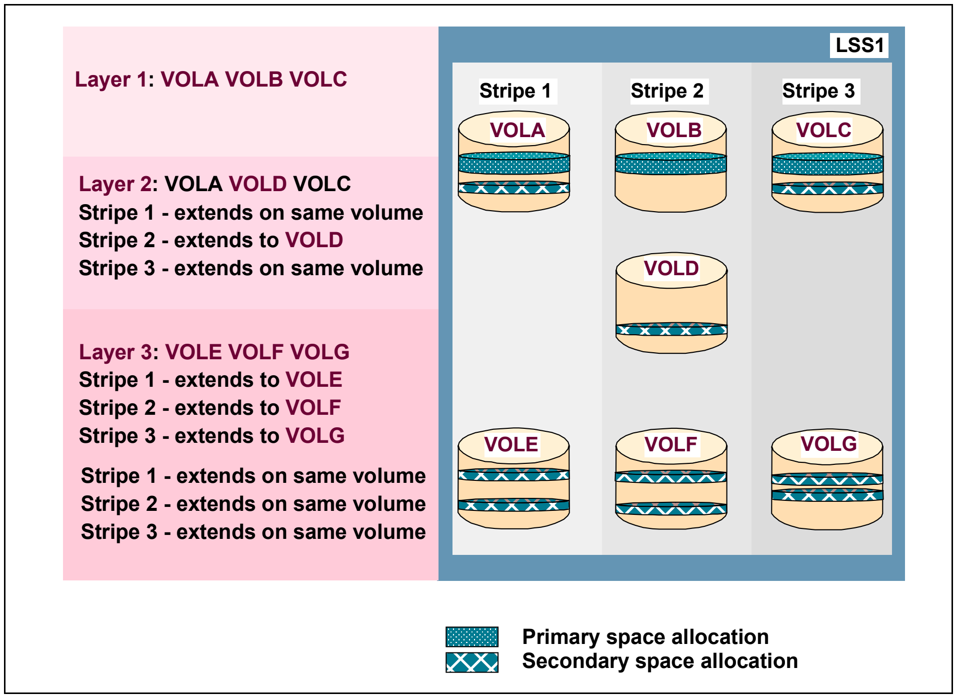

Layer 1: VOLA VOLB VOLC
Layer 2: VOLA VOLD VOLC
Stripe 1 - extends on same volume
Stripe 2 - extends to VOLD
Stripe 3 - extends on same volume
Layer 3: VOLE VOLF VOLG
Stripe 1 - extends to VOLE
Stripe 2 - extends to VOLF
Stripe 3 - extends to VOLG
Stripe 1 - extends on same volume
Stripe 2 - extends on same volume
Stripe 3 - extends on same volume
LSS1
Stripe 1
Stripe 2
Stripe 3
VOLA
VOLB
VOLC
VOLD
VOLE
VOLF
VOLG
Primary space allocation
Secondary space allocation

Figure 2-10 Layering in VSAM data set striping

Implementing VSAM data striping

To create a striped VSAM data set, you must do the following:

- Define an SMS-managed VSAM data set in extended format.

  You should specify *Data Set Name Type* in the data class. This forces the allocation in a DASD controller supporting EF. *Data Set Name Type* may be set to EXT Preferred (SMS tries to allocate in a controller supporting EF). The data set is not defined as striped if it is not allocated in extended format.

- Specify the Sustained Data rate (SDR) parameter in the storage class for either guaranteed or non-guaranteed space. This tells SMS that you want to implement striping.

- For *guaranteed space*, specify the VOLUME COUNT in the data class or the VOLUMES parameter in JCL. The JCL specification overrides the data class.

  VOLUME COUNT represents the number of volume cells associated with data set in catalog entries. If additional volumes are required, the data set access must be closed, the IDCAMS ALTER ADDVOLUMES command must be issued. With z/OS DFSMS 1.3 the MAXIMUM VOLUME COUNT data class parameter is introduced. It represents actual maximum number of volumes an SMS-managed VSAM cluster can span. Now, volumes can be added dynamically to the cluster without manual intervention and without taking the application down.

If you do not specify either the volume count or the volume serial numbers, you only get a single stripe.

- For *non-guaranteed space*, just specify the SDR value in the storage class. SMS computes the number of stripes based on the rule that each volume delivers 4 MB/second rate. The volume count is ignored. Then, if you specify 16 MB/sec, VSAM generates a 4-stripes data set.

  As a simple example: if you specified a SDR of 17MB/sec, your data set will be defined with 4 stripes.

Examples of defining striped data sets

In this example, we create a striped VSAM data set using guaranteed space, and a specified volume count of 4. The data class name is KEYEDEXG, and the storage class is STRIPE.

```
//STRIPED  JOB 'DEF STRIPED-EF VSAM DS',MSGCLASS=X,NOTIFY=&SYSUID
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           DELETE DAWN.KSDSEXG
           DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXG)  -
              DATACLASS(KEYEDEXG) -
              STORAGECLASS(STRIPE))
           LISTC ALL ENTRIES(DAWN.KSDSEXG)
/*
```

Figure 2-11   Sample JCL to create a striped data set

The content of the KEYEDEXG data class is:

```
CDS Name  . . . : SYS1.SMS.SCDS
Data Class Name : KEYEDEXG

Description : USING GUARANTEED SPACE

Recorg . . . . . . . . . : KS
Recfm  . . . . . . . . . .:
Lrecl. . . . . . . . . . : 300
Keylen . . . . . . . . . : 8
Keyoff . . . . . . . . . : 0
Space Avgrec . . . . . . : K
      Avg Value  . . . . : 300
      Primary  . . . . . : 600
      Secondary  . . . . : 100
      Directory  . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . . : 4
  Add'l Volume Amount  . :
Imbed  . . . . . . . . . :
Replicate  . . . . . . . :
CIsize Data  . . . . . . : 4096
% Freespace CI . . . . . : 10
            CA . . . . . : 10
Shareoptions Xregion . . :
             Xsystem . . :
Compaction . . . . . . . :
Media Interchange
  Media Type . . . . . :
  Recording Technology :
Data Set Name Type  . . . : EXTENDED
  If Extended . . . . . . : REQUIRED
  Extended Addressability : NO
  Record Access Bias  . . : USER
Reuse . . . . . . . . . . : NO
Initial Load  . . . . . . : RECOVERY
Spanned / Nonspanned  . . :
BWO . . . . . . . . . . . :
Log . . . . . . . . . . . :
Logstream Id  . . . . . . :
Space Constraint Relief . : NO
  Reduce Space Up To (%)  :
```

Figure 2-12 Sample content of KEYEDEXG data class

The content of the STRIPE storage class is:

```
CDS Name  . . . . . . : SYS1.SMS.SCDS
Storage Class Name  : STRIPE
Description  : TO BE USED FOR STRIPING TEST

Performance Objectives
  Direct Millisecond Response  . . . :
  Direct Bias  . . . . . . . . . . . :
  Sequential Millisecond Response  . :
  Sequential Bias  . . . . . . . . . :
  Initial Access Response Seconds  . :
  Sustained Data Rate (MB/sec) . . . : 17
Availability . . . . . . . . . . . . : NOPREF
Accessibility  . . . . . . . . . . . : NOPREF
  Backup . . . . . . . . . . . . . . :
  Versioning . . . . . . . . . . . . :
Guaranteed Space . . . . . . . . . : YES
Guaranteed Synchronous Write . . : NO
Cache Set Name . . . . . . . . . :
CF Direct Weight . . . . . . . . :
CF Sequential Weight . . . . . . :
```

Figure 2-13 Sample content of STRIPE storage class

This is the job output. The data set is explicitly defined with four stripes with the specified volume count of 4:

```
DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXG)  -
              DATACLASS(KEYEDEXG) -
              STORAGECLASS(STRIPE))
IGD17070I DATA SET DAWN.KSDSEXG ALLOCATED
SUCCESSFULLY WITH 4 STRIPE(S).
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME MHLV14 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME MHLV14 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0512I NAME GENERATED-(D) DAWN.KSDSEXG.DATA
IDC0512I NAME GENERATED-(I) DAWN.KSDSEXG.INDEX
IDC0181I STORAGECLASS USED IS STRIPE
IDC0181I MANAGEMENTCLASS USED IS MCDB22
IDC0181I DATACLASS USED IS KEYEDEXG
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 2-14 Sample IDCAMS control statements and output for four stripes

In this example, we create a striped VSAM data set using non-guaranteed space and SDR 17 MB/sec:

```
//STRIPED  JOB 'DEF STRIPED-EF VSAM DS',MSGCLASS=X,NOTIFY=&SYSUID
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           DELETE DAWN.KSDSEXT
           DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXT)  -
              DATACLASS(KEYEDEXT) -
              STORAGECLASS(STRIPE))
           LISTC ALL ENTRIES(DAWN.KSDSEXT)
/*
```

The content of the KEYEDEXT data class is:

```
CDS Name  . . . : SYS1.SMS.SCDS
Data Class Name : KEYEDEXT

Description : TO BE USED FOR STRIPING TEST

Recorg . . . . . . . . . : KS
Recfm  . . . . . . . . . :
Lrecl  . . . . . . . . . : 300
Keylen . . . . . . . . . : 8
Keyoff . . . . . . . . . : 0
Space Avgrec . . . . . . : K
      Avg Value  . . . . : 300
      Primary  . . . . . : 600
      Secondary  . . . . : 100
      Directory  . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . . : 8
  Add'l Volume Amount  . :
Imbed  . . . . . . . . . :
Replicate  . . . . . . . :
CIsize Data  . . . . . . : 4096
% Freespace CI . . . . . : 10
            CA . . . . . : 10
Shareoptions Xregion . . :
             Xsystem . . :
Compaction . . . . . . . :
Media Interchange
  Media Type . . . . . :
  Recording Technology :
Data Set Name Type  . . . : EXTENDED
  If Extended . . . . . . : REQUIRED
  Extended Addressability : NO
  Record Access Bias  . . : USER
Reuse . . . . . . . . . . : NO
Initial Load  . . . . . . : RECOVERY
Spanned / Nonspanned  . . :
BWO . . . . . . . . . . . :
Log . . . . . . . . . . . :
Logstream Id  . . . . . . :
Space Constraint Relief . : NO
  Reduce Space Up To (%)  :
```

Figure 2-15   Example of KEYEDEXT data class

The content of the STRIPE storage class using non-guaranteed space is shown here:

```
CDS Name  . . . . . : SYS1.SMS.SCDS
Storage Class Name  : STRIPE
Description  : TO BE USED FOR STRIPING TEST
Performance Objectives
  Direct Millisecond Response  . . . :
  Direct Bias  . . . . . . . . . . . :
  Sequential Millisecond Response  . :
  Sequential Bias  . . . . . . . . . :
  Initial Access Response Seconds  . :
  Sustained Data Rate (MB/sec) . . . : 17
Availability . . . . . . . . . . . . : NOPREF
Accessibility  . . . . . . . . . . . : NOPREF
  Backup . . . . . . . . . . . . . . :
  Versioning . . . . . . . . . . . . :
Guaranteed Space . . . . . . . . . : NO
Guaranteed Synchronous Write . . : NO
Cache Set Name . . . . . . . . . :
CF Direct Weight . . . . . . . . :
CF Sequential Weight . . . . . . :
```

Figure 2-16   Example of content of STRIPE storage class

This is the job output of our second example. The data set is defined with five stripes. SMS is allocated on 5 volumes out of the 8 volumes defined in the storage group. Only the data component is striped:

```
DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXT)  -
              DATACLASS(KEYEDEXT) -
              STORAGECLASS(STRIPE))
IGD17070I DATA SET DAWN.KSDSEXT ALLOCATED
SUCCESSFULLY WITH 5 STRIPE(S).
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX29 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME MHLV13 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDCAMS  SYSTEM SERVICES
IDC0512I NAME GENERATED-(D) DAWN.KSDSEXT.DATA
IDC0512I NAME GENERATED-(I) DAWN.KSDSEXT.INDEX
IDC0181I STORAGECLASS USED IS STRIPE
IDC0181I MANAGEMENTCLASS USED IS MCDB22
IDC0181I DATACLASS USED IS KEYEDEXT
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 2-17 Second example with five stripes

After loading the data set, the LISTCAT output shows the HURBA only on the first volume. For the other volumes, HURBA=0.

CA size considerations

CA size calculations are affected by striping. Normally, the CA size is the lesser of the primary or secondary value and must not exceed 15 tracks if allocation is in tracks; or a cylinder, if allocation is in cylinders.

In striped data sets, the CA size amount must be a multiple of the number of stripes. That is, a CA cannot end in the middle of a stripe.

To meet this restriction, the CA size may have to be rounded to the next integral of the stripe count. Also, since the maximum stripe count is 16, a CA size of 16 tracks must be allowed to accommodate 16 stripes. Note that CA size maximum is increased to 16 tracks from previous 1 cylinder (15 tracks) allocation.

For striped data sets, all computations for CA size are performed using the equivalent amount of tracks, for example:

- A data set has seven stripes.
- The equivalent allocation in tracks is (45 30), so 15 tracks are used.
- 15 tracks rounded to the next integral of stripe count is 21. But 21 is greater than maximum of 16 tracks, so CA size is rounded down to 14 tracks.

Performance considerations

Consider the following when using VSAM striped data sets:

- The effective throughput gains on sequential access increases almost in proportion to the number of stripes, unless there is contention at the device, subsystem, path or channel level
- The throughput gains decreases from the expected, if you work with many stripes, or the Law of Diminishing Returns starts to work due to the raising of management. For KSDS, it has been experienced that after the fourth stripe, the processing improvement curve flattens. You may not get performance improvements if you define more than 4 stripes.
- The I/O response time for a striped data set is as long as the longest I/O in the stripe or the effective throughput is limited to the slowest stripe.
- The I/O post to the application is done when all the I/O to the stripes ends. Likewise, if an I/O error occurs in one of the stripes/volumes, the I/O operation ends with an error.
- For striped data sets, you should use SMB to determine the number of buffers, or you should allocate a larger value for the BUFND, depending on your application.

  If you are not using SMB, specify a BUFND value that is at least equal to the number of stripes. SMS needs at least one buffer for each stripe/volume accessed by the I/O request. Refer to 2.6.13, “Buffering options” on page 63 for more information. Using the default BUFND eliminates some of the benefits of striping.
- Using LSR for a VSAM striped data set is not rejected. However, you may not see a performance improvement in the way that NSR does.
- As we have mentioned, SMS computes the number of stripes based in your SDR parameter divided by a 4 MB/second rate. Because your DASD volume performs much better than this figure, you may get an actual better rate than what you have specified in the SDR in the storage class.
- Striping does not give a benefit for data sets accessed directly. However, these data sets are also accessed sequentially during backup, report generating, and so on. Therefore, you may want to consider striping all your VSAM data sets
- VSAM striping is not supported in RLS mode.

Recommendations

We make the following recommendations:

- For striped data sets, use SMB to determine the number of buffers or allocate a larger value for the BUFND, depending on your application. Using the default BUFND eliminates some of the benefits of striping.

- All the volumes that contain the stripes should have the same speed.
- As a rule of thumb for VSAM, do not go beyond 4 stripes — if you do, you will overload the processor.
- Define striping only for sequential processing.

2.7 VSAM performance by scenarios

Here, we simulate a scenario, where one or more VSAM data sets are causing performance problems to key transactions in a real installation. We cover all the aspects that may affect the VSAM I/O delay time, (Tw(IO)); and the I/O service time, (Ts(IO)). Also, because VSAM CPU service time, (Ts(CPU)), which is used by VSAM, has some effect on the total response time, we offer suggestions on how to decrease it.

2.7.1 Performance scenario using RMF reports

Before we start let us say a few words about RMF™.

RMF is an IBM program product which measures z/OS system performance. RMF has three Monitors and a Postprocessor function.

Each Monitor has a Data Gatherer and a Data Reporter. For Monitor I and Monitor II, both functions are clustered in the same address space. For Monitor III, they are in distinct address spaces.

- Monitor I is an ongoing non-interactive monitor measuring system variables.
- Monitor II is interactive, showing data about address spaces.
- Monitor III is interactive, showing the performance of specific transactions

In this VSAM performance management topic, we use an approach in steps that are based on RMF Monitor III, and assumes you are in WLM goal mode:

1. Look for your most important service class period that is not reaching the goal in the RMF Monitor III SYSSUM report (see Figure 2-18).

```
 ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
              Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

BATCH    W          100                               0.000 0.000  0.000  0.000
BATCHLOW S  5    25 100                          0.25  0.000 0.000  0.000  0.000
HTTPW1   W          100                               0.000 0.000  0.000  0.000
HTTPS1   S  1        25   0.80 AVG  1.60   AVG   2.0  44.4  0.000  1.600  1.600
OMVS     W           97                               0.030 0.001  0.711  0.712
OMVS     S           98                               0.030 0.001  0.711  0.712
         1  2       0.0  0.500 AVG  0.001  AVG  0.00  0.010 0.000  0.001  0.001
```

Figure 2-18 RMF SYSSUM report

HTTPS1 is a service class associated with the transactions of an HHTP scalable server. These transactions code run in tasks under enclaves. Its importance (I) is one (the maximum), its goal is average response time of 0.80 seconds. Its actual response time is 1.6 seconds and the performance index is 2.0 meaning 100% out of the target.

2. Let us look at the Enclave report in Figure 2-19 to see where the problem is:

```
ENCLAVE Attribute  CLS/GRP  P Goal    %   D    EAppl%   TCPU    USG  DLY  IDL

ENC0003 CCT        HTTPS1   1 0.80    25         18.75  26.78    30   88  0.0
        HTML
        COLLOR
ENC0001 CTT        HTTPS1   1 0.80    24         16.27   23.12   29   89  0.0
          HTML
        CARDOSO
```

Figure 2-19 RMF Enclave report

3. In this report, you see two enclaves belonging to the service class HTTPS1 suffering 88% and 89% of delay, respectively. It means that those transactions for this percentage of time do not progress. There are two types of resource delays, as far as WLM is concerned:
   - Delays for resources controlled by WLM. A resource controlled by WLM is the one where WLM can change the priority of the request in the queue. Therefore, those delays are measured by WLM in order to be minimized (by playing with queue priority), depending on the performance index of the service class. They are:
     - CPU
     - I/O (if you are in WLM I/O Management option)
     - Storage

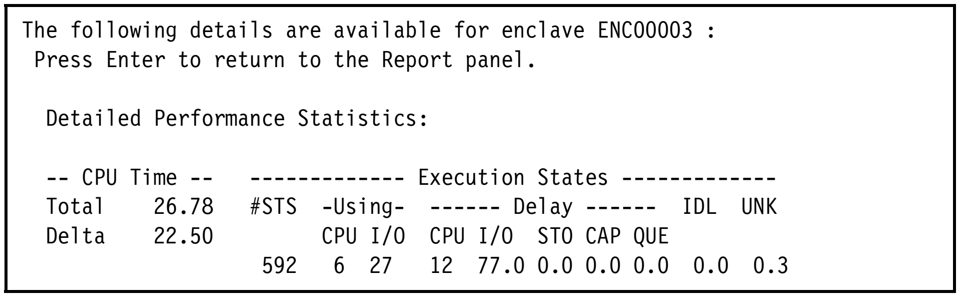
- Delay for HTTP Queue Server
- WLM capping

The 88% and 89% values pictured in the at RMF report refer to such WLM managed resources.

– Here is a list of delays tracked by RMF Monitor III but not managed by WLM:

  - ENQ delays
  - Operator delays (mount and messages)
  - Subsystem delays (JES, HSM and XCF)

4. Now let us zoom in to these delays in the report Enclave Classification Data in Figure 2-20:



```
The following details are available for enclave ENC00003 :
 Press Enter to return to the Report panel.

  Detailed Performance Statistics:

  -- CPU Time --   ------------- Execution States -------------
  Total    26.78   #STS  -Using-  ------ Delay ------  IDL  UNK
  Delta    22.50         CPU I/O  CPU I/O  STO CAP QUE
                    592   6  27   12  77.0 0.0 0.0 0.0  0.0  0.3
```

Figure 2-20 Enclave Classification Data report

5. As you can see, detailing the enclave ENC00003, 77% of the WLM managed delays are caused by I/O delays, that is, I/O requests from the tasks enclaves being delayed in the UCB (IOS queue time) or in the channel subsystem (pending time). The Using I/O value is 27%. The Using I/O concept has an ambiguous meaning depending on who is creating such figure:

   – In RMF terms (as the case) the task enclave application is executing a channel program (connected plus disconnect) for 27% of the observed time.

   – In WLM terms, the I/O disconnect time is not included. WLM uses this figure to derive the Execution Velocity values

6. Next we zoom a little more, going to the Monitor III Device Delay report (DEV) in Figure 2-21.

```
            Service  DLY USG CON  ------------ Main Delay Volume(s) -
Jobname  C Class    %   %   %    %  VOLSER   %  VOLSER   %  VOLSER

HTTPS000 S SYSSTC    77  27  23   70 VSMS15  11 VSMS19
MICHAELL B NRPRIME   39  15  14   39 BPXLK1
MCPDUMP  S SYSSTC    36  18  20   36 D24PK2
CHARLESR B NRPRIME   33  13  13   28 BPXLK1    3 HSML02    2 BPXSSK
DFHSM    S SYSSTC    30  83  35   10 HSML17    5 SMS026    4 HSMOCD
SHUMA3   T TSOPRIME  18  52  53   13 D83ID0    5 HSML02
```

Figure 2-21   Monitor III Device Delay report

To date, there is no RMF report showing details (at a volume level) about the I/O use and I/O delay of an enclave. You must know the name of the address space where the HTTP transactions are doing I/O. It must be one of the HTTP server address spaces. In this case it is the HTTPS000. Here we can see that the volume VSMS15 is responsible for 70% of the delays experienced by the enclave the HTTP transactions.

7. Then, we go to the Monitor III DEVN report in Figure 2-22 to see more details about the volume.

```
Device Identification --    -- Activity --     ACT CON DSC - Pending - - Jobs -
VolSer Num  Type    CU      S  Rate RspT IosQ    %   %   %   %  Rsn. %  USG  DEL

VSMS15 006C  33903  3990-3   S  42.1 .022 .006  68   8  60   2 DB    1  0.0  0.8
VSMS10 0051  33903  3990-3   S  80.7 .011 .005  47  24   1  22 DB   11  0.2  0.7
JOBL17 0703  33903  3990-3   S  52.2 .015 .000  76  22  54   0          0.2  0.6
TSO015 006E  33903  3990-3   S  11.1 .024 .001  26   3  20   3          0.0  0.3
VSMS06 0056  33903  3990-3   S   8.9 .034 .001  30   9  18   3 DB    2  0.1  0.2
```

Figure 2-22   Monitor III DEVN report

In the DEVN report for volume VSM15, we have the I/O response time (.022 seconds), the I/O rate (42.1 I/Os per second), the IOSQ time (0.006 second) and the percent distribution of connect, disconnect and pending. If you want to know how much of connect per I/O operation (AVG CONN TIME) follow this line of thought: “If the RMF range is 100 seconds, the total connect time was 8 seconds. In 100 seconds, it was executed (42.1 * 100) I/O instructions. So, dividing 8 by 4210, we have 2 milliseconds of AVG CONN TIME”.

8. The final step in using RMF reports is to discover the data set involved in the performance problem. We can do this through the Monitor III DSNV report in Figure 2-23.

```
------------------------- Volume VSM15 Device Data --------------------------
Number:   006C        Active:       68%      Pending:    1%     Average Users
Device:    33903      Connect:       8%      Delay DB:   1%        Delayed
Shared:    Yes        Disconnect:   60%      Delay CU:   1%          1.4
                                             Delay DP:   0%
-------------- Data Set Name ---------------   Jobname   ASID  DUSG% DDLY%
```

Figure 2-23 Monitor III DSNV report

Finally we have all the information — the volume and the data set name (which, incidentally, is a VSAM data set PROD.KSDS.MARCH.U12). You should determine the largest figure between IOSQ, pending, connected, and disconnected in your installation (in our example it is disconnect time). IOSQ is better to pick it up in average ms per I/O operation frm the DEVN report. Depending on the one you pick, go to the corresponding topic in this chapter, where you find recommendations to improve it. Remember that all of these suggestions refer to one of the three ways of solving performance problems, that is: *buy*, *tune*, or *steal.*

However, before you try to find a way to improve your I/O performance, refer to 2.7.2, “Reducing the number of I/Os” on page 119, which offers suggestions to eliminate this problem.

2.7.2 Reducing the number of I/Os

The general rule, that the best I/O is the one that is not executed, still applies to VSAM. Therefore, before trying to improve the I/O operation, let us focus on how to avoid these I/Os. The general techniques for doing that are explained in the following sections.

Buffering

Buffering is one of the most important VSAM features that can be used to avoid I/Os, and consequently to improve performance. There are two types of buffering: VSAM controlled buffer pools, and Hiperbatch.

VSAM controlled buffer pools

VSAM controlled buffer pools are controlled by LSR, NSR, RLS, and GSR buffering modes. Some of them allocated in the application address space and some in the hiperspace. Refer to 2.6.13, “Buffering options” on page 63.

Let us look at the value of buffering from the perspective I/O performance by creating the following scenario.

Scenario

- KSDS cluster requiring 0.5 GB (512 * 1024 * 1024 bytes) in a 3390
- Data CI size is 4K
- FREESPACE (15 9)
- The control area size is not restricted, then it is 15 tracks
- There is 180 data CIs in each control area
- The index control interval size is 1536 bytes
- There is 729 Control Areas, and therefore 729 index sequence set CIs
- There is 5 index set records
- There is 3 index levels

Supposing no hits in the buffer

- Program application generating 45 random reads per second, with a M/M/1queue behavior. This M/M/1 means normal distribution for the inter-arrival time (M) and for the service time (M) plus just one (1) server
- 100% of DASD caching hits for index, each I/O at 0.5 msec
- 100% of DASD caching hits for index causing an average service time of 0.5 msec
- 50% of DASD caching hits for data, each causing an average service time of 6 msec

Calculation

- I/O Service_Time per average Read caused by three index I/Os and one data I/O: (3 x 0.5 + 6) = 7.5 ms
- Device Utilization (U)% = IO_Rate x Service_Time x 100= 45 x 0.75 = 33.75%
- I/O Queue_Time = I/O_Service_Time x (U / (1-U))= 3.8 ms
- I/O_Response_Time = 7.5 + 3.8 = 11.13 ms

Supposing all index set in the buffer

Supposing all index set in the buffer (66% buffer hit) and 50% buffer hits for data:

- I/O Service_Time per average Read caused by one index sequence set I/O and one data I/O: (1 x 0.5 + 3) = 3.50 ms
- Device Utilization (U)% = IO_Rate x Service_Time x 100= 45 x 0.3 = 15.75%
- I/O Queue_Time for M/M/1= I/O_Service_Time x (U / (1-U)) = 0.65 ms
- I/O_Response_Time = 3.00 + 0.65 = 3.65 ms
- I/O_Response_Time gain proportion = 11.13 / 3.65 = 3.04 times faster

The price of the I/O improvement.

To improve 304% your I/O use buffers, and consequently a trade off between I/O and memory. How much more buffers?

- For five index set buffers: 5 x 1536 = 7680 KB
- For data buffers calculation we may use a derivation of the 80/20 rule. To have 80% hits we need 20% of buffers. To have 50% hits (as in the scenario) maybe we just need 10 percent. We have 180 x .91 = 163 CIs not free per CA.

  10% of the total number of CIs is: 729 x 163 x 0.1 x 4 _KB = 47.5_ MB

As you can see in an address space of 2Gb and with central storage much larger than 2Gb the storage price is minimum for the I/O performance gain We have a bargain.

Hiperbatch

Hiperbatch is designed to eliminate the problems caused by:

- Multiple jobs in one z/OS image requesting data from the same QSAM/VSAM NSR data set simultaneously. Each job causes I/O operations to the device holding the data set. The more jobs that access the data set concurrently, the more I/O operations and contention for the device, and the longer the wait time for each I/O request.
- Jobs or job steps passing temporary or short-lived QSAM or VSAM NSR data sets to subsequent jobs. As a job completes, it puts the data used back onto DASD.

One important aspect of Hiperbatch is that installations can take advantage of these performance benefits without having to change existing application programs or the JCL required to run them.

Hiperbatch depends on the data lookaside facility (DLF) address space, to control access to an Hiperspace Expanded Storage Only (HS ESO).

RACF DLFCLASS profiles provides the data set name list that DLF needs. The existence of a DLFCLASS profile for a VSAM data set identifies that data set as one that is eligible to be processed as a DLF object.

When DLF is active, the first attempt to access a QSAM or VSAM data set defined to DLF causes it to create a DLF object (like a data set in the HS ESO). A DLF object contains data from a single data set managed by Hiperbatch. The user (an application program) is connected to the DLF object, and the connected user can then access the data in the object through normal QSAM or VSAM macro instructions.

When subsequent users access the data set, they are connected to the object. The system manages shared access to the DLF object in the same way it would manage shared access to the data set. When a user relinquishes access, DLF disconnects that user from the object.

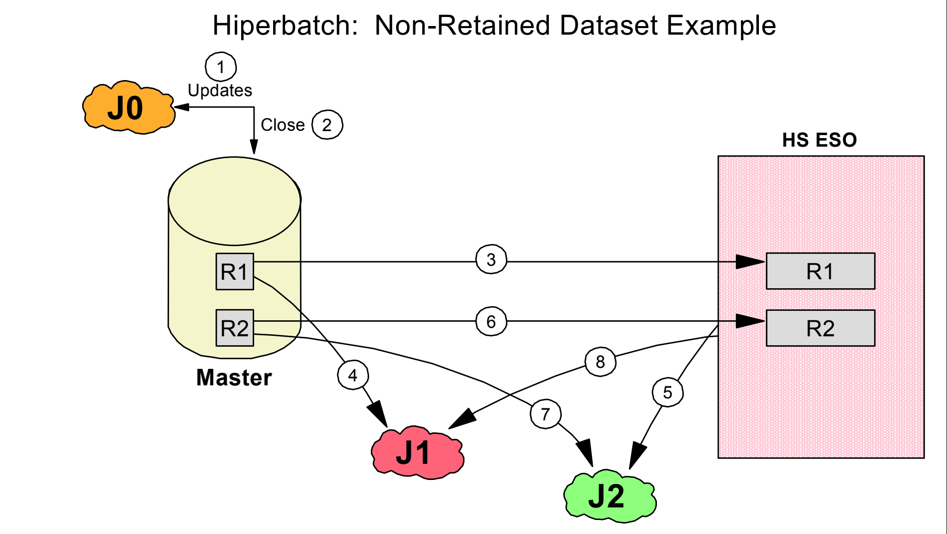
A DLF object exists until there are no users of the data set, at which time DLF deletes the object. As long as there is at least one user of a data set the access pattern means that the DLF object exists.

However, if a batch job or job step creates a data set and passes it as input to another job or job step, there is not always one user of the data set, and the system would delete the DLF object. To prevent this automatic deletion of an object, define the data set to DLF as a *retained DLF object.* A retained DLF object is one that the system does not automatically delete when there are no users of the data set.

Refer to Figure 2-24 for the following example:

We have a non-retained data set (called Master), that is a DLF object. All the reading jobs should be started in parallel (as usual without Hiperbatch). The first job (J1) reads sequentially the record one (R1) and suffers the I/O delay. After the I/O completion, one copy of R1 is delivered to J1, and another copy is kept in the HS ESO by Hiperbatch. When J2 requests an I/O for R1 reading, it is intercepted and fulfilled with the R1 copy in the HS ESO. Then, for N Jobs reading M records each, from the Master, we have only M accesses to DASD, instead of M * N.

Figure 2-24 highlights the I/O operations not executed.



Note: ⓧ means chronological events

(1) J0	Updates master
(2) J0	Closes and unlockes master
(3) J1	Reads R1 (copy to HS ESO)
(4) J1	R1 (copy to J1 buffer)
(5) J2	**Reads R1 (from HS ESO)**
(6) J2	Reads R2 (copy to HS ESO)
(7) J2	R2 (copy to J2 buffer)
(8) J1	**Reads R2 (from HS ESO)**

Figure 2-24 Hiperbatch example

Here are some comments about using Hiperbatch with VSAM:

- VSAM non-shared resource (NSR) access is a requirement.
- Hiperbatch does not support extended format data sets.
- VSAM organizations KSDS, ESDS, and RRDS with a control interval of 4096 bytes, or a multiple of 4096 bytes, are eligible.
- The EXCP counts in SMF records used to record I/O use do not change; the counts reflect I/O operations requested regardless of whether a request is satisfied by a physical I/O operation or from a DLF object in expanded storage.
- If the data set is a VSAM key sequenced data set (KSDS), the DLF object contains only the data component, not the index component.

- To avoid data integrity exposures, Hiperbatch does not process a VSAM data set with shareoptions 3 or 4, even if that data set has been defined as eligible for Hiperbatch.
- VSAM data sets with the number of strings (ACBSTRNO) value greater than 1, cannot be Hiperbatch objects.
- VSAM data sets must be opened by the base cluster name.
- If a random (or sequential) program changes data in the data set, that change is also made to the DLF object.
- This method is best suited for sequential processing.

IBM provides an online monitor that you can use to track Hiperbatch activity on your system. Using the monitor, you can evaluate:

- How Hiperbatch is using expanded storage (or central storage in z/Architecture™)
- The use patterns for individual data sets
- The I/O operation savings for individual jobs

For information about how to install the Hiperbatch monitor, see *MVS Hiperbatch Guide*, GC28-1470. Once it is installed, you can invoke the monitor under TSO/E by issuing the following command:

```
COFDMON
```

I/Os associated with CA splits

These I/Os should be avoided. CA splits generate many I/O operations. CA splits occur in a KSDS or VRRDS along inclusions and increase the logical record size during an update. CA splits may be minimized by the use of CA free space. Refer to “Splits” on page 15. However, a CI splits is not a real performance problem because it needs less than five I/Os.

Secondary space allocations

Every secondary allocation implies going through End-of-Volume processing with several I/Os in the catalog and VTOC. Refer to 2.6.1, “Allocation units” on page 49. You should require a consistent amount of secondary allocation.

Write checks

Write checks needlessly increases the number of I/O operations. These should be avoided.

RECOVERY option

Use the SPEED option instead, when loading a VSAM data set. Refer to 2.6.11, "Initial load option" on page 59.

2.7.3 I/O wait time (IOSQ) for VSAM data sets

Tw(IO) has two components respectively: IOS Queue Time and Pending Time. First lets consider IOSQ time. Let us suppose that you are reading this because the IOSQ time is the dominant factor in the I/O of your VSAM data set.

IOS Queue Time is the time waiting for the device availability in the z/OS operating system. For a non-ESS device, Input Output Supervisor (IOS) does not start an I/O operation to a device if there is a previous one executing. In this case, the I/O operation is queued in the UCB (a control block representing the device to IOS).

A queue starts to build up when the unique server (device) is utilized above 35 percent. This rule applies to the IOSQ Time. This utilization can be caused by activity coming from this sysplex image or from another image (in a shared DASD case), or both.

To decrease the IOSQ Time, you can:

- Buy a faster device/channel to decrease the I/O Service Time (Ts(IO)) and consequently the utilization (for the same I/O load) and the queue time.
- Decrease the Ts(IO) by tuning. Refer to 2.7.4, "I/O service time (connect) for VSAM data sets" on page 129.
- Change the service class goal containing the transaction which is generating the I/Os causing the IOSQ time delay. Increase the importance of the goal or change its numerical value to make it more difficult to be obtained. In consequence, the I/O priority is raised by the WLM goal mode. This happens when the transaction is not reaching its goal and the major delay is the I/O delay. Be aware that, in this case, you are not improving the I/O in general, but just improving the response time of your favorite transactions.
- Decrease the I/O rate against the device by avoiding placement of several active data sets on the same volume, (mainly index and data from the same KSDS). If this happens, verify your ACS routines, perhaps by using guaranteed space to force the index in a specific volume. Use different storage groups or the new function DFSMS Data Set Separation announced in DFSMS z/OS 1.3, where you can separate data sets from each other in different physical DASD controllers.
- Increasing the parallelism as a way to decrease queue times:

- If your controller has support for dynamic parallel access volume (PAV) as in the ESS controller, the most effective action to decrease your high IOSQ time is to implement such hardware/software feature. With PAV you may have parallel I/Os against the same logical 3390/3380 device (two writes in the same extent are not allowed) consequently decreasing the IOSQ time. However, PAV has a price that is, to have *N* parallel access, we need to have *N* alias UCBs and UCWs. If dynamic PAV is activate, WLM manages the number of UCBs and UCWs per device, increasing the parallelism in certain devices (and decreasing in others) in order to fulfill the goals of the most important service classes. Refer to "VSAM and ESS controllers" on page 134.
- Use the great level of parallelism implemented through FICON channels.

DASD cache highlights

A DASD cache is fast storage located in the controller. In a sense it resembles the buffer pool in memory. A cache, has two functions:

- Minimize access to disks (by having cache hits).
- Serve as a *speed matching buffer* to synchronize hardware elements with different speeds (like channels and disks) along a cache miss during sequential access.

In this explanation the term disk does not have the same meaning of DASD. Disk implies the RAID media, where data is stored in fixed block architecture (FBA) blocks through an SSA or an SCSI protocol as used by modern controllers. DASD is the logical 3390/3380 device as perceived by you, your application, and your MVS operating system.

To have random cache hits (saving disk access) for reads and writes, the I/O workload must frequently access the same data or index CI in VSAM terms. Typically there are two types of hits, when the application revisits data:

- Exactly the same logical record, in a CI is already in cache.
- Different logical record in the same CI already in cache because another logical record was previously accessed.

For sequential access, it is important to say that cache does not save data CI disk I/O operations. The cache only tries to match the speed of the disks and channels. Consequently, the faster resource is less utilized. The controller has a sequential algorithm for reads that implements a look ahead mechanism.

VSAM hints to decrease disconnect time due to cache

If the average disconnect time of your key VSAM data sets is above two milliseconds, read this topic. In our example, refer to the scenario described in Step 6 on page 117.

In the explanations given in this chapter, we still use Monitor III data.

For this performance scenario, let us look at the RMF Monitor III Volume Cache report in Figure 2-25.

```
The following details are available for Volume VSM015 on SSID 0043
Press Enter to return to the Report panel.

Cache: Active       DFW: Active       Pinned: None

       ------ Read ------   --------- Write ---------   Read   Tracks
        Rate    Hit  Hit%    Rate   Fast    Hit  Hit%     %
Norm     3.7    3.6  79.8     0.8    0.8    0.8   100   82.2      0.1
Seq      0.0    0.0   100     0.1    0.1    0.1  93.3   21.1      0.0
CFW      0.0    0.0   0.0     0.0    0.0    0.0   0.0    0.0
Total    3.7    3.7  98.9     0.9    0.9    0.9  99.1   80.0

------ Misc ------  - Non-Cache -  --- CKD ----  - Record Caching -
DFW Bypass :   0.0  ICL   :   0.0  Write:   0.0  Read Miss :   0.0
```

Figure 2-25 Volume Cache report

There are no cache reports per data set in RMF. You can use the volume (the one that contains your VSAM data set) report to reach your conclusions.

Look at the Inhibit cache load (ICL) value first. If it is consistently non-zero, this may mean:

1. IDCAMS does not properly set the cache attributes in the volume, verify this by checking whether CACHE and DFW are active in the report header. Then, change it to normal and DFW.
2. It is an SMS managed data set, and its cache attribute is never-cache. Then, change its MSR value to allow always-cache or may-cache.
3. It is an SMS data set, and its cache attribute is may-cache. Then change it to allow always-cache.

If, after making the modifications listed above, the disconnect time does decrease, or your ICL is close to zero, we have one of two cases:

- Cache overloaded.
- Your data set is cache-unfriendly.

Follow the logic below to determine in your situation. If we reach the conclusion that your data set is cache unfriendly, better to undo modifications 2 and 3 above, and read “Decrease VSAM disconnect time in general” on page 129.

- If the access is bounded to writes (READ% consistently below 50%), take a look in the WRITE HIT% column.

  Pick up the line corresponding to the highest value in Write RATE (between NORM and SEQ).

  – If NORM is the larger, you have predominantly a random access. Compare in the NORM line, the FAST value with RATE.

    • If they are not the same (very seldom), it implies that SMS is not always using DFW cache mode.

    • If they are almost the same, check the WRITE HIT% value. It must be higher than 95%, if less the controller “almost 100% quick writes” (almost 100% of the random writes should be hits) is not working.

  – If SEQ is the larger value (you are writing sequential) and WRITE HIT% less than 95%, take a look in the DFW Bypass figure. If consistently non-zero, this means:

    • The write sequential load saturated the NVS and DFW bypass occurred. It could be the records are sent synchronously from volatile cache to disks, or it could be the records are sent asynchronous from volatile cache to disks and this time is included in disconnect time. In this case your workload is the cache-unfriendly type. Refer to “Decrease VSAM disconnect time in general” on page 129.

- If the access is dominant in reads (READ% consistently above 50%), look in the READ HIT% column. Pick up the value corresponding to the highest value in READ RATE (between NORM and SEQ).

  – If NORM is the larger, you have a random access. If the corresponding READ HIT% is less than 80%:

    • A low random Read hit. This means that your reads are candidates for using cache, but they are not having enough hits. Your workload may be cache unfriendly, if so, here are two suggestions:

      Be sure that SMS is using record level cache (RLC) for reads. It avoids polluting the cache with the rest of the logical 3390/3380 physical track.

      Try using a smaller CI for read data or less free space in the CIs. It avoids polluting the cache with other non-referenced logical records or free space.

      Read “Decrease VSAM disconnect time in general” on page 129

  – If NORM is the smaller, you have a sequential access. If the corresponding READ HIT% is less than 80%:

- A low sequential read hit. This means that the speed of the channel is higher than the disk speed. In this case refer to “Decrease VSAM disconnect time in general” on page 129.
- The KSDS has many CI splits, impeding the controller in recognizing the sequential pattern; then there is no look-ahead and the cache is treated in LRU mode. Reorganizing the data set may be an answer. Refer to 4.1, “Reorganization considerations” on page 204.

Decrease VSAM disconnect time in general

One of the reasons to be reading this section is that you have reached the conclusion that your data set is cache unfriendly.

If cache cannot help your VSAM data set, you need to reduce the contention on the disks. Here are some suggestions.

- Reducing I/O rate (demand) against the controller. This can be done by:
  - Either compression or use of smaller CIs for random processing. If you are accessing data randomly, try to use a smaller CI for read data or less free space in the CIs. It avoids polluting the cache with other non-referenced logical records or free space

    For compression, refer to 2.6.15, “Data compression” on page 94. For use of smaller CIs, refer to 2.6.6, “Control interval size” on page 54.
  - Reorganize of the data set, if there is plenty of free space in the CIs. Refer to 2.7.4, “I/O service time (connect) for VSAM data sets” on page 129. However, this reorganization may cause splits.
  - Reducing the number of active data sets in the controller.
- Decrease the number of writes in the controller by moving data sets, to avoid the effect of write penalty in RAID-5. Use the SHARK feature, JBOD, (“Just a Bunch of Disks”) for your temporary data sets to get rid of write penalty. Use Shark 800 RAID-10 (RAID-1 plus striping) well suited for intensive write workloads.
- Try to allocate your data set in 3390/3380 logical volumes mapped in less capacity disks (18.2 Gb) spinning at 15,000 RPM instead of 10,000RPM.

2.7.4 I/O service time (connect) for VSAM data sets

Connect time means the period of time when the channel is transferring data from or to the cache or exchanging control information with the controller about one I/O operation.

The average connect time per I/O operation depends on:

- The speed of the communication (including protocol) between the channel and the controller.
- The amount of data transferred per I/O operation (in all of the chained Read or Write CCWs)

To analyze your I/O connect time, we recommend that you know the type of access your VSAM data set is facing: sequential or direct. Let us divide our analysis following these two types:

- Sequential access. If you did not change your channel and controller and you observe a relatively high connect time for your sequential access VSAM data set, things are working well. It means that a great deal of data is being transferred in just one I/O operation, through one CCW with a big blocksize or many CCWs (many buffers). So, the recommendation is to increase the average connect time. You can get that by using plenty of buffer space in the buffer pool for sequential processing.

  Doing that, you decrease the number of SSCHs instructions (I/O operations). For every I/O operation starting, there is a standard conversation between the channel and the controller. If you decrease the number of SSCH instructions, but are transferring the same amount of data, you save in total connect time. In other words your average connect time (per I/O operation) increases, but your total connect time decreases. Refer to Table 2-5 on page 73 and Table 2-14 on page 101. If the cluster is going to be accessed sequentially and randomly, the CI size should be made smaller (4 KB). To solve the performance problem in sequential access, you can define many data buffers, thus causing VSAM to chain CIs in just one channel program. Refer to 2.6.6, “Control interval size” on page 54.
- Direct access. Here the smaller the average connect time, the better. You should have small CIs. This avoids bringing unneeded logical records into memory.

Decrease average connect time

What follows a set of recommendations to decrease the average connect time for both direct and sequential accesses:

- Use FICON native or FICON Express channels for sequential access. For that your controllers must have a host adapter able to understand such protocol.
- Use data compression in CPU:

  There is a difference between compaction and compression. Compaction has to do with the simple task of extracting blanks and zeroes from a text. Compression is a much more elaborate task, where dictionaries may be used to obtain the best result (like the Ziv-Lempel algorithm). These dictionaries contain the most repeated set of characters found in the text and their respective compressed substitution. Refer to 2.6.15, “Data compression” on page 94.

- Use the ECKD™ extended format:

  ECKD extended format (also called extended format) is a technique that effects the way count key data is stored in a 3390/3380 logical track. It improves performance for certain types of I/O operation, by decreasing Ts(I/O) with a better channel program. Extended format is the base for using VSAM features such as SMB, data striping, compression. and It is recommended that, if time permits, you convert your data sets to extended format to get this better performance. Refer to Table 2-15, showing the following results from our tests of random processing: extended format versus non-extended format data sets.

Table 2-15 Random processing: extended format versus non-extended format

Ext format	Buffering	EXCPs	Connect time (sec)
No	Default	861744	124
Yes	Default	764333	120

  Here, for random processing, you can see some decrease in the number of EXCPs and in the connect time. However, the big performance appeal of extended format is that it allows the use of strong performance capabilities such as striping, compressing, and SMB.

- Less free space in the CIs:

  The existence of a significant amount of free space in a CI, may increase the I/O connect time. This free space may be caused by an excess in the definition of the data set, or by CI and CA splits. Remember that any CI with a logical record is moved to storage in a sequential read. This is not true with totally free CIs, where no I/O operations are executed towards them.

Note: We are not saying that all insertions increase the average free space per byte in the data set.

  The solution here may be a reorganization. Refer to 4.1, “Reorganization considerations” on page 204, watching for an increase in the number of splits.

- Parallel VSAM I/O operations:

Sometimes, if you cannot decrease the service time (Ts), you may increase the ETR (and decrease the queue time) by introducing parallelism. There are two types of I/O parallel processing:

- Within a transaction:

  The VSAM option which allows I/O parallelism within a transaction or task (when accessing a VSAM cluster) is data striping. Refer to 2.6.16, “VSAM Data striping” on page 103.

- Between transactions:

  There are VSAM options which allow parallelism between transactions or tasks, when accessing a VSAM cluster, such as:

  - STRNO: In multiple string processing, there can be multiple independent Request Parameter Lists (RPL) within an address space for the same data set. The data set can have multiple tasks that share a common control block structure. There are several ACB and RPL arrangements to indicate that multiple string processing will occur.
  - In the first ACB opened, STRNO or BSTRNO is greater than 1.
  - Multiple ACBs are opened for the same data set within the same address space and are connected to the same control block structure.
  - Multiple concurrent RPLs are active against the same ACB using asynchronous requests.
  - Multiple RPLs are active against the same ACB using synchronous processing with each requiring positioning to be held.

  Refer to *z/OS DFSMS Using Data Sets,* SC26-7410, for more information on multiple string processing.

In ESS, the PAV and Multiple Allegiance features implement I/O parallelism in the same volume, within or among transactions.

2.7.5 Decreasing VSAM CPU time

The major theme of this section is to decrease the Tr(I/O) for VSAM I/O operations, in order to decrease the response time of key transactions using VSAM data sets. However, chances are that the enhancements introduced by your changes may shift the bottleneck to the CPU side. You will find that:

- If you compress your VSAM data set, your transaction response time decreases, which is good.
- If you stripe your VSAM data set, your transaction response time decreases, which is also good.
- If you compress and stripe your VSAM data set, your response time gets bigger (which is bad). The reason is that CPU is extremely busy, causing huge CPU delays negating the I/O gains.

VSAM buffering

The adequate use of buffering (mainly for direct access) may result in savings in the CPU usage, as you can see in Table 2-16.

Increasing the number of buffers decreases the number of EXCPs, that is, the number of executed I/O operations. In VSAM: one EXCP corresponds to one SSCH and corresponds to one I/O operation. For the same amount of logic in your code, the CPU time that you spend is a direct function of the number of EXCPs. Because in our lab, the read data is not processed, we can say that:

- SRB time is caused by I/O interrupts (back end) processing.
- TCB time is the I/O operation preparation and buffer pool management.

It is clear that increasing the number of buffers means the management cycles dominate the savings in the I/O operations. Refer to “Our test environment” on page 396.

Table 2-16 NSR: Read sequential varying the number of buffers

Data buffers	**Index buffers**	**EXCPs**	**SRB time (seconds)**	**TCB time (seconds)**
Default=2	Default=1	167828	2.12	7.52
10	1	33568	0.53	3.99
30	1	11577	0.26	3.44
181	1	3476	0.19	3.86
SMB		4633	0.23	3.45

Also, the use of better techniques to manage your buffer pool, such as BLSR or system management buffers (SMB), can save enormous CPU cycles. See Table 2-17, which shows data obtained from our lab.

Table 2-17 Direct access: benefits of using SMB: Reads and insertions

Ext format	**Buffering**	**EXCPs**	**connect time (sec)**	**CPU time (sec)**
No	Default	861744	123.5	52.46
No	BLSR	330852	73.14	24.42
Yes	SMB	280276	62.5	19.22
Gain using SMB (%)		**68%**	**50**	**63**

Use of extended format

The use of extended format data sets can save a consistent amount of CPU time (TCB plus SRB). Consider the values shown in Table 2-18.

Table 2-18 Direct access: benefits of using SMB: Reads

Ext format	Buffering	EXCPs	CPU time (sec)
No	Default	794950	31.76
Yes	SMB	102697	8.53
Gain using SMB (%)		**87%**	**73%**

Refer to 1.9, "Extended format data set" on page 28, for more information.

Compression

Compression of data can really increase the CPU time in your program.

- Track versus cylinder allocation

  It is not true anymore that, when allocating VSAM data sets in cylinders, you consume less CPU than using track allocation. Because, if the channel program is crossing an extent boundary, it is informed about the correct extents through the Define Extent CCW.

2.8 VSAM and ESS controllers

In the previous versions of this book, we made extensive tests with VSAM clusters in order to measure how effective some VSAM features are. All of these experiences were made on a ESS model F model. In this version of this book, we change completely our I/O workload. However, we repeat two of those old measurements (with the old I/O workload), just changing the type of channels (from ESCON® to FICON) and using the new ESS model 800. The objective is to compare the gains in VSAM by just changing the I/O hardware.

2.8.1 ESS model 800 enhancements

Before we show the results, we cover the new main enhancements of ESS model 800 Shark in order to correlate them with the measured results:

- More processor power

  Model 800 offers 4 x 600 MHz processors per cluster (old model F20 it was 4 x 225 MHz per cluster). There is also a turbo feature implementing 6 x 668 MHz per cluster (our ESS is not a turbo).

- Aggregate bandwidth

  Model 800 allows 4.8 GB/second of aggregate bandwidth against 2.7 GB/second in model F20.
- Up to 64 GB of cache, formatted in 4 KB segments, so for small data blocks (4 KB and 8 KB are common database block sizes) minimum cache is wasted. Our CI size is 4 KB.
- High performance 15,000 rpm disks drives. We did not use such disks in our tests
- New CCWs. For the z/OS environments, the ESS supports channel command words (CCWs) that reduce the characteristic overhead associated to the previous (3990) CCW chains. Basically, with these CCWs, the ESS can read or write more data with fewer CCWs. CCW chains using the old CCWs are converted to the new CCWs whenever possible. The cooperation of z/OS software and the ESS provides the best benefits for the application's performance.

2.8.2 Lab experiments

We repeated two runs, as we did in the previous version of this book. The first one accessing sequentially and the second accessing randomly:

- Initial load with speed and SMB.

  The results are shown in Table 2-19.

Table 2-19 Comparison initial load with ESS

Initial Load	Extended Format	EXCPs	Connect Time (msec)	CPU Time (sec)	Elapsed Time (sec)
Shark - F	YES	2341	21.5	2.7	27
Shark-800	YES	2401	5.6	1.81	10

  As you can see, there is four times less connect time for almost the same number of EXCPS. What Shark and FICON can do is to decrease the connect time, and they did it. About the number of EXCPs depends on the number of buffers. There is a decrease in CP time because we jump from a 9672 to the z900 1C7.
- Direct access with SMB (inserts and updates)

Table 2-20 Comparison Direct access with ESS

	SMB	EXCPs	Connect Time (sec)	CPU Time (sec)	Elapsed Time (sec)
Shark - F	YES	79,741	75	11.04	150
Shark-800	YES	28,405	6	3.99	7

The connect time per EXCP in the first run is 0.9 ms. The same figure in the second run is 0.2 ms. So, the gain in connect was more than 8 fold. Note also the decrease in the elapsed time.

2.9 Performance monitors

Follows a list of performance monitors, whose output can help you in solving SMSVSAM performance problems.

2.9.1 Resource measurement facility

RMF product issues reports about performance problems as they occur, so that your installation can take action before the problems become critical. Your installation can be used RMF to:

- Determine that your system is running smoothly
- Detect system bottlenecks caused by contention for resources
- Evaluate the service your installation provides to different groups of users
- Identify the workload delayed and the reason for the delay
- Monitor system failures, system stalls, and failures of selected applications

RMF comes with three monitors, Monitor I, II and III.

- Monitor III with its ability to determine the cause of delay is where we start. It provides short term data collection and online reports for continuous monitoring of system status and solving performance problems. It allows the system tuner to distinguish between delays for important jobs and delays for jobs that are not as important to overall system performance
- Monitor I provides long term data collection for system workload and resource utilization. The Monitor I session is continuous, and measures various areas of system activity over a long period of time.

  You can get Monitor I reports directly as real-time reports for each completed interval (single-system reports only), or you let run the Postprocessor to create the reports, either as single-system or as sysplex reports. Many installations produce daily reports of RMF data for ongoing performance management. Sometimes a report is called a Monitor I report (for example, the Workload Activity report) although it can be created only by the Postprocessor.

- Monitor II provides online measurements on demand for use in solving immediate problems. This Monitor II session can be regarded as a snapshot session. Unlike the continuous Monitor I session, a Monitor II session generates a requested report from a single data sample. Since Monitor II is an ISPF application, you might use Monitor II and Monitor III simultaneously in split-screen mode to get different views of the performance of your system.

In addition, you can use the Spreadsheet Reporter for further processing the measurement data on a workstation by help of spreadsheet applications. The following chapters provide sample reports including the name of the corresponding macro.

2.9.2 Tivoli Decision Support (TDS)

TDS is product contained in Tivoli® Information Management for z/OS suite of products.

Tivoli Information Management for z/OS is a process automation tool that delivers Systems Management Services, including Incident, Problem, Change and Asset management applications, for your end-to-end managed environment.

With TDS you can also have immediate access to key enterprise information about performance and capacity, stored in a DB2 database. TDS provides ready-to-use views and reports, so that you can identify key problems. You can design your own reports and views through the use of SQL language easy code.

Information viewed with TDS can be published on a Web site, or as an information ticker that streams across the user's desktop if your company uses broadcast tools. Users can then check the information before calling their internal help desk to report a problem.

2.9.3 Generalized Trace Facility (GTF)

GTF is an MVS component able to capture very detailed information about events occurring in the system. Among those events, we have: SSCHs and I/O interruptions. If you have a complex VSAM DASD I/O performance situation, you may run GTF with CCW trace option as shown in the "REXX code to list compression ratio" on page 384. When you can use IPCS to format the data produced by GTF or you may use a home made product.

For example, we use GTF data to calculate in our lab the total connected time associated to one data sets.

3

VSAM problem determination and recovery

In this chapter we provide some general tips on VSAM problem determination. We then go through a number of common VSAM problems and explain how to recover from them. We then follow with some suggestions to avoid problems and then we point you to some useful documentation that will assist you with your problem determination and recovery.

The topics cover:

- VSAM problem determination hints and tips
- Some common VSAM problems
- What documentation to collect
- How to recover a damaged VSAM data set
- Prevention is better than cure
- Where to look for more information
- IDC3009I message
- IDCAMS LISTCAT output fields

3.1 VSAM problem determination hints and tips

In the complex environment of computing today, users have an array of tools and disciplines at their disposal to aid problem determination and recovery of VSAM spheres and data sets. z/OS writes numerous records like SMF, LOGREC and syslog that can be used to debug a problem and recover data.

You may have heard user comments like these:

- Everything was working yesterday; we haven't changed anything since then.
- We are starting to get strange error messages, and the explanations in the manual are meaningless.
- We cannot access the data, and do not know where to turn for help.

Your data is one of the most valuable assets of your business. The challenge, in order to save your data, is to use all the error information available to answer these three questions:

- What caused this problem?
- How you can recover your data?
- What you can do to avoid these problems in the future?

In this section we will give you some general tips on what to look for and how to determine what caused the problem and how to recovery from a problem.

3.1.1 How to check your VSAM data set

A number of IDCAMS commands are available to check your VSAM data sets.

EXAMINE

You can use the EXAMINE IDCAMS command to analyze and report on the structural integrity of the index and data components of a key-sequenced data set cluster (KSDS) and of a variable-length relative record data set cluster (VRRDS). Any problems with the VSAM data set will be reported by one of the IDCxxxxx messages. Look in MVS System Messages, Volume 3 (GDE - IEB) for the explanation for these messages.

The EXAMINE command is described in detail in "EXAMINE command" on page 165.

VERIFY

The VERIFY command causes a catalog to correctly reflect the end of a VSAM data set after an error occurs while closing a VSAM data set. The error might have caused the catalog to be incorrect. For details of this command refer to "VERIFY command" on page 166

DIAGNOSE

The DIAGNOSE command can be used to scan a basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structures and detect structure errors. See "DIAGNOSE command" on page 166 for more details.

DFSMSdss PRINT command

With the PRINT command, you can print:

- A single volume non-VSAM data set, as specified by a fully qualified name. You must specify the volume where the data set resides, but you do not need to specify the range of tracks it occupies.
- A single-volume VSAM data set component (not cluster). The component name specified must be the name in the VTOC, not the name in the catalog.
- Ranges of tracks.
- All or part of the VTOC. The VTOC location need not be known.

LISTCAT

You can list the catalog entries using the LISTCAT command. It is described in "IDCAMS LISTCAT output fields" on page 187.

3.1.2 z/OS system messages

OS/390 components issue messages with the IEA prefix, associated with data set allocation, master scheduler functions, and RTM. The IOS prefix is for IOS functions. Usually those messages follow abends:

- IEC070I — RC32, RC202, RC104, or RC203
- IOS000I — Command reject (IOS errors in general)

3.1.3 Catalog Search Interface IGGCSIVS program

This Catalog Search Interface (CSI) produces a list of data set names defined in a given catalog that reside on a specific volume. Refer to 4.5, "Catalog Search Interface" on page 231, for more information.

3.1.4 System LOGREC messages

Often the system will recover from a problem and so will be transparent to you; but it will generate error records in the LOGREC. If you suspect a problem LOGREC may give you some valuable information.

3.1.5 GTF CCW traces

You can collect GTF CCW traces to get detailed information on I/Os to VSAM data sets. These traces can be formatted using IPCS. For sample JCL to collect a GTF trace refer to Example 3-6, “Sampler JCL for GTF trace” on page 165.

3.1.6 DITTO/ESA output

This utility can be used to browse VSAM records. For details, refer to “DITTO/ESA” on page 35

3.1.7 What can you get from the SMF records?

You can get a lot of information from SMF record types 60 to 69 to analyze VSAM problems. SMF record type 42 contains information on VSAM RLS statistics. Refer to 3.9, “SMF record types related to VSAM data sets” on page 194 for detailed information on these SMF records. You can use our SMF 64 sample program as described in that topic.

3.2 Some common VSAM problems

Here we describe some problems which may affect the processing and the existence of your VSAM data sets. To simplify our search for a solution to each problem, we can use the three “Whats”, for example, occurrence, recovery, and avoidance.

In each subsection we cover the three “Whats”:

- What happened to cause this problem?
- What must you do in order to recover your data now?
- What must you do to avoid this problem in the future?

Broken data sets can be caused by many different circumstances, including user errors. When diagnosing these types of problems, the first thing that must be done is to identify what is actually wrong with the data set. The first sign of a problem is the VSAM or the z/OS system messages. A single error can often generate numerous messages. You should focus your attention on the return code presented and the companion explanation. This return code will be the one passed by the system component that first encountered the error. In most cases, you will need additional documentation; refer to 3.3, “What documentation to collect” on page 162.

In this section we group the errors by categories. However, this is not an easy task. Some of the categories overlap and even interact with others. For example, a bad channel program may be caused by an improper sharing, which caused a structural damage.

3.2.1 Lack of virtual storage

The following messages may indicate a lack of virtual storage:

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 136 (Close): Not enough virtual storage was available in the program's address space for a work area for Close.
  - 132 (Open): One of the following errors occurred:
    - Not enough storage was available for work areas.
    - The format-1 DSCB or the catalog cluster record is incorrect.
  - 136 (Open): Not enough virtual storage space is available in the program's address space for work areas, control blocks, or buffers.
  - 40 (I/O): Insufficient virtual storage in the user's address space to complete the request.
- `IEC161I 001 [(087)]-ccc,jjj, sss,ddname,dev,ser,xxx, dsname,cat`

What happened?

Due to the lack of virtual storage, an abend occurs. In this case, a symptom dump may be included.

What to do for recovery?

Because the data set processing was interrupted (abended) in apparently unknown circumstances, there are two cases:

- VSAM data set is being accessed by a subsystem as CICS, then CICS was doing the synchpoint, journaling, and is able to recover your data by rolling it back.
- VSAM data set being accessed by your program; then you should correct the virtual storage problem and re-run the program (if possible) or restore the backup and re-run the job.

Sometimes, the message is not the result of an abend. It can be an alert as with IEC161I, where BLDVRP macro indicates that there was not enough virtual storage to satisfy the request done by System Management Buffering (SMB). SMB gets the available storage, and processing goes on.

What to do to avoid future problems?

Initially, read 2.6.12, “Region size” on page 61. If possible, increase your region below or above, or decrease the common area below 16 MB, or force your software to be in R31 mode.

Determine if the VSAM buffers and their control blocks are below or above the 16 MB line. If below, read “Locating VSAM buffers above 16 MB” on page 81 to learn how to move them above with integrity.

3.2.2 Initial loading problems

The following messages may indicate initial load problems:

- `IDC3308I ** DUPLICATE RECORD xxx`

  The output data set of a REPRO command already contains a record with the same key or record number.

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 8 (I/O): You attempted to store a record with a duplicate key, or there is a duplicate record for an alternate index with the unique key option
  - 12 (I/O): You attempted to store a record out of ascending key sequence in skip-sequential mode; record had a duplicate key; for skip-sequential processing, your GET, PUT, and POINT requests are not referencing records in ascending sequence; or, for skip-sequential retrieval, the key requested is lower than the previous key requested. For shared resources, the buffer pool is full.
  - 116 (I/O): During initial data set loading (that is, when records are being stored in the data set the first time it is opened), GET, POINT, ERASE, direct PUT, and skip-sequential PUT with OPTCD=UPD are not allowed. During initial data set loading, for initial loading of a relative record data set, the request was other than a PUT insert.

What happened?

These messages point to problems with initial load or mass insertion (also called skip sequential) of a VSAM cluster.

Initial load of your data set can be done by IDCAMS REPRO or by a program of yours. Refer to 2.6.11, “Initial load option” on page 59, for more information.

When loading a VSAM KSDS data set, the logical records must be sorted in a key sequenced order. No out-of-sequence or duplicated keys are allowed. Refer to the above messages for a more detailed explanation about which of these requirements were not fulfilled.

If the *duplicated keys* message applies to the initial load of an alternate index (AIX) cluster, remember that for AIX is possible to have duplicated keys, but in this case you should not use the UNIQUE parameter, which would definitely cause this error.

Mass insertion resembles initial load in the sense that all the added logical records also need to be sorted by a key field. The difference is that in mass insertion, we are loading sequentially logical records to a data set which already has previous data.

What to do for recovery?

Here, there is not a need for recovery. Your data is saved in the input file. It is a matter of sorting and re-running the program.

What to do to avoid future problems?

After correcting the error, introduce a procedure in production to avoid having the same error again.

3.2.3 Mismatch between catalog and data set

The following error messages may indicate a mismatch between catalog and data set information:

- `IDC3351I ** VSAM OPEN RETURN CODE IS 108`

  108: Attention message: the time stamps of a data component and an index component do not match. This indicates that either the data or the index has been updated separately from the other. Check for possible duplicate VVRs.
- `IDC3351I DATA SET IS ALREADY OPEN FOR OUTPUT OR WAS NOT CLOSED CORRECTLY`
- The data set is already OPEN for output by a user on another system, or was not previously closed.
- `IDC11709I DATA HIGH-USED RBA IS GREATER THAN HIGH-ALLOCATED RBA`

  The data component high-used relative byte address is greater than the high-allocated relative byte address. Supportive messages display pertinent data, and processing continues.
- `IDC11712I DATA HIGH-ALLOCATED RBA IS NOT A MULTIPLE OF CI SIZE`

  The high-allocated relative byte address is not an integral multiple of the control interval size.

- `IDC11727I INDEX HIGH-USED RBA IS GREATER THAN HIGH-ALLOCATED RBA`

  The index component high-used relative byte address is greater than the high-allocated relative byte address.
- `IDC3350I synad[SYNAD] NO RECORD FOUND from VSAM`

What happened?

The data set may be intact, but the catalog information describing the data set mismatch problems. It sometimes results in an open failure.

The most common discrepancies between the catalog and cluster are these:

- There were different time stamps between index and data components.
- HURBA and HARBA not correctly updated, mainly caused by an abend, without a normal close.
- Open-for-output bit on for a closed cluster; mainly caused by an abend, without a normal close, or other task accessing from other system. Refer to 3.4.5, “Abend task scenario” on page 171, for more information.
- RBAs fields in the VVR do not match the data set attributes, for example:
  - If the RBA of the high-level index CI is corrupted, you are not be able to perform direct requests against the data set.
  - If the RBA of the sequence set index CI is corrupted, you are not able to perform sequential access.

  These last two discrepancies are not covered here; refer to 3.2.6, “Structural damage” on page 150.

A LISTCAT output (or CSI report) can help with the documentation for problem determination.

What to do for recovery?

IDCAMS VERIFY is a requirement in this type of problem. In this case, VERIFY can correct HURBA and verify the open-for-output bit.

- Different time stamps; Open does not abend your task, so continuation or abend of the application depends on the program that issues the OPEN. In general, it will keep processing, but another problem is likely to occur.

  We suggest you run VERIFY (to be sure and document the mismatch) and then run EXAMINE to guarantee no structural damage exists for KSDS and VRRDS, with a test for the index option only (INDEXTEST).

  Completion of EXAMINE without error proves that there are no structural damages. If the index component shows damage at this point, it must be restored before further use. Refer to 3.4.4, “Broken Index scenario” on page 169, to get some information on doing that. Note that EXAMINE may provide messages containing only informational data that may not require restoring the cluster.

- HURBA and HARBA not correctly updated.

  If the HURBA is not updated, when the data set is subsequently opened and the user's program attempts to process records beyond end-of-data or end-of-key range, a read operation results in a “no record found” error, and a write operation might write records over previously written records. To avoid this, you can use the VERIFY command which corrects the catalog information.

- Open-for-output bit on for a closed cluster.

  At next OPEN, VSAM implicitly issues a VERIFY command, when it detects an open-for-output indicator on and issues an informational message (maybe the one that you are seeing) stating whether the VERIFY command is successful.

  If a subsequent OPEN is issued for output, VSAM turns off the open-for-output indicator at successful CLOSE. If the data set is opened for input, however, the open-for-output indicator is left on.

3.2.4 Hardware errors

Messages that indicate hardware errors include:

- `IOS000I dev,chp,err,cmd,stat, dcbctfd,ser,mbe,eod, jobname,sens text`

  The system found an uncorrectable I/O error in device error recovery. Text is one of the following types:

  – Channel interface, or protocol error
  – Device has exceeded long busy timeout
  – Permanent error — volume fenced
  – Permanent error — device reported unknown message code = cde
  – Channel control, data, chaining, program, protection, interface check
  – Unable to obtain sense data from the device

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`

  – 184 (Open): An uncorrectable I/O error occurred while VSAM was completing outstanding I/O requests.

  – 246 (Close): The compression management services (CMS) close function failed.

  – 184 (Open): An uncorrectable I/O error occurred while VSAM was completing an I/O request.

- 245 (I/O): A severe error was detected by the compression management services (CMS) during compression processing).
- 246 (I/O): A severe error was detected by the compression management services (CMS) during decompression processing.
- 250 (I/O): A valid dictionary token does not exist for the compressed data set. The data record cannot be decompressed.

What happened?

Hardware I/O errors usually mean that the I/O hardware (channel, controller, device) had a problem executing that I/O. For compressed VSAM data sets, you may have hardware problems with the CPU compression assist function.

The first thing to do is break down the message to get more details on the error. An IDCAMS LISTCAT (if possible) of the data set is also helpful to give the attributes of the file in the logical 3390/3380, such as: the physical record size, the device type and the CCHH of all extents. LOGREC output is also valuable. A GTF CCW trace may be necessary to run DIAGNOSE on the problem, however, for such tools, you may need to recreate the situation.

VERIFY IGGCSIVS is a program from SYS1.SAMPLIB accessing Catalog Search Interface (CSI) which produces a list of data set names defined in a given catalog that reside on a specific volume. Such a list might be helpful in a recovery situation affecting that volume.

When accessing with VSAM macros, a VSAM data set where a physical error was detected, the register 15 comes with return code equal to 12.

What to do for recovery?

In the case of a media error, do not use ICKDFS to run an Analyze and Inspect function. The characteristics of the physical devices that make up the RAID devices family do not allow the use of the ICKDSF commands that perform installation, media maintenance and problem determination functions, such as Install, Analyze, and Inspect.

If the problem happens with the compress assist feature, run the program again, switching off data compression in the data class.

What to do to avoid future problems?

If the error is in the DASD controller, keep a log of such type of occurrences to force a better quality of the manufacturer or change to other. Another solution (for some media problems) is to implement RAID-1 dual copy (in the same controller) or remote copy (in two controllers), mainly for your most critical data, such as logs.

3.2.5 Bad data or bad channel program

The following message may indicate bad data or bad channel program:

```
IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
```

- 140 (Open): The catalog indicates this data set has an incorrect physical record size.
- 16 (I/O): Record not found.
- 88 (EOV): A previous extend error has occurred during EOV processing of the data set.

What happened?

This problem may be pervasive, but in general, it is usually caused by two major reasons:

- Duplicate VVRs
- A bad channel, causing data overlays and corrupting indexes

The existence of damaged or duplicate VVRs on a volume may cause data sets to be overlaid with data from other data sets.

VSAM volume record (VVR) is a logical record within VSAM volume data set (VVDS). VVDS is a data set that describes the dynamic characteristics of VSAM and system-managed data sets residing on a given DASD volume. Together with the BCS, it is a part of an integrated catalog facility.

There are many things that can cause the channel program to be bad. The most common causes of a bad channel program is that the data that describes the data set is bad. If a bad VVR is picked up at Open time, VSAM may try to access cylinder and tracks that do not belong to the data set getting various I/O errors.

If another program overlays the VSAM data set, this can cause the channel program to fail at that spot where the other data exists. For instance, if the CI size of the VSAM file that is broken is 4 KB, the channel program is built to read records of that size. If another program has overlaid the file with records of, say, 16 KB size, the channel programs for the record size of 4 KB fails on all cylinder/tracks/heads that do not have this record size. This situation is usually referred to as bad data.

Theoretically, system code problems can cause VSAM to build channel program incorrectly, or in some cases they may be built correctly, but they are getting modified incorrectly and redriven by ERP or even third party products. Luckily, these reasons are not too common, even with new device types.

In regard to corrupted indexes, refer to 3.2.6, “Structural damage” on page 150.

With problems like these, it is important to get as much information about the data set as you can before the customer restores it. We suggest using:

- LISTCAT or CSI.
- DFSMSdss (PRINT command) or Ditto to print the 3390/3380 logical cylinder/track/head that the I/O error is occurring. It can indicate whether there is any data at all on the track, or if the data that is there belongs to a different data set.
- SMF records, mainly the type 6x connected to catalog and VSAM data sets.

After the data set has been recovered, the only “history” that can be EXAMINEed is the SMF records. If the problem “clears up” after the data set is closed, but without the data set being recovered, you might suspect a problem with an internal control block being overlaid, rather than something on DASD.

What to do for recovery?

This overlay is a hard failure and the data set has to be manually restored from a backup. Often, in this case, the bad data itself gives a clue as to what data set or application has caused the overlay. Trying to avoid the restore for a bad KSDS data component, you may try to skip past the bad data records, and recover only those records that can be properly read.

A DIAGNOSE command even after the data set has been recovered can check for this problem (since only a DELETE VVR can get rid of an orphan after one occurs). Luckily, many enhancements have been introduced to Catalog and Open processes in the last few years to check for duplicate VVRs at OPEN time, so this should be less of a problem.

What to do to avoid future problems?

Experience has shown that the majority of such errors are caused by improper sharing. If you are sharing your VSAM data set, refer to 3.2.7, “Improper sharing” on page 153.

Because such types of errors may also be caused by system errors, you may want to investigate the possibility of APARs and PTFs related to the problem.

3.2.6 Structural damage

The following message may indicate structural damage:

```
IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
```

- 128 (Close): Index search horizontal chain pointer loop encountered.
- 190 (Open): An incorrect high-allocated RBA was found in the catalog entry for this data set. The catalog entry is bad and will have to be restored.

- 76 (Open): Attention message: The interrupt recognition flag (IRF) was detected for a data set opened for input processing. This indicates that DELETE processing was interrupted.
- 4 (I/O): End of data set encountered (during sequential retrieval), or the search argument is greater than the high key of the data set. Either no EODAD routine is provided, or one is provided and it returned to VSAM and the processing program issued another GET.
- 32 (I/O): An RBA specified that does not give the address of any data record in the data set
- 128 (I/O)): A loop exists in the index horizontal pointer chain during index search processing.
- 144 (I/O): Incorrect pointer (no associated base record) in an alternate index.
- 156 (I/O): An addressed GET UPD request failed because the control interval flag was on, or an incorrect control interval was detected during keyed processing. In the latter case, the control interval is incorrect for one of the following reasons:
  - A key is not greater than the previous key.
  - A key is not in the current control interval.
  - A spanned record RDF is present.
  - A free space pointer is incorrect.
  - The number of records does not match a group RDF record count.
  - A record definition field is incorrect.
  - An index CI format is incorrect (logical I/O error)

What happened to cause this problem?

KSDS or VRRDS VSAM data set organizations can “break” in more ways than other data sets, because they have an index component with logical pointers to other data and index CIs. If these pointers become corrupted, data can be lost or duplicated.

Also, much structural information about such data sets is located in the ICF catalog. For example, two RBA fields in the VVR are very important in accessing a KSDS data set, for example:

- If the RBA of the high-level index CI is corrupted, you are not be able to perform direct requests against the data set.
- If the RBA of the sequence set index CI is corrupted, you are not able to perform sequential access.

Refer to 3.8, “IDCAMS LISTCAT output fields” on page 187, for more information about RBA data in the VVDS catalog.

Then, if these fields are corrupted, errors may prevent you from accessing the data even though the data is intact. Also, it is possible that the index CI's horizontal chain is destroyed, inhibiting the access to the data. One common way these fields can get corrupted is due to overlays of the AMDSB control block while the data set was open, which then get updated back to the VVDS at close time. Another way is through improper sharing of the data set during initial load mode processing.

Because these fields are stored in the ″Statistics Block″ (AMDSB), jobs that only opened the data set for input still update this information at close time and for that reason do not dismiss any possibilities just because the job was not updating the file.

SMF records are very helpful when diagnosing VVR damage. By investigating the SMF records (for example, type 60, 62 and 64) from all systems, improper access of the data set can be identified as well as the time frame of the corruption.

What to do for recovery?

When dealing with KSDS/VRRDS, it is crucial that EXAMINE is run on the data set as part of the diagnosis. Also, the sooner the EXAMINE is run after the data set is broken, the better, since some types of damage can actually cause more breakage until the data set is so badly broken it is impossible to tell what actually happened first. Remember that the EXAMINE command provides details about the nature of data set damage.

Sometimes, the IDCAMS DIAGNOSE command can be used to check the data set for structural error in the catalog itself.

When losing the index in a KSDS/VRRDS, one possible recovery path is to read the data (in physical sequential mode) via its data component. Here you may use an assembler program (MACRF=ADR in ACB and OPTCD=ADR in RPL), or by IDCAMS Repro. Then, classify by the key and use an IDCAMS Define and REPRO to recreate the KSDS. Refer to 3.4.4, “Broken Index scenario” on page 169 for more details.

What to do to avoid future problems?

Experience has shown that some of such errors are caused by improper sharing. If you are sharing your VSAM data set, refer to 3.2.7, “Improper sharing” on page 153.

Because such types of errors maybe caused by system errors, you may want to investigate the possibility of APARs and PTFs related to the problem.

3.2.7 Improper sharing

The following message may indicate improper sharing:

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 16 (I/O): Record not found.
  - 20 (I/O): Record already held in exclusive control by another requester.
  - 28 (I/O): Data set cannot be extended because VSAM cannot allocate additional DASD space.
  - 88 (Open): A previous extend error has occurred during EOV processing of the data set.
  - 96 (Open): Attention message: an unusable data set was opened for input.
  - 116 (Open): Attention message: the data set was not properly closed or was not opened. If the data set was not properly closed, then data may be lost if processing continues. The data set was not properly closed.
  - 236 (I/O): Validity check error for SHAREOPTIONS 3 or 4.
- `IDC11705I INDEX RECORD CONTAINS DUPLICATE INDEX POINTERS pointer-value`

What happened?

Improper sharing is one of the most common causes of broken data sets. This section covers some of the things to check for, to make sure the share options are proper. Refer to “VSAM SHAREOPTIONS” on page 223, for information on share options. Here are some causes of sharing problems:

- Sharing a data set across regions (cross-region) or across systems (cross-system) without using proper enqueuing procedures to protect data set integrity. For shareoptions shr(3 3) shr(4 3) shr(3 4) see related information in OY36328.
- Sharing a data set across systems — even using the appropriated share option — but without propagating ENQ name SYSVSAM around the GRS ring. This is the *most* common user error. It results in duplicate index pointers in the high level index records. See message IDC11705I.
- When a data set is defined using the model parameter, and the original data set has SPEED as an attribute of the index, data set damage can occur. VSAM does not support speed as an attribute for the index (speed is only supported for the data).
- Another area related to broken data sets that is specific to CBUF processing is the VSI control block. Every time a data set is opened on a system for CBUF processing, a VSI is built for the data set and added to the VSI chain. This control block is then updated by the user to communicate information

from one region to another. If the user does this improperly, a broken data set can result. Refer to 4.3.6, “Protecting VSAM data set through DISP parameter” on page 227, for more information.

Documentation is of paramount importance to address sharing problems. The necessary documents should be obtained at each system (or address space) accessing the troubled data set. The major document needed here is the IDCAMS LISTC or CSI report. List the catalog entry for the affected data set to show the allocation and RBA data (beware OY61232).

The SMF 62 and 64 records can help you determine if the users have the data set open for output from different applications at the same time. A common user error in this area is applications or ISV products that were not intended to be run from multiple systems (and so they have no logic to serialize updates), but the customer is using them in this manner.

What to do for recovery?

Here is a list of recovery options:

Use the Access Method Services VERIFY command to attempt to close the data set properly. In a cross-system shared DASD environment, the ACBERFLG = 116 may mean the data set was not properly closed. The warning "data set not properly closed" may indicate an error in a VSAM data set. It may mean one of the following:

- The "end of the data set" indicators (data set high-used RBA/CI) and so on) may be invalid.
- There may be missing records.
- There may be duplicate records.
- The data set statistics fields in the catalog may be invalid.

If VSAM OPEN cannot successfully do a VERIFY then a user cannot do a VERIFY either.

- Execute the IDCAMS EXAMINE command on the data set. Completion of EXAMINE without error proves that damage did not occur in a previous job. If the data set shows damage at this point, it must be restored before further use.
- Proceed with the application job execution.
- Execute IDCAMS EXAMINE on the data set when the job completes.
- If damage to the cluster has occurred, run EXAMINE on SMF records from all systems which do have the ability to access the DASD volume. If shared access to the data set has occurred, correct or eliminate the contention for the data set.

What to do to avoid future problems?

You should issue the VERIFY command every time you open a VSAM cluster that is shared across systems or address spaces. Read carefully 4.3, "Sharing VSAM data sets" on page 212 and use all the serializing techniques to avoid the structural damage and the data integrity of your data set.

3.2.8 Mismatch between catalog and VTOC

In this book we do not cover much catalog recovery; however, some of the catalog problems are mentioned as the ones associated with the IDC3009I message. Refer to 3.7, "IDC3009I message" on page 181, where a more detailed description of the return and reason codes from this message are presented.

```
IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
```

– 132 (Open): One of the following errors occurred:
  - Unable to read JFCB or Scheduler Work Block
  - Unable to connect to redo log during DFSMStvs open
  - The forward recovery log associated with the data set was altered.
  - DFSMStvs failed to write a record to the forward recovery log.
  - Caller TASK cancelled
  - The data set is in a forward recovery required state.

- `IDC3009I VSAM CATALOG RETURN CODE IS return-code – REASON CODE IS IGGOCLaa – reason-code`

3.2.9 VSAM does not produce expected output

Incorrect output failures can be identified by the following results:

- Expected output is missing.
- Output is different than expected.
- Output should not have been generated.
- System indicates damage to the VTOC or VTOC index.
- ISMF panel information or flow is erroneous.

Incorrect output can be the result of a previous failure and can often be difficult to analyze because the component affected might not be the one that caused the problem. Review previous messages, abends, console logs, or other system responses. They could indicate the source of the failure.

Accumulating as much information as possible

It can help you isolate or resolve your problem, and the IBM Support Center will request it if trap or trace information is needed, such as the following.

- When was the problem first noticed?

- How was the problem identified (good output versus bad output)?
- Were any system changes or maintenance recently applied? For example, a new device, software product, APAR, or PTF?
- Does the problem occur with a specific data set, device, time of day, and so forth?
- Does the problem occur in batch or TSO mode?
- Is the problem solid or intermittent?
- Can the problem be re-created?
- EXAMINE the system and console logs for failure-related abends, messages, or return codes. A damaged VSAM data set can also cause incorrect output.
- Add any failure-related return codes to the keyword string, exactly as the system presents them. You can also add the abend or message type-of-failure keywords to the incorrect output keyword string to define the symptoms more closely:
- Determine whether failure-related record management return codes and reason codes exist.

  VSAM provides return codes in register 15 and reason codes in either the access method control block (ACB) or the request parameter list (RPL). Reason codes in the ACB indicate VSAM open or close errors. Reason codes in the RPL indicate VSAM record management error indications returned to the caller of record management. Reason codes returned to the caller of record management in the RPL indicate VSAM record management errors.
- Determine whether you have a damaged VSAM data set.

  Some incorrect output failures involve a damaged VSAM data set. To determine whether you have a damaged data set, use the IDCAMS EXAMINE command as described in the chapter on functional command format in DFSMS/MVS Access Method Services for ICF and the chapter on checking a VSAM key-sequenced data set cluster for structural errors in DFSMS/MVS Using Data Sets. The EXAMINE command provides details about the nature of data set damage. If these service aids indicate that the data set is not damaged, inform the IBM Support Center if you call for assistance. If they indicate that the data set is damaged, keep a copy of the output for possible use by the IBM Support Center. Be prepared to describe the type of data set damage. You should attempt to recover the data set and rerun the failing job to determine whether the problem is resolved.

3.2.10 VSAM RLS problems

This is described in 5.5, “RLS problem determination and recovery” on page 266

3.2.11 VSAM and DFSMStvs considerations

Refer to Chapter 6, “DFSMStvs” on page 323.

3.2.12 OEM problems

Overlay of control blocks, abends, loops, deadlocks and performance problems can occur due to errors in the use of VSAM data sets. A large number of problems have been reported by many IBM products and non-IBM products that use VSAM data sets. These problems are described in a number of information APARs in IBMLINK. At the time of writing this book, 270 problems have been documented in these APARs as follows.

- II10001 - Problems 1-60
- II11013 - Problems 61-104
- II11513 - Problems 105-157
- II12140 - Problems 158-200
- II12615 - Problems 201-238
- II13278 - Problems 239-270

3.2.13 Enqueue issues

OPEN message IEC161I 052-084 is a common informational message. Most often it simply means that another job already has the dataset open when this job is trying to open it. As it usually indicates a scheduling problem rather than a system problem. In this section we provide information on how to determine what jobs are in contention for the same dataset.

VSAM OPEN processing determines this condition by checking the GRS data. Depending on the shareoptions (SHR) of the dataset, and the attributes of the OPEN, VSAM will enqueue against major name SYSVSAM with a minor name of the dataset name/catalog name plus other information. Thus, GRS is the mechanism OPEN processing uses to insure serialization of the OPEN process itself as well as the shareoptions of the file.

When VSAM issues an ENQUEUE for a SYSVSAM resource, VSAM will add an ENQRNIND indicator to the wanted resource name:

- ENQRNIND = B = BUSY
- ENQRNIND = I = INPUT
- ENQRNIND = N = Non RLS Open
- ENQRNIND = O = OUTPUT
- ENQRNIND = R = RESERVE
- ENQRNIND = S = Sphere
- ENQRNIND = C = CLOSE
- ENQRNIND = R = RLS Read

- ENQRNIND = W = RLS Write

The ENQ for BUSY will be held throughout the period of the initial operation being performed, that is, OPEN, EOV, CLOSE, TCLOSE or CHECKPOINT RESTART. When the operation is complete or is failed, then, the 'B' resource will be dequeued. Example:

- ENQ DATASET/CAT/B (OPEN) EXCLUSIVE
- ENQ DATASET/CAT/O
- DEQ DATASET/CAT/B (process data)
- ENQ DATASET/CAT/B (CLOSE) EXCLUSIVE
- DEQ DATASET/CAT/O
- DEQ DATASET/CAT/B

The first place to look to determine which job is enqueued on the dataset is GRS data. You can view this data with the DISPLAY GRS command or through a monitor program. Look at both the global and local queues under the major name of SYSVSAM for the dataset that is getting the OPEN failure. Unfortunately, the GRS data is transient and may well change before you can find the job that was responsible for the message, but, do try the GRS command. Issue this GRS command from all systems that can share the DASD:

```
D GRS,RES=(SYSVSAM,*)
```

Note: If you are using IPCS to view GRSDATA remember that SYSVSAM will be there *twice*, once for Local System Resources and a second time for Global Systems Resources.

The next place to look is the console log. You may be able to determine if there were any backups/copies/dumps being taken at the time or even if there were unexpected batch jobs running at the time.

Another place to look for information is the SMF type 62 (SMF62) and SMF64 records. This will show what jobs were accessing the dataset at the time of the error. This also may not be conclusive since some access of the dataset will not produce SMF records (that is, Media Manager, DB2, DFSMS/dss).

Ensure the availability of the resource by means of JCL DD statements. This means, check your JCL and ensure that you have requested the proper disposition, that is, DISP=SHR.

Use the AMS command ALTER, to reset the Update Inhibit indicator in the data set's catalog record then, rerun the job. This means, run LISTC against the failed cluster, check for an attribute of INH-UPDATE. If found ON, then this dataset is in READONLY mode and will fail an OPEN for UPDATE request. Use the AMS command 'ALTER UNINHIBIT' to place the file in READ/WRITE mode, then, rerun the job. Example 3-1 shows a sample job for Cluster DFP1.JIMONE.KSDS (on a D/T3390 with volser=339000).

Example 3-1 Sample AMS JCL

```
//SYSIN DD *
      PRINT  INDATASET(SYS1.VVDS.V339000) DUMP
      ALTER  DFP1.JIMONE.KSDS.DATA  UNINHIBIT
      ALTER  DFP1.JIMONE.KSDS.INDEX UNINHIBIT
      VERIFY DATASET(DFP1.JIMONE.KSDS)
      LISTC  ENT(DFP1.JIMONE.KSDS) ALL
 /*
```

In a VSAM catalog, INHIBIT UPDATE bit is in the BCS DINC record. In an ICF catalog, INHIBIT UPDATE bit is in the VVDS VVR record.

If you are NOT able to determine what job is the cause of the IEC161I message, open a problem record with the Support Center and include your RMID of IDA0192B. You will be given a SLIP to capture a dump (which includes the GRSQ data) at the time the message is issued.

The SLIP will be set in CSECT IDA0192B of load module IDA0192A upon entry to a procedure named, PROBDT2B. IBM will need to know the PTF level of IDA0192B in order to resolve the offset of xxx in the SLIP. The user will need an AMBLIST of IDA0192A. A sample of the slip is shown in Example 3-2.

Example 3-2 SLIP to determine the job causing IEC16I

```
SLIP SET,IF,A=SYNCSVCD,L=(IDA0192A,xxx),DATA=(13R?+F8,EQ,34),
      SDATA=(ALLNUC,CSA,GRSQ,LPA,LSQA,RGN,SQA,SUM,TRT),END
```

IDA0192B is an offset into load module IDA0192A and PROBDT2B is an offset into IDA0192B. xxx is the sum of these 2 offsets. The value of xxx and values specified in the SLIP DATA parms may change with different releases and maintenance levels of CSECT IDA0192B. Be sure to check with IBM Support Personnel to ensure an accurate SLIP for your level.

3.2.14 Migration issues

Please refer to the relevant migration manuals to be aware of changes you need to make when you migrate to a new release of z/OS.

Enhancements to calculation of default CISIZE is an example. This is described in 4.2, “New Index CI size calculation algorithm” on page 208.

3.2.15 Performance considerations

In a sense a performance problem may appear as a lack of availability. Refer to Chapter 2, “Performance” on page 43, to see several aspects of VSAM performance and specifically to 2.9, “Performance monitors” on page 136, to learn about performance monitors.

3.2.16 Deadlocks

Deadlocks are a performance and also a problem determination subject.

Here we cover define a deadlock, how to prevent them, how to detect a deadlock and what to do when we have one.

What is a deadlock?

Locks are used in VSAM by the DFSMS lock manager when the same control block structures are shared by strings in task programs. In such a case, Share Options do not apply. Refer to “Sharing data in a single VSAM control block structure” on page 218. If the VSAM data set is processed in non-RLS mode there is one lock per CI, when in RLS there is a lock per each logical record.

Then, contention for VSAM CIs locks (or logical records locks) can lead to deadlocks, as such: A delays B in one lock and B delays A in other lock.

How to prevent a deadlock

The major rule is try to avoid to lock more than one logical records concurrently. Also a key recommendation is to avoid consistent reads (CR). If the RLS exploiter cannot follow such rules, it could create a hierarchy of locks. Then, when more than one lock is required concurrently, they need to be required in a pre-determined sequence.

How to detect a deadlock

The DFSMS lock manager provides a deadlock detection routine, the frequency with which it runs is determined by the installation. This routine considers a string task program in a deadlock situation, if it is waiting for a lock for more than an installation specified amount of time. There are two deadlock detection routines:

- Deadlocks within a system
- Deadlocks between systems

The frequencies of the deadlock detection routines are specified in GDSMSxx parameter of Sys1.Parmlib.

In a Sysplex, the first system that is initialized with an IGDSMSxx member having a valid DEADLOCK_DETECTION specification determines this keyword for the other systems in the Sysplex. You can change this value through the SETSMS or SET SMS commands.

This keyword specifies the intervals for running local and global deadlock detection routine.

The first subparameter nnnn is the local deadlock detection cycle and specifies the interval in seconds for detecting deadlocks within a system.

The second subparameter is the global deadlock detection cycle and specifies the interval for detecting deadlocks between systems. This value is specified as the number of local detection cycles that occur before global deadlock detection is initiated.

To determine if a string task program is in a dead lock situation DFSMS lock manager compares the RLSTMOUT keyword with the amount of time a string task program is waiting for a lock.

`RLSTMOUT({nnnn|0})`

Specifies the maximum time, in seconds, that a VSAM RLS request must wait for a required lock before the request is assumed to be in deadlock.

You can specify a value between 0 to 9999 (in seconds). A value of 0 means that the VSAM RLS request has no time out value and the request waits for as long as necessary to obtain the required lock.

RLSTMOUT can be specified only once in a sysplex and applies across all systems in the sysplex.

What to do when a deadlock is detected

When a lock request is found in a deadlock, VSAM rejects the request in wait.

This results in the VSAM request completing with a deadlock error response. Your applications must be prepared to accept locking error return codes that may be returned on GET or POINT NRI requests. However, normally such errors do not occur.

3.2.17 Beware of some VSAM restrictions

Keep this important restriction in mind when using the REPRO command.

REPRO to empty VSAM data set

If you try to repro into an empty VSAM data set, you will get an error IDC3351I ** VSAM OPEN RETURN CODE =160. This is a permanent restriction of VSAM. One work-around is to "prime" the dataset by adding and deleting one record.

This will increment the HI-U-RBA to a value other than zero, and allow the REPRO to proceed.

3.3 What documentation to collect

First familiarize yourself with getting the required documentation, such as logs, dumps, traces, and messages associated with errors. There is an information APAR II12927 in IBMLINK that tells you what information you need to collect for general VSAM problems.

Note: It is generally a good idea to include an AMBLIST of the VSAM load modules IDA019L1 and IDA0192A when sending a VSAM problem to IBM level2 support.

3.3.1 Catalog performance problems

Here is the information to collect and questions to consider.

- Provide a clear and adequate description
- What is not performing properly?
- What is the basis for the performance expectation?
- What changes appear to trigger the performance change?
- Maintenance, release, application?
- Dump of CAS (and associated user ASIDs)
- Recovery actions (if Catalog is involved)
- Performance report
- Comparison report (what was performing in the last release?)

For details see information APAR II10752.

3.3.2 VSAM RLS problems

You would need to collect dumps of SMSVSAM and other address spaces depending on the problem. You may require the following documentation:

- A dump of SMSVSAM ASIDs on all systems at the same time. See Example 3-3.
- A dump SMSVSAM and XCF on all the systems in the sysplex See Example 3-4.
- A dump of the SMSVSAM ASID, SMSVSAM data spaces and CICS regions involved. See Example 3-5.

Example 3-3 Dump SMSVSAM ASIDs on all systems

```
DUMP COMM=(Some meaningful dump title),
  R nn,JOBNAME=(SMSVSAM),CONT
  R nn,SDATA=(GRSQ,RGN,ALLNUC,LPA,LSQA,CSA,PSA,SQA,SUM,SWA,TRT),
  R nn,DSPNAME=('SMSVSAM',*),
  R nn,REMOTE=(SYSLIST=*('SMSVSAM'),DSPNAME,SDATA),END
```

Example 3-4 Dump SMSVSAM and XCF on all systems

```
DUMP COMM=(SMSVSAM and XCF)
  R XX,JOBNAME=(SMSVSAM,XCFAS),DSPNAME=('XCFAS'.*,'SMSVSAM'.*),CONT
  R YY,SDATA=(GRSQ,RGN,ALLNUC,LPA,LSQA,CSA,PSA,SQA,SUM,SWA,TRT), CONT
  R ZZ,REMOTE=(SYSLIST=(*,('XCFAS','SMSVSAM',DSPNAME,SDATA)))
```

Example 3-5 Dump of SMSVSAM and CICS

```
DUMP COMM=(CICS & SMSVSAM HANG)
  R nn,JOBNAME=(SMSVSAM, CICS1, CICS2, CICS3, ETC),
  R nn,DSPNAME=('SMSVSAM'.*),
  R nn,SDATA=(PSA,NUC,SQA,LSQA,SUM,RGN,GRSQ,LPA,TRT,CSA),CONT
  R nn,REMOTE=(SYSLIST=*('SMSVSAM'),DSPNAME,SDATA),END
  where CICS1, CICS2, CICS3, ETC are the names of the CICS regions.
```

3.3.3 IDCAMS problems

The following are documentation requirements for IDCAMS problems:

- Syslog (including all messages, JCL and commands)
- VERIFY output
- LISTCAT ALL output
- EXAMINE (ITEST and DTEST)
- DIAGNOSE

3.3.4 Broken VSAM data set

You will need to collect the following documentation:

- SYSLOG/JOBLOG (including all messages, JCL and commands)
- EXAMINE (both ITEST and DTEST) output
- LISTCAT ALL output
- DIAGNOSE output

Please see information APAR II08859 for more details.

3.3.5 Broken catalog

Collect the following documentation for this error condition:

- LISTCAT ENT(catname)ALL CATALOG(catname)
- DIAGNOSE vvds
- DIAGNOSE bcs
- DIAGNOSE COMPARE vvds to bcs
- DIAGNOSE COMPARE bcs to vvds
- EXAMINE (both ITEST and DTEST)
- Any error messages from joblog

3.3.6 How to obtain VSAM record management trace?

Sometimes you may need to collect VSAM record management traces. This section describes how to collect VSAM record management traces.

1. GTF must be activated before you allocate the data set that is being traced. Start GTF with TRACE=USRP,USR=FF5, END
2. The SYS1.TRACE data set should be approximately 100 cylinders, or larger if the space is available. If, however, the activity on this file will be quite heavy there is a possibility of "lost trace entries." This means that the amount of trace records coming in is more than GTF can keep up with. Splitting the SYS1.TRACE data set into multiple data sets usually prevents this. See the informational APAR II10072 for further information. An example has also been provided following #5 (below).
3. Place an AMP=('TRACE=(subparameters)') on the DD statement of the data set that you want to trace. VSAM Level 2 will provide you with the coding of the AMP parameter.

Note: If the data set being traced is dynamically allocated a usermode must be provided from L2 and applied to your system. For example:

```
//ddname DD ...,
//   AMP=('TRACE=(PARM1=64C4A1802000,HOOK=(0,1,3,9,10,11,25,26))')
```

4. Restart applicable CICS region(s), run batch jobs, and so on. It is important that GTF be started BEFORE any data set allocation occurs, otherwise, the control blocks for tracing will not be built and no tracing will occur.
5. Recreate the problem.
6. Stop trace as soon as job terminates.An example of JCL that can be used to send the GTF output to more than one data set is shown in Example 3-6. This is often necessary on high output traces:

Example 3-6 Sampler JCL for GTF trace

```
//GTFABC PROC MEMBER=GTFPARM
//IEFPROC EXEC PGM=AHLGTF,REGION=2880K,TIME=1440,
// PARM=('MODE=EXT,DEBUG=NO,TIME=NO')
//IEFRDER DD DSNAME=SYS1.GTFTRC,UNIT=SYSDA,
// SPACE=(4096,20),DISP=(NEW,KEEP)
//SYSLIB DD DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
//GTFOUT1 DD DSNAME=SYS1.TRACE1,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT2 DD DSNAME=SYS1.TRACE2,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT3 DD DSNAME=SYS1.TRACE3,UNIT=SYSDA,DISP=(NEW,KEEP)
```

GTF would be started with: `S GTFABC`.

SYS1.PARMLIB(GTFPARM) would be the parmlib member that contains the trace parms to be used, `TRACE=USRP,USR=(FF5),END`.

3.4 How to recover a damaged VSAM data set

IDCAMS has three important commands used to recover VSAM clusters and catalogs. They are: EXAMINE, DIAGNOSE, and VERIFY.

3.4.1 EXAMINE command

EXAMINE is an IDCAMS command that allows the user to analyze and collect information on the structural consistency of KSDS data set clusters and of a VVRDS data set cluster. In addition, EXAMINE can analyze and report on the structural integrity of the basic catalog structure (BCS) of an ICF catalog. This function cannot share a cluster opened for output or update by other task.

The EXAMINE function executes two possible tests:

- Test the Index portion (INDEXTEST), the default

  Evaluates the full index component of the KSDS/VRRDS cluster by cross-checking vertical and horizontal pointers contained within the index control intervals, and by performing analysis of the index information. Usually is a medium to small resource consuming task.

- Test the Data portion (DATATEST)

  Evaluates the index sequence set and data component of the key-sequenced data set cluster by sequentially reading all data control intervals, including free space control intervals. Tests are then carried out to ensure record and control interval integrity, free space conditions, spanned record update capacity, and the integrity of various internal VSAM pointers contained within the control interval. Usually this is a long resource consuming task.

You can also limit the number of error messages generated (ERRORLIMIT) by EXAMINE. Refer to “Broken data set” on page 179 for information about what to do whether EXAMINE is reporting errors in the Index or Data portion. Refer also to “IDCAMS EXAMINE messages” on page 420.

3.4.2 DIAGNOSE command

To analyze a catalog for synchronization errors, you can use the IDCAMS DIAGNOSE command. With this command, you can analyze the content of catalog records in the BCS and VVDS, and compare VVDS information with DSCB information in the VTOC. Besides checking for synchronization errors, DIAGNOSE also checks for invalid data, or invalid relationships between entries.

Because the DIAGNOSE command checks the content of the catalog records, and the records might, for example, contain damaged length field values, there is a possibility that the DIAGNOSE job will abend. For detailed information on using DIAGNOSE, see *DFSMS/MVS Managing Catalogs*, SC26-4914.

3.4.3 VERIFY command

Some clarification of the word VERIFY is necessary in many cases. VERIFY is a record management macro (just like GET or PUT). It can be used with certain types of opened VSAM data sets to ensure that various fields in the VSAM control blocks in catalog are accurate. It checks the ICF catalog against the VSAM clusters.

What record management does on receiving this macro is to start reading the data set CI by CI starting with the current high used RBA value stored in a memory control block. If the data set is a KSDS, then both the index and the data are VERIFYed.

VERIFY is also an IDCAMS command. In this case, IDCAMS opens the requested data set for output, issue the record management VERIFY macro and then close the data set. When the data set is closed, VSAM Close processing uses its catalog interface to call Catalog to update the VVR information from the new information in the VSAM control blocks. It is important to understand that IDCAMS is neither updating VSAM control blocks nor the catalog directly.

Clusters, alternate indexes, entry-sequenced data sets, and catalogs can be verified. Paths over an alternate index and linear data sets cannot be verified, and the same is true of data sets in RLS mode. Paths defined directly over a base cluster can be verified.

When a data set is closed, its end-of-data and end-of-key-range information is used to update the data sets cataloged information (located in the VVR AMDSB cell). Among other fields, we have:

- High used RBA/CI for the data set
- High key RBA/CI
- Number of index levels
- RBA and the CI number of the first sequence set record
- System time stamp

Refer to 3.8, “IDCAMS LISTCAT output fields” on page 187, for more information.

A successful VERIFY means:

- The "end of the data set" indicators (data set high-used RBA/CI) etc.) are probably valid.
- There may be missing records.
- There may be duplicate records.
- The data set statistics fields in the catalog may be invalid.

The recovery actions should be the same for a VSAM data set not properly closed, whether VERIFY runs successfully or not.

Implicit VERIFY versus explicit VERIFY

Another area that is confusing when talking about VERIFY involves the terms “explicit VERIFY” and “implicit VERIFY”. An explicit VERIFY refers to the user initiating the VERIFY himself by issuing an IDCAMS VERIFY job against the data set. An implicit VERIFY refers to VSAM Open processing internally issuing the VERIFY macro against the data set when it determines that the data set was not previously closed properly; this means that a previous job which had the data set open for output has either abended or failed to close the data set for some reason.

Open processing uses a bit in the catalog to determine this. When a job opens a data set for output processing, this bit (the “open for output” bit) is turned on. It is not turned off until the job closes the data set normally. Open processing, if it finds this bit on in a subsequent open, uses GRS (enqueueing against the data set name) to determine if the data set is currently open for output. If not, then it concludes that the last close was abnormal and issues the VERIFY before completing the Open process. It also issues IEC161I messages (rc56 and rc62) to indicate that it has done this. All Open jobs against the data set will get this implicit VERIFY and the associated messages until the data set is opened for Output.

VSAM OPEN will NOT do an "implicit verify" for the following OPENs:

- The data set is being opened for input
- The data set is being opened for RESET processing
- The user requested verify to be suppressed
- The data set is being opened for Improved control interval processing
- The data set is being opened for RESTART processing
- The data set is being opened for non-ESDS create mode processing
- The data set is a Linear data set

VSAM OPEN will do an "implicit verify" or without issuing a message when the ACB being opened is the first open for output connecting to an existing control block structure and shareoptions=2,x.

Implicit VERIFY problems

Here we discuss a couple of problems with the implicit VERIFY that are not inherently obvious.

Normally, a VERIFY does not take much overhead because it is reading in CI mode from the HURBA in the catalog to the “real” HURBA. However, if the data set had much activity in the output job before it abended (for example, a log file that was being loaded), then the VERIFY could take many minutes, as it basically reads the entire file. This could also be a severe impact if there were many files open in the application.

CICS provides a good example of this problem. If the CICS region comes down hard (for example, if a CEMT P SHUT IMM is issued), then all TCBs are abended, and all files open for output at the time will have the “open for output” bit on in the VVR. This will means they will all need to be implicitly VERIFY'd as the region is brought back up. If this is the case, you need to have a procedure in place to explicitly VERIFY the important files in the case of an abend situation so that the system can come up and the important applications can start running immediately.

ACTION=REFRESH

One addition which was made to VSAM quite some time ago was the ACTION=REFRESH parameter of the VERIFY macro. Without this parameter, the VERIFY macro will only read up to the current HARBA of the data set (as set in VSAM control block). Since this value could have changed since the last time the data set was opened by the current job, this facility allows the control blocks for a data set to be updated to reflect the current structure of the data or index (from the values in the catalog/VVR). To accomplish this, record management through EOV uses catalog interface to update the in storage control blocks with the current boundaries of the data set.

Use the cluster or alternate index name as the target of your VERIFY command. Although the data and index components of a key-sequenced cluster or alternate index can be verified, the timestamps of the two components are different following the separate Verifies, possibly causing further OPEN errors.

The following sections provide some real life scenarios covering the rise and fall of a VSAM data set.

3.4.4 Broken Index scenario

In this scenario, you observe the following conditions:

- You have a program which opens a KSDS data set for update. The access argument is through RBA. However, by mistake, someone has prepared JCL pointing to the index name instead of cluster name or data name.
- The program finishes with a return code equal to zero, but with an unknown damaged index. However, the index time stamp is now different from the one in the data component, as LISTCAT shows. Nevertheless, RACF does not prohibit this action, because the user is the owner of the cluster. See Figure 3-1.

```
//*JCL -------------------------------------
//SEQ     EXEC PGM=TSTIDX
//VSAM     DD DSN=KSDS.K4REP.INDEX,DISP=SHR

TSTIDX PROGRAM:
ACB,DDNAME=VSAM,MACRF=(ADR,SEQ,OUT)
RPL,ACB=(R2),OPTCD=(ADR,SEQ,UPD,MVE)

SYSOUT MESSAGE:
JOBNAME  STEPNAME PROCSTEP    RC
TSTIDX            SEQ         00

LISTCAT ENTRY('KSDS.K4REP') ALL:
DATA: SYSTEM-TIMESTAMP: X'B3E245958A0845C4'
INDEX: SYSTEM-TIMESTAMP: X'B3E278BEC0D91601'
```

Figure 3-1 JCL, program, message and catalog entry

- The next time that you open the cluster or the data, the following happens:
  - At open, the message IEC161I 058(018)-061 is issued and the OPEN return code is 4. The processing continues. See the message at Figure 3-2.

```
IEC161I 058(018)-061:

058      The time stamp for the index does not match the time stamp for
          the data set. This could occur if the data set was updated
          without the index being open.

          System Action: OPEN processing continues. The error flags
         in the ACB (access method control block) for the data set are set
to 108.

          Programmer Response: You can continue to process the data
         set, but errors can occur if the data set and index do not
          correspond. Check for possible duplicate VVRs.
```

Figure 3-2 IEC161I message

- Going on with the processing, in a GET or PUT, the program receives a return code 8, reason code X'9C', indicating that an invalid control interval was detected. The program reading the data issues a message indicating the error. When the program is IDCAMS, the processing stops and the messages below (Figure 3-3) are issued in the SYSPRINT ddname file. If the data set is open for output, the timestamp is corrected, but the I/O error remains, once the index is damaged.

```
IDC3300I  ERROR OPENING KSDS.K4REP
IDC3351I ** VSAM OPEN RETURN CODE IS 108
IDC3302I  ACTION ERROR ON KSDS.K4REP
IDC3351I ** VSAM I/O RETURN CODE IS 156 - RPLFDBWD = X'D708009C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.
```

Figure 3-3 IDC message in the console

- You run an EXAMINE IDCAMS command, getting back the messages shown. In our test, we caused the error by filling the first control interval with X'00': See Figure 3-4.
- For recovering the index:
  - Use the REPRO Command to copy just the *data* component of the KSDS to a sequential data set. Specify the data component name (not the Cluster Name) in the REPRO INFILE parameter.
  - Sort the sequential data set by key.
  - DELETE and reDEFINE the damaged Cluster.
  - REPRO from the Sorted Sequential File to the newly defined Cluster. Record Management will rebuild the Index Component.

Note: This method does not work for a VSAM Catalog, Integrated Catalog (ICF), or for a Spanned KSDS.

```
INDEXTEST BEGINS
HIGH-LEVEL INDEX CI EXPECTED BUT NOT ACQUIRED
CURRENT INDEX LEVEL IS 3
INDEX CONTROL INTERVAL DISPLAY AT RBA/CI 245760 FOLLOWS
 000000  00000000 00000000 00000000 00000000   00000000 00000000 00000000
000000
00   *...............................*
 000020  00000000 00000000 00000000 00000000   00000000 00000000 00000000
000000
...

ERROR LOCATED AT OFFSET 00000010
MAJOR ERRORS FOUND BY INDEXTEST
LASTCC=8
```

Figure 3-4 Examine messages

3.4.5 Abend task scenario

At task abend, RTM does not properly close the VSAM data set, then:

- Buffers are not flushed (with the exception of cross systems shareoptions 4) and the HURBA is not updated in the catalog. In Language Environment there is an option (TRAP ON) which forces the close (with flush and HURBA actualization), using a STAE exit.

  If the HURBA was not updated, when the data set is subsequently opened and the user's program attempts to process records beyond end-of-data or end-of-key range, a read operation results in a “no record found” error, and a write operation might write records over previously written records. To avoid this, you can use the VERIFY command, which corrects the catalog information. For additional information about recovering a data set, see *DFSMS/MVS Managing Catalogs*, SC26-4914-04.

- If the data set was opened for output, the open-for-output indicator is left on. In this case, at next Open, VSAM implicitly issues a VERIFY command, when it detects an open-for-output indicator on and issues an informational message stating whether the VERIFY command is successful. However, a successful VERIFY does not mean that the data set is error free.

  If a subsequent Open is issued for update, VSAM turns off the open-for-output indicator at successful Close. If the data set is opened for input, however, the open-for-output indicator is left on. Refer to 3.2.3, “Mismatch between catalog and data set” on page 145, for more information.

3.4.6 Recovering damaged BCS entries

To recover damaged BCS entries, follow these steps:

1. Remove the sphere or base record, if it exists.

   The damage detected might not be in a sphere or base record. If it is not, the entry name of the sphere or base record is indicated in messages IDC21364I and IDC21365I.

2. Remove any remaining association records.

   You can re-execute the DIAGNOSE command after you remove the sphere or base record to identify any unwanted truename or association entries in the BCS. You can remove these entries by using the DELETE command with the TRUENAME parameter.

3. Reintroduce the removed entries into the catalog.

   After the damaged entries have been removed, you can redefine the data sets. For VSAM and SMS-managed non-VSAM data sets, you should specify the RECATALOG option of the DEFINE command.

If you are recovering generation data group entries, use the same procedure. However, you must reintroduce the current generation data sets into the catalog in the proper order after the generation data group has been redefined. You can use the LISTCAT command to determine the current generation data sets.

3.4.7 Recovering damaged VVDS entries

To recover damaged VVDS entries, complete these steps:

1. Remove the entries in the BCS for the data set, if they exist.

   Before the damaged VVDS records can be removed, you must remove the entries in the BCS. See 3.4.6, “Recovering damaged BCS entries” on page 172, for more details on removing BCS entries.

2. Remove the damaged VVDS records.

   After you have removed the BCS entries, you can remove the VVDS records by using the Delete command and specifying VVR or NVR. Delete VVR or NVR also removes the Format 1 DSCB from the VTOC.

3. Recover the data set from a backup copy.

If a backup copy of the data set does not exist and the data set can be opened, you can attempt to recover some of the data. Depending on the extent and type of damage in the VVDS record, you might be unable to recover any data. The data that you do recover might be damaged or out of sequence.

3.5 Prevention is better than cure

You can take a number of actions to prevent VSAM problems. Here we provide some recommendations.

3.5.1 Back up your VSAM data sets

- Back up all your critical data by using methods such as:
  - SMS management class with ABARS for backups, to allow restore of your data in the case of a hardware error and/or application error, and for use in disaster recovery.
  - Remote copy for disaster recovery.

Here, we discuss some backup issues you need to consider

IDCAMS export and import

First, we explain some of the unique characteristics of the EXPORT and IMPORT commands:

- The EXPORT command either exports a cluster or an alternate index or creates a backup copy of an integrated catalog facility catalog.

  Exporting means to store the cluster or AIX data in other media in a non-processable format, together with catalog information about the data set. An empty candidate volume cannot be exported. Access Method Services acknowledge and preserve the SMS classes during EXPORT.

  Figure 3-5 is an example in which a key-sequenced cluster, ZZZ.EXAMPLE.KSDS1, is exported from a user catalog, HHHUCAT1. The cluster is copied to a portable file, TAPE2, and its catalog entries are modified to prevent the cluster's data records from being updated, added to, or erased.

```
//EXPORT1  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//RECEIVE  DD    DSNAME=TAPE2,UNIT=(TAPE,,DEFER),
//         DISP=NEW,VOL=SER=003030,
//         DCB=(BLKSIZE=6000,DEN=3),LABEL=(1,SL)
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
EXPORT -
ZZZ.EXAMPLE.KSDS1 -
OUTFILE(RECEIVE) -
TEMPORARY -
INHIBITSOURCE
/*
```

Figure 3-5 Exporting KSDS

- IMPORT is the opposite operation to EXPORT. Here, you reload the cluster or AIX data and recatalog its catalog information in an active catalog.

  Figure 3-6 is an example in which a key-sequenced cluster, BCN.EXAMPLE.KSDS1, that was previously EXPORTed, is IMPORTed. The OUTFILE and its associated DD statement are provided to allocate the data set. The original copy of BCN.EXAMPLE.KSDS1 is replaced with the imported copy, TAPE2. Access Method Services finds and deletes the duplicate name, BCN.EXAMPLE.KSDS1, in the catalog VCBUCAT1.

  A duplicate name exists because TEMPORARY was specified when the cluster was exported. Access Method Services then redefines BCN.EXAMPLE.KSDS1, using the catalog information from the portable file TAPE2.

```
//IMPORT2  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SOURCE   DD    DSNAME=TAPE2,UNIT=(TAPE,,DEFER),
//   VOL=SER=003030,DISP=OLD,
//   DCB=(BLKSIZE=6000,LRECL=479,DEN=3),LABEL=(1,SL)
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
IMPORT -
INFILE(SOURCE) -
OUTDATASET(BCN.EXAMPLE.KSDS1) -
CATALOG(VCBUCAT1)
/*
```

Figure 3-6 IMPORT of KSDS cluster

Backup while open considerations

Backup-while-open (BWO) allows DFSMSdss logical dump for an IMS or a CICS/VSAM data set while open-for-update. BWO works with Concurrent Copy (in 9390), or Snapshot (in RVAs), or Flashcopy (in ESS). The VSAM data set must be SMS managed.

The DFSMSdss BWO function does not apply to: Catalogs, VVDSs, LDSs, physical dump, and restore.

When you define the cluster with IDCAMS, you must declare it as a BWO. The options are:

- TYPECICS

  Use the TYPECICS parameter to specify BWO in a CICS environment. For RLS processing, this activates BWO processing for CICS. For non-RLS processing, CICS determines whether to use this specification or the specification in the CICS control data named FCT.

  Note: If CICS determines that it uses the specification in the CICS FCT, the specification might override the TYPECICS or NO parameters.

- TYPEIMS

  If you want to use BWO processing in an IMS environment, use the TYPEIMS parameter.

- NO

  Use this parameter when BWO does not apply to the cluster.

To have BWO with total security and integrity, the following products are modified:

- DFSMSdfp, where catalog services have been changed to prevent unauthorized alterations of BWO indicators. DFSMSdfp does not allow deletion of a data set that DFSMSdss dumps as a BWO data set.
- DFSMSdss enqueue serialization has been changed to prevent data integrity exposures when performing defrag, dump, or restore operations.
- DFSMShsm, used for incremental backups and aggregate backups of a data set, invokes DFSMSdss to perform BWO.

In the catalog the following fields are referring to BWO:

- **BWO**: Data set is enabled for backup-while-open.
- **BWO STATUS**: Indicates the status of the data set. Status can be:
  - Data set is enabled for backup-while-open.
  - Control interval or control area split is in progress.

  - Data set has been restored and is down level. It might need to be updated with forward recovery logs.
- **BWO TIMESTAMP**: A CICS timestamp that indicates the time from which forward recovery logs have to be applied to a restored copy of the data set.

Space constraint relief

Before we start explaining more complicated recovery situations, let us address a common abend situation prior to DFSMS/MVS 1.4.

Users occasionally encounter data set allocation or extension failures (the X37 type of abends), because there is not enough space available on a volume to satisfy the request. Incidentally, VSAM does not externalize an X37 abend. You recognize the out-of-space condition by the message IEC070I 203-204, where 203 is the reason code. You find the explanation in the message IEC161I, return code 203, when no secondary space allocation quantity was specified. Return code 204 is issued when a new extend was attempted, but the maximum number of extents was reached.

SMS alleviates this out-of-space situation to some extent by performing volume selection, checking all candidate volumes before failing an allocation.

With DFSMS/MVS 1.4, you can also use the Space Constraint Relief and Reduce Space Up To (%) attributes in the data class to request that an allocation be retried if it fails due to space constraints. SMS retries the allocation by combining any of the following:

- Spreading the requested quantity over multiple volumes
- Allocating a percentage of the requested quantity
- Using more than 5 extents

Space Constraint Relief specifies whether or not to retry an allocation that was unsuccessful due to space constraints on the volume. Note that allocation is attempted on all candidate volumes before it is retried. This attribute applies only to system-managed data sets, and is limited to new data set allocations, and while extending the data set on new volumes.

Out-of-space conditions are now further reduced for new volume processing of SMS-managed data sets. VSAM and non-VSAM data sets can now acquire up to 123 extents instead of just 5 extents on a volume. Multivolume VSAM data sets can now have a maximum of 255 extents across volumes for each component, but no more than 123 extents per volume.

Two new parameters, Space Constraint Relief and Reduce Space Up To (%), are added to the SMS data class definition for this support.

Reduce Space Up to (%) specifies the amount by which you want to reduce the requested space quantity when the allocation is retried. You must specify Y for the Space Constraint Relief attribute. Valid values are 0 to 99.

If you specify Y for Space Constraint Relief, SMS begins the retry process. This is a one or two-step process, depending on the volume count you specified. For JCL allocations, SMS determines the volume count by taking the maximum of the unit, volume, or volser count. If these are not specified, SMS picks up a volume count from the data class. If there is no data class, SMS defaults the volume count to one:

- If the volume count is one (one-step process):

  SMS retries the allocation after reducing the requested space quantity based on the Reduce Space Up To attribute. SMS simultaneously removes the 5-extent limit, so that SMS can use as many extents as the data set type allows
- If the volume count is greater than one (two-step process):

  First, SMS uses a best-fit volume selection method to spread the primary quantity over more than one volume (up to the volume count). If this fails, SMS continues with the best fit method after reducing the primary quantity and removing the 5-extent limit.

If you request Space Constraint Relief but do not specify a percentage value (either 0 or blank), SMS does not reduce the requested space quantity. This implies your application cannot tolerate a reduction in the space to be allocated, so you want to remove the 5-extent limit, thereby allowing SMS to use more than 5 extents.

For extends to new volumes, Space Constraint Relief is strictly a one-step process. If regular volume selection has failed to allocate space, SMS reduces space or removes the 5-extent limit, but does not try the best-fit method.

The number of extents vary depending on data set type, as follows:

- Non-VSAM, non-extended format data sets, up to 16 extents on the volume
- Non-VSAM, extended format data sets, up to 123 extents
- PDSE, up to 123 extents on the volume
- VSAM data sets, up to 255 extents per component but only up to 123 extents per volume per component

When you request Space Constraint Relief in one or more data classes, you might notice any of the following:

- Very large allocations might succeed if a sufficiently large volume count is specified in the data class or through JCL.

- Existing data sets might end up with less space than initially requested on extents.
- The space allocated for new data sets might be less than requested.
- The number of extents used during initial allocation might result in fewer extents being subsequently available. For example, if the primary space allocation uses 10 extents when allocating a physical sequential data set, then only 6 extents are left for allocation of the secondary quantity.
- You might observe fewer X37 abends.

3.5.2 Keep your system at current maintenance levels

Keep your system at a current maintenance level. Apply PTF selective service in your operating system, especially the ones associated with VSAM. To find fixes associated with broken VSAM data sets, use the search word 'dsbreaker' in either retain or through ibmlink (askq).

3.5.3 Use Resource Recovery Management Services (RRMS)

RRMS is an z/OS component that allows you to coordinate resource recovery. Please refer to 3.10, “RRMS and VSAM” on page 199 for details.

3.6 Where to look for more information

Here we list documents and APARS that you may find helpful for further information.

3.6.1 IBM manuals and sources of relevant information

- ICF catalogs. Refer to *Integrated Catalog Facility Backup and Recovery*, SG24-5644-00.
- VSAM data sets in Record Level Sharing (RLS) mode. Refer to *CICS/ VSAM Record Level Sharing: Recovery Considerations*, SG24-4768.
- VSAM Knowledge Database, which is an interactive diagnostic tool. It is a question-and-answer driven knowledge database that resides on the DFSMS/MVS Technical Support Web site under “Technical Database” at:

  `http://knowledge.storage.ibm.com/`
- APAR, II08859 which has a methodology to assist you in fixing broken VSAM clusters.
- *DFSMS/MVS DFSMSdfp Diagnosis Reference*, LY27-9606-05.

3.6.2 Information APARs from IBMLINK on VSAM problems

There are several APARs in IBMLINK that provide useful information to assist you with VSAM problem determination.

We provide here a list of such APARs:

- II12927 - GUIDELINES FOR DOCO NEEDED BY VSAM LEVEL2
- II08859 - BROKEN DATASET - VSAM KSDS
- II10752 - ICF CATALOG PERFORMANCE PROBLEMS
- II12603 - SMSVSAM INITIALIZATION AND RECOVERY CONSIDERATIONS
- II12243 - MISC ABEND0F4 IN SMSVSAM FOLLOWING 'V XCF,'SYSNAME''
- II10001 - Document problems in VSAM, IDCAMS, and Catalog
- II13326 - COMMON PROBLEMS FROM SHCDS DEFINITION AND USAGE
- II02490 - Broken Data Set (see also II08859)
- II02516 - ATTACH macro
- II03551 - RPL error RC8 RSN28 (x'1C')
- II04390 - BLSR
- II05222 - Media Manager
- II05789 - Overlay by CCHH data
- II06620 - LSR
- II06778 - Hiperbatch
- II06941 - RNL
- II07664 - Enqueue
- II08631 - Compression Information
- II08685 - Compression Maintenance
- II08859 - Broken Data Set (see also II02490)
- II10001 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II11013 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II11513 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II12140 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II12615 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II13278 - OEM BUGS IN VSAM, CATALOG & IDCAMS
- II07664 - DIAGNOSTIC TIPS VSAMINFO : ENQUEUE

3.6.3 Information APARs on specific problems

Broken data set

- II08859
- II11955
- II08859
- II02490

ICF catalogs

- II10752

RLS

- II13326 - COMMON PROBLEMS FROM SHCDS DEFINITION AND USAGE
- II12603 - SMSVSAM INITIALIZATION AND RECOVERY CONSIDERATIONS
- II12243 - MISC ABEND0F4 IN SMSVSAM FOLLOWING 'V XCF,'SYSNAME''

TVS

OEM and other IBM products

- II10001
- II11013
- II11513
- II12140
- II12615
- II13278

Lock problems

- BDC000022923 - Loop if locksize is too large
- RTA000141603 - Rebuild of IGWLOCK00
- BDC000018772 - MAX IGWLOCK00
- BDC000011289 - VIOPLUS

Export / Import tips

- II07902

Memory shortage

- BDC000013015
- II05506

3.6.4 VSAM information on the Internet

VSAM Knowledge base

`http://knowledge.storage.ibm.com/`

DFSMS support sites

`http://ssddom02.storage.ibm.com/techsup/webnav.nsf/support/dfsms`

Flashes

`http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/Flashes`

3.7 IDC3009I message

When working with catalogs and VSAM data sets, the IDC3009I message is probably the most common message you receive. IDC3009I is issued as a result of a *catalog* error or exceptional condition involving a catalog.

The format of this message is shown here:

```
IDC3009I VSAM CATALOG RETURN CODE IS return-code – REASON CODE IS IGGOCLaa –
reason-code
```

The message specifies a return code and a reason code, and they are described in *OS/390 MVS System Messages, Vol 3 (GDE-IEB)*, GC28-1786. The return codes are as listed in Table 3-1. The table is not intended to replace the messages manual, but rather to serve as a quick reference.

Table 3-1 IDC3009I message

Return Code	A brief explanation
4	Error while performing open/close to a VSAM catalog. See reason code, can be either a small region size.
8	The specified entry does not exist if locate was the action *or* the entry already exists, if action is one which adds an entry to a catalog.
10	An incorrect record type was found in the catalog.
12	The component was not found. It can be an AIX, a data or an index, depending on reason code.
14	Catalog cell not found.: Run the Access Method Services DIAGNOSE command to check for additional information.
16	Request is not supported for SMS managed volumes.
18	ALTER for a backup *or* migrated data set failed.
20	VSAM catalog has run out-of-space.
22	Catalog field vector table (FVT) is zero or an incorrect FVT field was found.
24	Permanent read error in VSAM catalog.
26	ICF catalog: VSAM record management error (catalog record too big or too small).
28	Permanent I/O error in VSAM catalog. Messages IEC331I, IEC332I, and IEC333I have been printed to aid in determining the cause of the error and where the error occurred.
30	Automated Tape Library DataServer (ATLDS) processing error.

Return Code	A brief explanation
32	Error in the VSAM catalog parameter list and indicates an internal error in Access Method Services.
36	Data set not found or DSCB indicates a VSAM data set.
38	Error found in a catalog installation error (user or DFHSM supplied) while processing a CATALOG, INDEX or LOCATE macro. If DFHSM, messages (ARCxxxI) are issued before.
40	Two or more tasks are modifying a catalog entry, causing it to be extended in size, and one task finds that it was unable to specify sufficient virtual storage for catalog management's new requirements.
42	A DADSM error occurred on branch entry to DADSM back end (DADSM rename, locate or scratch, according to reason code).
44	The caller's work area is too small.
48	Incorrect VSAM catalog function.
50	An error has been detected in VVDS manager error.
52	Permanent I/O error on user volume. An attempt to run a direct execute channel program (EXCP) write of a DSCB to the VTOC failed. The reason codes are the event control block (ECB) completion codes returned after the EXCP write and you can the meaning in “Event Control Block Fields”,*DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921.
54	Incorrect use of JOBCAT or STEPCAT with SMS data sets.
56	A security verification failed.
58	Error while reading a DSCB into a work area. Reason code is DADSM OBTAIN Return Code and you can find in “Return Codes from OBTAIN”, *DFSMS/MVS DFSMSdfp Advanced Services*, SC26-4921.
60	Incorrect entry type for requested action.
62	Error while initializing the extension of a data set. The reason codes are the complement of the error codes returned from the DADSM extend routine; refer to *DFSMSdfp Diagnosis Reference*, LY27-9606.
64	VSAM catalog cannot find either a data or an index entry which is associated with a cluster or alternate index entry.
66	Bad DADSM parameter list. The reason code is the DADSM return code, and you can find in *DFSMSdfp Diagnosis Reference*, LY27-9606.
68	No space is available on the user volume for the ICF catalog. Only the primary volume will be used.
70	A generation data set component was not found in the GDG sphere record.

Return Code	A brief explanation
72	The user volume is not mounted. The reason codes are from VSAM open/close/end-of-volume, volume mount and verify routine IDA0192V.
74	Catalog Cell not found. Different reasons codes from those specified in return code 14.
76	No unit available for mounting or volume not mounted.
78	Subrecord move error. It can be: Catalog management was unable to obtain enough virtual storage to contain the catalog record with the addition of the subrecord *or* verification record was not within the length range of 1 to 256.
80	The object specified in the RELATE parameter of a DEFINE command does not exist, or is improper for the type of object being defined.
82	The number of data set entries passed exceeds the allowed maximum for the catalog name locate.
84	Date error: a DELETE to a data set with an unexpired purge date and PURGE was not specified *or* there are conflicting date format (rc=2).
86	Recatalog error, reason vary according to reason codes.
88	Error with a catalog recovery area (CRA) define operation: The total space specified was not able to contain the size specified for the catalog and the one cylinder of space required for the CRA.
90	Delete error.
92	The maximum number of extents was reached.
94	A DADSM OBTAIN request failed during a VSAM catalog delete request. The reason code is the OBTAIN return code and you can find out its meaning in "DADSM OBTAIN Function Return Codes", *DFSMSdfp Diagnosis Reference*, LY27-9606.
96	An error occurred in specifying key length, key position or record size for an alternate index or spanned cluster.
98	An unusual condition occurred during an ALTER name of a unique or non-VSAM data set. The reason code is *status byte* returned by the DADSM RENAME function. See the meaning in "Status Codes from RENAME", *DFSMSdfp Diagnosis Reference*, LY27-9606.
102	A DADSM SCRATCH request failed during a VSAM catalog delete request. for a unique or non-VSAM data set. The reason code is *status byte* returned by the DADSM SCRATCH function. See the meaning in "Status Codes from SCRATCH", *DFSMSdfp Diagnosis Reference*, LY27-9606.
104	A DEFINE command is attempting to define a second VSAM master catalog when a VSAM master catalog already exists and is open.
106	A format-4 DSCB processing error was encountered.

Return Code	**A brief explanation**
108	An incorrect field name was found in the field parameter list. The field name passed by AMS does not exist in the VSAM catalog management dictionary. The message indicates that the caller's AMS release level or maintenance level is different from the CATALOG level that is being called.
110	Unable to modify or delete RACF profile. It does not exist (but has RACF indicator (reason 4) or a rename processing for a RACF-protected data set failed because as a result of the new name, the data set cannot be defined to the security subsystem.
112	Incorrect Catalog field parameter list (FPL).
114	As a result of an IMPORT, IMPORTRA, or DEFINE command, VSAM has attempted to establish a RACF profile for a cluster/alternate index, data, or index object (reason codes 0, 4, and 8 respectively). This failed because a profile with the same name already exists.
116	VSAM catalog records are incorrect. The reason code explain for what kind of object the record can not be obtained.
118	The data set name is ineligible for RACF definition. User does not have authority (reason code 0) or RACF inactive (reason code 12).
120	Attempt to modify the non-existent or system field. This is a system error.
124	Incorrect control interval number.
126	Alter new name of a GDS, non-VSAM or cluster failed because an ACS service returns a non-zero return code. ACS reason code = catalog reason code + 1000. services reason codes from 1000 to 1255. The ACS services return and reason codes are documented in the *DFSMSdfp Diagnosis Reference*, LY27-9606.
128	A user-provided storage is outside the user region. Probable system error.
130	An ALTER RENAME recatalog error. Reason code explains why.
132	Incorrect pointer value in argument list. Probable system error.
136	Required parameters not supplied. Probable system error.
138	DADSM RENAME error. Reason code is the volume status code returned from DADSM.
142	DADSM OBTAIN error. Reason code is the return code from OBTAIN and you find in *DFSMSdfp Advanced Services*, SC26-4921.
144	An incorrect entry name format or the name has an initial character as a numeric or used a restricted name. See reason code.
148	Volume already owned by another VSAM catalog. Specify a different volume or use the Access Method Services ALTER REMOVEVOLUMES command to reset the volume ownership if a catalog should not own the volume.

Return Code	A brief explanation
150	Name length error for an SMS construct. Storage class (reason code 2) or data class (4) or management class (6).
152	A non-empty catalog cannot be deleted. If the catalog and all of its entries are to be deleted specify the FORCE parameter on the Access Method Services DELETE CATALOG command.
156	The volume does not contain a data space for another VSAM data set. There is insufficient space in the data spaces allocated on the volume to satisfy a request for suballocation.
160	DELETE space is requested for a volume containing a catalog. See explanation and programmer response in reason code.
164	There is insufficient virtual storage available for VSAM catalog management. Increase the region size available to the step.
168	Unsupported device type.
172	A DEFINE command specifies the name of a data set, with the UNIQUE attribute, but there is already a data set on the specified volume with that name.
176	There is no space in the VTOC for a DSCB. Delete any unneeded data sets or data spaces from the volume or recreate the volume with a larger VTOC.
178	An error occurred during ICF catalog processing of a VSAM partial release request.
180	Data space name not found. Probable system error. The catalog or a volume may have been totally or partially destroyed.
182	Bad DADSM UPDATE parameter list. The reason code is the Volume status code from DADSM. You can see a volume status using ISMF.
184	The data set is currently open and cannot be deleted or altered.
186	Error attempting to lock a catalog or access a locked catalog.
188	Catalog unavailable. See return code in the messages manual.
190	Authorization error on a facility class function applied to SMS data sets. The user need read access authority to 'STGADMIN.IGG.LIBRARY'.
192	Maximum logical record length specified is greater than 32,761 for a non-spanned data set.
194	An error occurred during multi-level alias (MLA) facility processing. See reason code in the message manual.
196	The data component control interval size specified is greater than 32,767.
198	An attempt has been made to use an unsupported feature. Related to a UCB's device defined above 16M, which resides a VSAM catalog.

Return Code	A brief explanation
200	The specified or defaulted control interval size of the index component is greater than the maximum block size of the index device. Reduce the control interval or use a device with a larger maximum block size.
202	Storage management subsystem call error. Redefine the data set with a management class with no retention limit or with a specified retention value equal to or exceeding the date specified in the ALTER command.
204	Key specification extends beyond end of maximum logical record. Reduce the key length, change the key position, or increase the logical record length.
208	The buffer space specified is too small. Do not specify BUFFERSPACE, let the default value.
210	Subsystem call error. Reason code is the return code from Subsystem call.
212	Control interval size calculation unsolvable. See reason code.
214	Subsystem call error. Reason code is the return code from Subsystem call.
216	The volume's VTOC is not interpretable. An incorrect VTOC was deleted during update extend processing for a VSAM data set. Restore the volume in order to correct the VTOC.
220	A DOS VTOC cannot be converted to an OS VTOC. Restore the volume in order to correct the VTOC.
222	Alter new name of a GDS, non-VSAM or cluster failed because an ACS service returns a non-zero return code. ACS reason code = catalog reason code + decimal 256. ACS services return and reason codes are documented in *DFSMSdfp Diagnosis Reference*, LY27-9606.
224	A field in a catalog entry has exceeded the maximum allowable number of repetitions. For example, trying to add more than 255 volumes or more than 125 alternate indexes in the upgrade set.
226	The caller is not authorized to perform the requested function. The caller must be running in key 0 - 7, must be in supervisor state, or must be APF authorized.
228	An error occurred while processing an Enhanced Catalog Sharing (ECS) request. Refer to reason code in the message manual to correct the problem.
230	VSAM catalog retrieve of a control interval failed to get a low range record from the VSAM catalog. Probable system error.
232	An error was encountered while VSAM Catalog Management was performing SMF processing. Use the reason code as a return code to IDC3009I to find out the reason of the error.
234	End of data encountered while reading the low data key range of the VSAM catalog. Probable system error.
236	An error was encountered in space-map. This condition arises when the catalog's volume entry is incorrect. Reconstruct the volume entry record. If that is not possible, restore the catalog.

Return Code	A brief explanation
238	No user catalog entry in the master catalog for Convert Volume processing.
240	Required DD statement not supplied. See reason code in the message manual.
242	A physical I/O error occurred trying to erase the data set being deleted. The reason code correspond to the VSAM Record Management error codes. See “Record Management Return and Reason Codes” in *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913. Run the job again with the NOERASE option. The data set cannot be deleted.
244	Erase action failed. The VSAM Catalog Management is unable to open the VSAM data set being deleted. The reason codes correspond to the VSAM OPEN error codes. See “OPEN Return Codes” in *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913. Alternatively, you run the DELETE command again with the NOERASE option.
246	CAS service task abended or detected an abnormal condition. See reason code in the messages manual.
248	A function requires a volume that is not owned by the VSAM catalog being used.
250	VSAM Record Management has found a *logical error* during erase processing while deleting a VSAM data set. The reason codes correspond to the VSAM record management logical error code. See “Record Management Return and Reason Codes” in *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913. Alternatively, you run the DELETE command again with the NOERASE option.
254	An error was encountered during catalog reorientation. The reason code indicates in what part the error was found: close (reason code 0), open (2), allocation (4) or an unexpected error (6).
Note: Reason codes, explanations, system actions and programmers response are described in z/OS V1R4.0 MVS System Messages, Vol 6 (GOS-IEA) under IDC3009I message.	

3.8 IDCAMS LISTCAT output fields

In the ICF catalog, VSAM maintains special RBA, CI, and Key values together with time stamps, which are very important in accessing a VSAM cluster. The most important ones are located in the VVR AMDSB in the VVDS cell and are updated only at Close time.

IDCAMS LISTCAT command displays this information, which is key to diagnosing what caused the error. The screen below shows some JCL you need in order to issue the LISTCAT command via BATCH. You can issue the same command under TSO/ISPF.

```
//LISTCAT  JOB 'LISTCCAT EXAMPLE',NOTIFY=&SYSUID
//*-----------------------------------------------------------------*
//EXAMPLE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           LISTC ENTRY(DAWN.KSDSEXG)  ALL
```

The output from the LISTCAT command is:

```
1IDCAMS  SYSTEM SERVICES                                          TIME:
21:15:46       04/20/00     PAGE      1
0
            LISTC ENTRY(DAWN.KSDSEXG)  ALL
00044403
OCLUSTER ------- DAWN.KSDSEXG
      IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
      HISTORY
        DATASET-OWNER-----(NULL)     CREATION--------2000.108
        RELEASE----------------2     EXPIRATION------0000.000
      SMSDATA
        STORAGECLASS -----STRIPE     MANAGEMENTCLASS---MCDB22
        DATACLASS ------KEYEDEXG     LBACKUP ---0000.000.0000
        BWO STATUS------00000000     BWO TIMESTAMP---00000 00:00:00.0
        BWO---------------(NULL)
      RLSDATA
        LOG ----------------(NULL)   RECOVERY REQUIRED --(NO)
        VSAM QUIESCED -------(NO)    RLS IN USE ---------(NO)
0       LOGSTREAMID----------------------------(NULL)
        RECOVERY TIMESTAMP LOCAL-----X'0000000000000000'
        RECOVERY TIMESTAMP GMT-------X'0000000000000000'
      PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
      ASSOCIATIONS
        DATA-----DAWN.KSDSEXG.DATA
        INDEX----DAWN.KSDSEXG.INDEX
0   DATA ------- DAWN.KSDSEXG.DATA
      IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
      HISTORY
        DATASET-OWNER-----(NULL)     CREATION--------2000.108
        RELEASE----------------2     EXPIRATION------0000.000
        ACCOUNT-INFO-----------------------------------(NULL)
      PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
      ASSOCIATIONS
        CLUSTER--DAWN.KSDSEXG
      ATTRIBUTES
        KEYLEN-----------------8     AVGLRECL-------------300
BUFSPACE-----------10240     CISIZE--------------4096
```

```
        RKP--------------------0     MAXLRECL-------------300
EXCPEXIT----------(NULL)     CI/CA----------------192
        STRIPE-COUNT-----------4
        SHROPTNS(1,3)   RECOVERY     UNIQUE          NOERASE     INDEXED
NOWRITECHK     NOIMBED       NOREPLICAT
        UNORDERED        NOREUSE     NONSPANNED      EXTENDED
      STATISTICS
        REC-TOTAL---------321595     SPLITS-CI-----------6466
EXCPS--------------82682
        REC-DELETED-------173530     SPLITS-CA-------------42
EXTENTS----------------4
        REC-INSERTED-------45123     FREESPACE-%CI---------10
SYSTEM-TIMESTAMP:
        REC-UPDATED--------56016     FREESPACE-%CA---------10
X'B3ECB2FDD72E7785'
        REC-RETRIEVED----3186326     FREESPC--------610766848
      ALLOCATION
        SPACE-TYPE---------TRACK     HI-A-RBA-------736100352
        SPACE-PRI-----------3744     HI-U-RBA-------203685888
        SPACE-SEC------------624
      VOLUME
1IDCAMS  SYSTEM SERVICES                                         TIME:
21:15:46        04/20/00     PAGE      2
0       VOLSER------------SBOX31     PHYREC-SIZE---------4096
HI-A-RBA-------736100352     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12
HI-U-RBA-------203685888     EXTENT-TYPE--------X'00'
        VOLFLAG------------PRIME     TRACKS/CA--------------4
        STRIPE-NUMBER----------1
        EXTENTS:
        LOW-CCHH-----X'000A000A'     LOW-RBA----------------0
TRACKS--------------3744
        HIGH-CCHH----X'01040003'     HIGH-RBA-------736100351
      VOLUME
        VOLSER------------SBOX30     PHYREC-SIZE---------4096
HI-A-RBA-------736100352     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
        VOLFLAG------------PRIME     TRACKS/CA--------------4
        STRIPE-NUMBER----------2
        EXTENTS:
        LOW-CCHH-----X'0104000B'     LOW-RBA----------------0
TRACKS--------------3744
        HIGH-CCHH----X'01FE0004'     HIGH-RBA-------736100351
      VOLUME
        VOLSER------------MHLV14     PHYREC-SIZE---------4096
HI-A-RBA-------736100352     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
```

```
        VOLFLAG------------PRIME     TRACKS/CA--------------4
        STRIPE-NUMBER----------3
        EXTENTS:
        LOW-CCHH-----X'0103000B'     LOW-RBA----------------0
TRACKS--------------3744
        HIGH-CCHH----X'01FD0004'     HIGH-RBA-------736100351
      VOLUME
        VOLSER------------SBOX28     PHYREC-SIZE---------4096
HI-A-RBA-------736100352     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
        VOLFLAG------------PRIME     TRACKS/CA--------------4
        STRIPE-NUMBER----------4
        EXTENTS:
        LOW-CCHH-----X'0103000B'     LOW-RBA----------------0
TRACKS--------------3744
        HIGH-CCHH----X'01FD0004'     HIGH-RBA-------736100351
0   INDEX ------ DAWN.KSDSEXG.INDEX
      IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
      HISTORY
        DATASET-OWNER-----(NULL)     CREATION--------2000.108
        RELEASE----------------2     EXPIRATION------0000.000
      PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
      ASSOCIATIONS
        CLUSTER--DAWN.KSDSEXG
      ATTRIBUTES
        KEYLEN-----------------8     AVGLRECL---------------0
BUFSPACE---------------0     CISIZE--------------2048
        RKP--------------------0     MAXLRECL------------2041
EXCPEXIT----------(NULL)     CI/CA-----------------21
       SHROPTNS(1,3)   RECOVERY     UNIQUE          NOERASE     NOWRITECHK
NOIMBED     NOREPLICAT     UNORDERED
        NOREUSE         EXTENDED
      STATISTICS
        REC-TOTAL------------263     SPLITS-CI-------------42
EXCPS--------------39045     INDEX:
        REC-DELETED------------0     SPLITS-CA--------------1
EXTENTS----------------2     LEVELS-----------------3
        REC-INSERTED-----------0     FREESPACE-%CI----------0
SYSTEM-TIMESTAMP:            ENTRIES/SECT----------13
        REC-UPDATED--------21884     FREESPACE-%CA----------0
X'B3ECB2FDD72E7785'     SEQ-SET-RBA------------0
      REC-RETRIEVED----------0   FREESPC------------63488
HI-LEVEL-RBA------436224
1IDCAMS  SYSTEM SERVICES                                            TIME:
21:15:46        04/20/00     PAGE      3
0     ALLOCATION
        SPACE-TYPE---------TRACK     HI-A-RBA----------602112
        SPACE-PRI-------------12     HI-U-RBA----------538624
        SPACE-SEC--------------2
      VOLUME
        VOLSER------------SBOX31     PHYREC-SIZE---------2048
HI-A-RBA----------602112     EXTENT-NUMBER----------2
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------21
HI-U-RBA----------538624     EXTENT-TYPE--------X'00'
        VOLFLAG------------PRIME     TRACKS/CA--------------1
        EXTENTS:
        LOW-CCHH-----X'00000002'     LOW-RBA----------------0
TRACKS----------------12
        HIGH-CCHH----X'0000000D'     HIGH-RBA----------516095
        LOW-CCHH-----X'01040004'     LOW-RBA-----------516096
TRACKS-----------------2
        HIGH-CCHH----X'01040005'     HIGH-RBA----------602111
      VOLUME
        VOLSER------------SBOX30     PHYREC-SIZE---------2048
HI-A-RBA---------1032192     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------21
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
        VOLFLAG-------CAND-SPACE     TRACKS/CA--------------1
        EXTENTS:
        LOW-CCHH-----X'01FE0005'     LOW-RBA-----------516096
TRACKS----------------12
        HIGH-CCHH----X'01FF0001'     HIGH-RBA---------1032191
      VOLUME
        VOLSER------------MHLV14     PHYREC-SIZE---------2048
HI-A-RBA---------1548288     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------21
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
        VOLFLAG-------CAND-SPACE     TRACKS/CA--------------1
        EXTENTS:
        LOW-CCHH-----X'01FD0005'     LOW-RBA----------1032192
TRACKS----------------12
        HIGH-CCHH----X'01FE0001'     HIGH-RBA---------1548287
      VOLUME
        VOLSER------------SBOX28     PHYREC-SIZE---------2048
HI-A-RBA---------2064384     EXTENT-NUMBER----------1
        DEVTYPE------X'3010200F'     PHYRECS/TRK-----------21
HI-U-RBA---------------0     EXTENT-TYPE--------X'40'
        VOLFLAG-------CAND-SPACE     TRACKS/CA--------------1
        EXTENTS:
        LOW-CCHH-----X'01FD0005'     LOW-RBA----------1548288
TRACKS----------------12
        HIGH-CCHH----X'01FE0001'     HIGH-RBA---------2064383
1IDCAMS  SYSTEM SERVICES                                          TIME:
21:15:46        04/20/00     PAGE      4
0         THE NUMBER OF ENTRIES PROCESSED WAS:
                    AIX -------------------0
                    ALIAS -----------------0
                    CLUSTER ---------------1
                    DATA ------------------1
                    GDG -------------------0
                    INDEX -----------------1
                    NONVSAM ---------------0
                    PAGESPACE -------------0
                    PATH ------------------0
                    SPACE -----------------0
                    USERCATALOG -----------0
                    TAPELIBRARY -----------0
                    TAPEVOLUME ------------0
                    TOTAL -----------------3
0         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0
0IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Use the output from the LISTCAT command listed above to follow the explanations about the most key catalog fields shown by LISTCAT.

3.8.1 High used RBA value (HURBA) for KSDS

There are two HURBAs per KSDS cluster: the data HURBA, and the index HURBA:

- Data HURBA

  This is the RBA of the physically highest record in data CI (it does not imply the highest key, because the existence of splits). In other words, it points to the end of the last CA which ever, at any time, contained data. It is incremented by one CA, if a CA split or add-to-the-end in a new CA occurs. It is always on a CA boundary.

  If all logical records are deleted the HURBA does not turn to zero — only via create or resetting. Also, if the data set was defined with the Reuse option, in this case at Open time, HURBA is reset to zero.

  When share option cross-system 4 is set to the data set, this value cannot be modified.

  If a data set has HURBA=0, it cannot be opened for input.

- Index HURBA

  This points to the end of the last index CI written. It is incremented by one CI if a new record is added to the index. Separate HURBA values are maintained for the imbedded sequence set, and the high level index if IMBED is used. Index HURBA is always on a CI boundary.

There are also two types of data components:

- ESDS data component

  The HURBA points to the end of the last CI which contains data. It is incremented by one CI, if a new CI is entered via add-to-end processing.

  It is always on a CI boundary.

- RRDS data component

  The HURBA points to the end of the last CA which ever contained data. It is incremented by add-to-end processing which enters a new CA, or by direct insert of a record whose slot number resolves to a new CA.

  It is always on a CA boundary.

The name, HURBA, does not mean that every byte up to the HURBA is used. There may be imbedded free space due to distributed space, record deletion, slots which have never been used (RRDS) as well as unused space at the end of the CI or CA. The HARBA is an RBA pointer to the end of the last current extent of that cluster component. Until the HURBA equals the HARBA for that component, another extent will not be taken.

3.8.2 High allocated RBA value (HARBA)

This is the highest RBA available within allocated space to store the data component, its key range, the index component, or the sequence set records of a key range. KSDS has two HARBAs: one for the index; another for data.

The difference (HARBA - HURBA) is the amount of space ready to be freed by the free space release option after close. It includes all the CAs in the physical end of the data sets with only free CIs.

3.8.3 FREESPC

This is the actual number of bytes of free space in the total amount of space allocated to the data or index component. Free space in partially used control intervals is not included in this statistic.

3.8.4 High key RBA/CI

It is the RBA of the logically highest data CI (the one with the record with the highest key). When the cross-system SHAREOPTION 4 is set to the data set, this value cannot be modified.

3.8.5 High-level index RBA value

Any time the data set is accessed directly through a key, VSAM uses this value to find the highest level index record to start the search through lower levels.

If the high-level index RBA is corrupted, the user will not be able to perform direct requests against the data set.

3.8.6 Sequence set first RBA value

If the data set is being read sequentially, from start to finish (for example, REPRO), VSAM uses this value to go directly to the first sequence set record. If Sequence Set RBA is corrupted sequential access will not be possible.

3.8.7 Number of index levels

The number of levels of index records in the index. for an KSDS/VRRDS cluster. If this number is greater than four (meaning a very big data set), maybe it is an indication for reorganization increasing the size of the CI index.

3.8.8 Time stamps

At close time, if the cluster is open for output, the KSDS time stamps are updated with the current system time (same value for both data and index). However, each component¢s individual time stamp is updated in the catalog only if it is greater than the component's time stamp currently in the catalog. Prior to updating time stamps IN THE CATALOG, Close writes SMF record type 64. The time stamp ordering the SMF record should slightly precede that in the catalog.

At Open time, if the time stamp of the index component is less than that of the data component, the data component is updated separately and after the index component, or vice-versa.

3.9 SMF record types related to VSAM data sets

Following are the SMF records related to VSAM recovery and VSAM performance.

3.9.1 SMF record type 60

Record type 60 is written when a record is inserted, updated, or deleted from a VSAM Volume Data Set (VVDS). For example, when a VSAM cluster is defined, closed, or deleted.

VVDS is a part of ICF catalog structure (the other is BCS), located in the volume which contains the described data sets. It contains dynamic information, as statistics, about these data sets. VSAM data sets and SMS data sets must be cataloged in an ICF catalog. The record related to a VSAM data set is a VSAM Volume Record (VVR), while the record related to non-VSAM data sets is a Non-VSAM Volume Record (NVR).

One type 60 record is written for each VVR or NVR written or deleted. This record:

- Identifies the VVDS in which the VVR or NVR is written or deleted.
- Gives the new, updated, or deleted VVR or NVR.
- Identifies the job by job log and user identifiers.

3.9.2 SMF record type 61

One type 61 record is written for each record inserted or updated in a catalog. This record:

- Identifies the entry being defined and the catalog in which the catalog record is written.
- Gives the new or updated catalog record.
- Identifies the job by job log and user identifiers.

3.9.3 SMF record type 62

Record type 62 is written at the successful or unsuccessful opening of a VSAM component or cluster. The record:

- Identifies the VSAM component or cluster.
- Indicates whether it was successfully opened.
- Names the VSAM catalog in which the object is defined and the volumes on which the catalog and object are stored.
- It identifies the job that issued the OPEN macro by job log identification and user identification.

This record is not generated when a system task issues the OPEN macro.

3.9.4 SMF record type 63

Record type 63 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is defined by the DEFINE Access Method Services command and when that definition is altered. For example, when a VSAM catalog entry is altered with new space allocation information (that is, when the VSAM End-Of-Volume (EOV) routine extends the entries object) or, if the entry is changed by the Alter Access Method Services command. One record type 63 is written for each newly created or altered entry. This record is not written when a VSAM catalog is renamed. In that case record type 68 is written. This record:

- Identifies the catalog in which the object is defined.
- Gives the catalog record for the newly defined object, and, for an alteration, gives the parts of the old catalog record before they were altered.
- Identifies the job and the user that caused the record to be written. If it was caused by a system task, the job-name and the user-identification fields contain blanks and the time and date fields contain zeros.

3.9.5 SMF record type 64

Record type 64 is written when:

- A VSAM component or cluster is closed.
- VSAM must switch to another volume to continue to read or write.
- The component ran out of space and EOV is called to extend the component.

When a cluster is closed, one record is written for each component in the SMF record type 64. The reason why the record was created is indicated in the record.

SMF record type 64 description

The record describes the device and volume(s) on which the object is stored, and gives the extents of the object on the volume(s). It gives statistics about various processing events that have occurred since the object was defined, such as the number of records in the data component, the number of records that were inserted, and the number of control intervals that were split.

The record written when the VSAM component or cluster is closed contains changes in statistics from OPEN to time of EOV and CLOSE.

SMF64 sample program

The IDCAMS LISTCAT command shows the cumulative number of EXCPs, since the initial load. Sometimes is more important to know the number of EXCPs executed between OPEN and CLOSE, mainly when you are doing tuning in buffering. The section “Sample programs extract from SMF record type 64” on page 367, contains sample source assembler code. It can be used for showing more information about your VSAM data sets. By using JCL parameters, you can determine an EXCPs threshold. Then, the program only shows data covering the data sets with equal or more EXCPS than the threshold that occurred between open and close. The report generated is based on the SMF 64 record, generated each time a VSAM data set is closed. It can help you to determine the characteristics of the data sets with high I/O activity. To reduce the I/O activity:

- You can use SMB if the data sets are already in extended format.
- You can convert data sets to extended format and then use SMB.
- In the case of LSR buffering and direct access, you can find out if the data sets are using defer write, and if not, whether they could be.
- You can determine whether the VSAM buffers are below 16 MB, and whether to move them above 16 MB.
- For KSDS and VRRDS data sets, when the total number of free control intervals is much higher than free space defined to the data set consider reorganization. Do not reorganize data sets if it is not necessary.
- When doing changes in buffering, you can use the report to see how the numbers of EXCPs decreased.

For a description of SMF type 64 fields, refer to *OS/390 MVS System Management Facilities (SMF)*, GC28-1783.

3.9.6 SMF record type 65

Record type 65 is written during any processing that results in a DELETE request to Catalog management services, such as:

- IDCAMS DELETE
- IEHPROGM UNCATLG

One type 65 record is written for each record updated or deleted from a catalog. The record:

- Identifies the entry being deleted.
- Identifies the catalog in which the catalog record is updated or deleted.
- Gives the updated or deleted catalog record.
- Indicates whether a VSAM cluster or non-VSAM data set was scratched, or whether only catalog information was deleted.
- Identifies the job and the user. If a system task caused the record to be written, the job name and user identification fields contain blanks, and the time and date fields contain zeros.

3.9.7 SMF record type 66

Record type 66 is written during any processing that results in an ALTER request to Catalog Management Services, such as IDCAMS ALTER.

One type 66 record is written for each record written or deleted from a catalog. The record:

- Identifies the entry being altered.
- Identifies the catalog in which the catalog record is written or deleted.
- Gives the new, updated, or deleted catalog record.
- Indicates if the entry was renamed and, if so, gives the old and new names of the entry.
- Identifies the job and the user. If a system task caused the record to be written, the job name and user identification fields contain blanks and the time and date fields contain zeros.

3.9.8 SMF record type 67

Record type 67 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is deleted. A record is written for each entry affected by the DELETE Access Method Services command. For example, three records are written for an indexed cluster; one for the relationship between the components of the cluster, one for the data component, and one for the index component. The record:

- Identifies the deleted entry.
- Identifies the VSAM catalog in which the entry was defined.
- Gives the total logical VSAM catalog record.
- Identifies the job and user that caused the record to be written. If it was caused by a system task the job-name and the user-identification fields contain blanks and the time and date fields contain zeroes.

3.9.9 SMF record type 68

Record type 68 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is renamed using the ALTER Access Method Services command. This record:

- Identifies the VSAM catalog in which the object is defined.
- Gives the old and new names for the object.
- Identifies the job and user that renamed the data set. If a system task caused the record to be written, the job-name and user-identification fields contain blanks and the time and date fields contain zeros.

3.9.10 SMF record type 69

Record type 69 is written when a VSAM data space is defined, extended, or deleted using the DEFINE or DELETE Access Method Services commands. Record type 69 is not written when a catalog or a unique data set is defined or deleted. This record:

- Identifies the catalog in which the data space is defined.
- Identifies the volume on which it is (or was) allocated.
- Gives the number of free data space extents and the amount of unallocated space on the affected volume after the definition, extension, or deletion.
- Identifies the job and the user that caused the record to be written. If it is caused by a system task, the job-name and user-identification fields contain blanks and the time and date fields.

3.9.11 SMF record type 42

This record provides VSAM record level sharing (RLS) statistics.

3.10 RRMS and VSAM

S/390 provides Resource Recovery Management Services (RRMS), comprising:

- Resource Recovery Services (RRS), which provide sync-point services.
- Registration Services, which allow a resource manager to define itself to the operating system.
- Context services, which allow a resource manager to indicate interest in a work context. A context represents the resources for a work request; a context consists of the application program requesting the work and the protected resources involved in the work. A context represents a business unit of work: one or more units of recovery with the associated application programs, resource managers, and protected resources.

RRS is an OS/390 component capable to coordinate Resource Recovery in MVS.

Resource recovery includes a set of APIs and protocols allowing a transaction executing an application program to *modify* consistently multiple protected resources. The most common is 2-phase commit protocol.

Among these resources, we may have databases, VSAM data sets, and any product specific resource, managed by distinct resource managers. These resource managers maybe located in different systems.

Resource recovery scenarios has three agents:

- Application Program (AP): Requests the changes.
- Resource Manager (RM): Controls the access to the resource, for example, Data Manager (DB2, DL/I, VSAM) and Work Manager (CICS, IMS/DC).
- Synchpoint Manager (SM) guarantees resource recovery by the implementation of 2-phase commit. RRS is an example of an SM.

When an AP is running the same unit of work (transaction) under CICS and IMS/DC, it is able to update multiple data located in DB2, DL/I, VSAM files. In this case, there is not a need for RRS, because the SM role is executed by CICS or IMS/DC through the Commit and Rollback functions.

RRS allows Resource Recovery (2-phase commit) when an unit of work crosses multiple subsystems (MQSeries®, CICS, IMS) and multiple OS/390 images.

Unit of recovery (UR) is a set of changes that *all* must be done or *none* is done. It guarantees the integrity of the updates.

Two-Phase Commit Protocol aims the execution of an UR, that is, all or nothing. Has two phases:

- Phase 1:
  - App informs the changes to RMs.
  - RMs log the old (UNDO) and the new (REDO) data.
  - AP asks a commit to Synchpoint Manager.
  - Synchpoint Manager asks RMs if they can commit (prepare commit). If all say “yes”, then Synchpoint Manager hardens the commit in its journal. If *not* all say “yes”, the commit is *not* hardened, and the backout is commanded at once to the RMs (erase the log) at phase-2.
- Phase 2:
  - Synchpoint Manager orders the commit (if hardened) or the backout (if not hardened) of the changes represented by the UR.
  - If all RMs return OK, Synchpoint Manager returns a return code committed to the AP. If not, then application changes will be made during restart.

Then, along a 2-phase commit, we may have two functions: commit or backout.

In a pure VSAM application guaranteeing the APIs for a 2-phase commit when you want to implement a unit of recovery, due to updates in multiple VSAM data sets, you will find the RRs to be very helpful.

For more information, refer to *OS/390 V2R8.0 MVS Programming: Resource Recovery,* GC28-1739.

4

Managing your VSAM data sets

In this chapter we provide practical tips related to the daily maintenance of your VSAM data sets:

- Reorganization
- VSAM data sharing
- Extended Addressability
- Processing VSAM data sets: Media Manager, OPEN, CLOSE
- Record Management: GET, PUT, EOV routine
- VSAM using real address above 2 GB

4.1 Reorganization considerations

The new DASD storage technology is characterized by:

- ► Numerous high-capacity, small-size FBA, SCSI/SSA disks
- ► Redundancy in different types of RAID
- ► Generous amounts of cache, mainly to avoid the write penalty caused by RAID
- ► Plenty of microcode in order to support RAID, cache algorithms, the mapping between the disks and the logical 3390/3380 volumes (as seen by z/OS), and in the RVA case, the virtualization of the 3390/3380

Because of that, all the old considerations to avoid long seeks and RPS misses in a 3390/3380 logical volume are out of date. Does this apply to VSAM KSDS need of reorganization? Let us look at all the performance reasons (old and new) justifying the VSAM reorganization.

4.1.1 CI/CA splits

Consider CI/CA splits, causing long seeks in the data set — *this reason does not hold true* in the new disk controller scenario. Do *not* reorganize your KSDS or VRRDS data set to avoid long seeks.

Usually, the worst part of the split is when it occurs. After that, the data set has more space for insertions and normally the quantity of splits tend to decrease. Try to avoid splits, if not possible, *do not* reorganize *only* because splits occurred. Monitor the response time and the growing in the number of CA splits. Chances are that, after reorganization, the number of splits will increase.

4.1.2 The loss of useful space in Data CA

The reasons causing this type of waste are:

- ► There is no CA reclaim in VSAM KSDS/VRRDS for those CAs that became free before HURBA. Let's see why in Figure 4-1 on page 205

  In the Index set, each CI has only one record. Each record has a pointer to the highest possible key in an index record in the next lower level, and a pointer to the beginning of that index record.

  The sequence set is the lower level of the index, where the pointers are to data CI. In the sequence set, each index CI contains only one record. Each record governs one CA and has pointers to each used data CI, its higher possible key value (in compressed form) and pointers to free data CI in the data CA.

Then, for example, if at load time an specific data CA was loaded with records belonging to a specific key range and later on the majority of them are deleted (and not re-utilized, as a timestamp key), the CA is underpopulated, and this free space is not reclaimed. The free space pointers are in the index sequence set records, but can be used only by keys in that range.

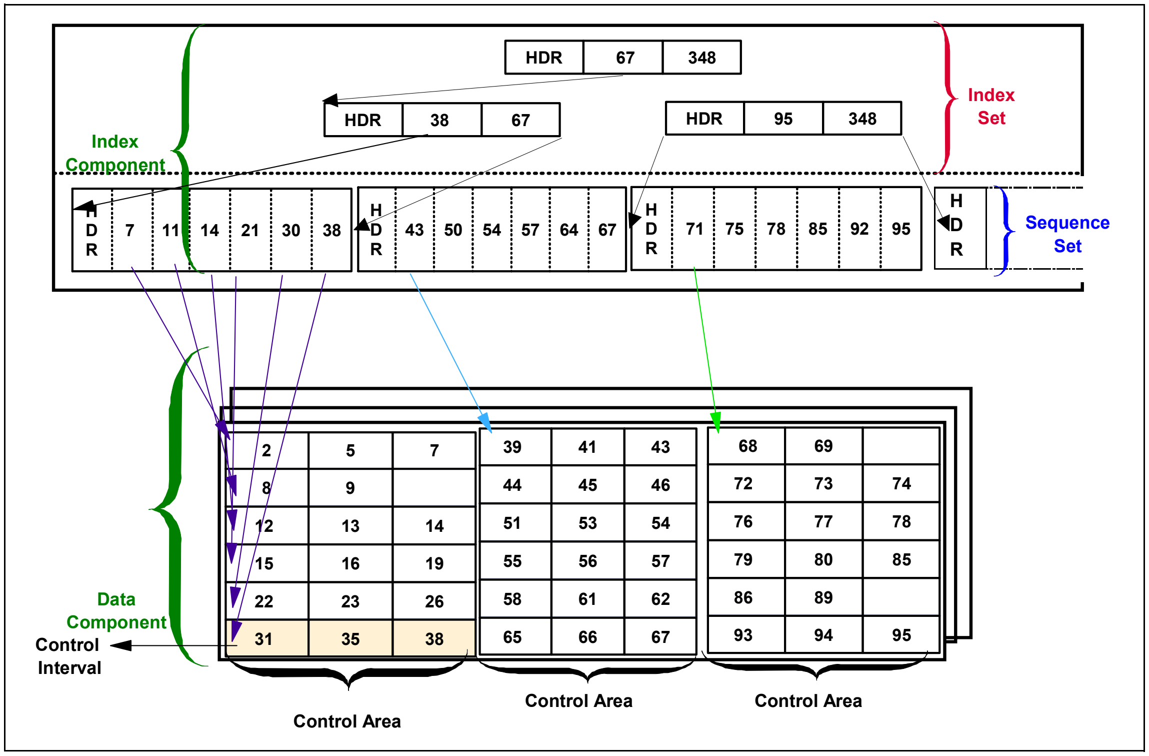


Figure 4-1 Indexes of KSDS

How do we measure that? To answer this question, refer to 3.8, “IDCAMS LISTCAT output fields” on page 187, and the diagram shown in Figure 4-2, as you follow along with our explanation.

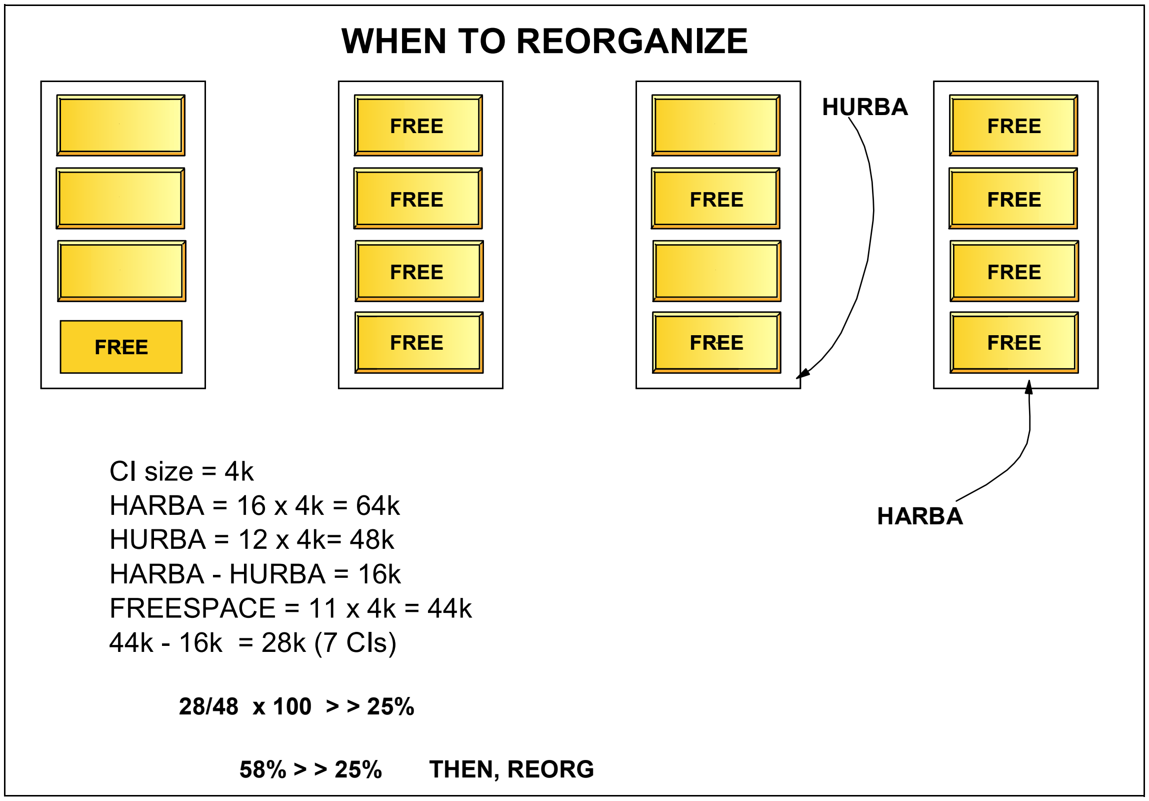


Figure 4-2 HARBA, HURBA, and free space

From this diagram showing HARBA, HURBA, and free space, you can see:

The subtraction, HARBA - HURBA = X, gives the amount of free bytes in the free CIs allocated in the CAs beyond HURBA.

If you subtract FREESPC - X = Y, it gives the amount of free bytes in free CIs embedded in CAs included in HURBA.

Y includes the CA Freespace%, 25%, for example.

If (Y / HURBA) * 100 is consistently greater than CA Freespace% (25%), and the number of deleted records (in catalog) is a large figure, it means that it is time to reorganize due to non-reclaimed CAs.

Here is a numeric example:

```
HI-A-RBA-------184320000 (HARBA)
HI-U-RBA-------176209920 (HURBA)
FREESPACE-%CA---------10
FREESPC---------89616384
```

In view of these figures, is it time for a reorganization?

A strong argument to reorganize is the knowledge that those deleted keys are not going to be inserted again.

- Another reason to reorganize, is when the Index CI size is defined too small and is not big enough to contain information about all the data CIs in the data CA. An example is shown in Figure 4-3. In this case the CA is truncated and the rest is unused. In such a case, you have to redefine the data set with a larger index CI size.

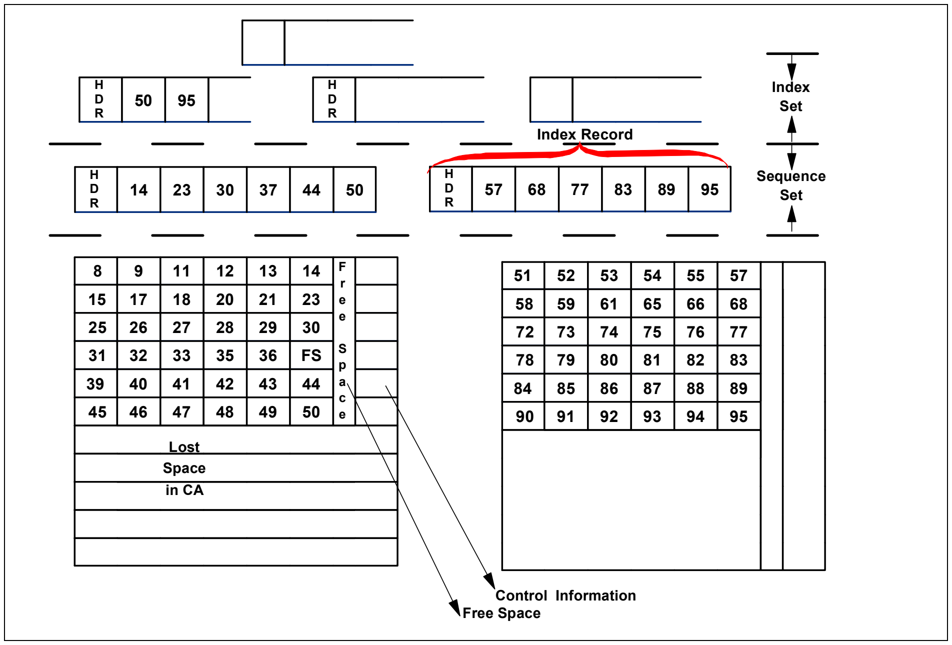


Figure 4-3 Lost space in data CA due to Index CI size too small

How do we measure that? The only way to distinguish between wasted CAs due to smaller index CIs or due to deletions is the previous knowledge of the deletions. Then:

– If (Y / HURBA) * 100 is consistently greater than CA Freespace% and the number of deleted records in the catalog is a small figure, it is time to reorganize, due to the small size of the index CIs. The index CI size must be re-specified as a larger value.

– DFSMShsm issues message ARC0909E advising that is time to reorganize the VSAM data set, due to a free space threshold being reached.

- In z/OS V1 R3, VSAM changed the algorithm to calculate the KSDS and VRRDS index component minimum CI size, when the Index CI size is not informed in DEFINE command. The higher key value in a data CI is stored in compressed format in the Index CI record. The previous algorithm assumed that, after compression, the keys would reduce the size of keys to approximately 5 bytes. With this assumption, for data sets with large keys, the index CI records are bigger and the Index CI size may not be enough to point all Data CIs in a Data CA. In such case, the Data CA is under utilized. Since z/OS V1 R3, VSAM assumes that keys have a compression rate 3:1. Then, for example, a key with 21 bytes, after compression has 7 bytes, and so on.

  For those data sets defined before z/OS V1 R3 and having large keys, you can have data CA under utilized. How do you find these data sets? IBM supplied a tool to help migration to z/OS V1 R3. This IBM tool searches the catalog information for those data sets whose mask is passed in SYSIN DD statement and reports, by data set name, the current index CI size and the new index CI size according to new algorithm.

  For more information about changes in the index CI size calculation, refer to "New Index CI size calculation algorithm" on page 208.

4.1.3 CI/CA splits causing free space increase

After CI and or CA splits the free space per CI (excluding the totally freed) tends to increase. In sequential read processing there is some impact on performance because more free bytes are moved to storage. Refer to 2.7.4, "I/O service time (connect) for VSAM data sets" on page 129.

How do we measure that? By the number of CI and CA splits in the catalog, together with the free space information.

A general comment about CI/CA splits is that their number *usually* grows steady, after reorganization. But do not be surprised if, after reorganization, the number of splits increases. It means that before reorganization, the data set had better CI/CA space for insertions.

4.2 New Index CI size calculation algorithm

Use your automation console function to detect the message in Figure 4-4, generating an alert.

```
IDC3351I ** VSAM I/O RETURN CODE IS 212
Unable to split index; increase index CI size
```

Figure 4-4 Message due to index CI size too small

After that, carefully read this topic.

VSAM uses a default minimum value for index CI size, if it determines that the one specified by you is not large enough. The algorithm that calculates this value was changed in z/OS V1 R3.

This change was to increase the likelihood that the index CI would be large enough to hold all the keys that can exist in a data CA. In a number of situations the default value calculated was not large enough. Because of this, there was a lot of space wasted in the VSAM data set that cannot be used. Also, this forced unnecessary increase in index levels and impacted the performance.

As you know, VSAM compresses the key value when stored the key in the index record. The previous algorithm for determining the minimum size always assumed that all data keys compressed to 5 bytes; in reality they were often much larger. The *new algorithm* assumes a 3:1 compression of the keys, for example that 45-byte keys compress to 15 bytes in the index record. So the newly computed CI sizes will be higher.

This change does not bring impact on existing data sets. But the *new default value* is used for any new VSAM *data set created under z/OS 1.3 or higher*. This also applies to any data sets created because of backup-restore or migrate-recall operations.

The main implication of this change is for applications that open data sets in LSR mode. VSAM data sets opened in LSR mode, might fail with IEC161I 120-053, if the buffer pool created by BLDVRP macro is not large enough to contain the increased CI sizes.

CICS and IMS open VSAM data sets in LSR mode. You may need to review LSRPOOL definition for CICS and DFSVSAMP definition for IMS. Both CICS and IMS use these definitions to determine the size of the buffer pool to be created by BLDVRP. This could lead to database open failures or poor performance.

If you are using VSAM RLS, there is not such issue because in RLS, SMSVSAM builds it's own pools for data sets.

If you are letting CICS calculate the number of buffers in your LSR buffer pools, then you should have no problem. If you are explicitly coding the number of buffers of each size in the LSRPOOL then you should review the number of buffers you specify.

4.2.1 Analyze existing data sets

You need to analyze your VSAM data sets and compare the current index CI size with the new value computed by z/OS V1 R3. We recommend you specify CI sizes for both index and data components of your VSAM data sets and compute the new value for the index CI size. If you do not do it, the index CI size of your VSAM data sets will change to the new value when they get redefined.

Sample program to identify LSR mode data sets

You need to identify all applications that open VSAM data sets in LSR mode and review the possibility of change in index CI size. To identify data sets opened in LSR mode, you can use the SMF 62 and 64 records. The fields SMF62MC3 and SMF64MC3 hold a one byte value. If the byte is in the format B'x1xxxxxx' then this indicates the data set was opened or closed in LSR mode. These records also include the data set name and jobname. You may also use the sample code In the “SMFLSR sample program” on page 380, to generate a list of data sets opened in LSR mode.

IBM CI sizer tool

IBM provides a tool to analyze your existing VSAM data sets. This tool gives a report showing:

- Data set name
- Current index CI size
- Minimum (default) index CI size
- Data CI size
- Data set creation date

See Figure 4-5 for a sample output of this tool.

```
                        LISTING FOR FILTER KEY: U705040.**
 DATA SET NAME                                           CURRENT  PRE-1.3   1.3+
DATA CI   CREDT
 U705040.DDIR                                              512      512     2560
18432 2001.114
 U705040.INFI.SDIDS                                       2560     2560     3072
2048 1989.278
 U705040.INF0420.SDIDS                                    4096     4096     5120
1024 1991.260
       THE NUMBER OF ENTRIES PROCESSED WAS:
                TOTAL CLUSTERS:           24
                 TOTAL AIX:                0
                KSDS PROCESSED:           20
                INDEX CISIZE CHANGE:        3
                SKIPPED DUE TO ERRORS:      0
                SKIPPED (OFFLINE):         0
                SKIPPED DUE TO FIELDS:       0
```

Figure 4-5 Sample output of the CISIZE tool

The tool name is INDXCISZ and can be downloaded to your MVS system from IBM FTP site as follows:

- Allocate a data set with LRECL=1024,BLKSIZE=6144,RECFM=FB, DSORG=PS and RECFM=FB.
- Enter the TSO ftp command.
- At the Connect to? prompt, enter: `ftp.software.ibm.com`
- At the NAME prompt, enter: `anonymous`
- At the PASSWORD prompt, enter your e-mail address.
- At the Command prompt, enter: `cd s390/mvs/tools`
- At the Command prompt, enter: `binary`
- At the Command prompt, enter: `get INDXCISZ.JCL.TRSD 'your data set’`
- Once the file is placed on your MVS system, you need to unterse it.
- You need to use TRSMAIN program to unterse the file.
- The TRSMAIN program can be downloaded from:

  `http://techsupport.services.ibm.com/390/trsmain.html`

You can also download it using a batch job. For a sample job to download the tool and unterse it, see Figure 4-6.

```
//MHLRES1E JOB (999,POK),MSGLEVEL=1,NOTIFY=MHLRES1
//ALLOC   EXEC PGM=IEFBR14
//SYSPRINT DD  SYSOUT=(*)
//DD1T     DD  DISP=(NEW,CATLG,KEEP),SPACE=(CYL,(2,1),RLSE),
//             DCB=(LRECL=1024,BLKSIZE=6144,RECFM=FB),UNIT=SYSDA,
//             DSORG=PS,
//             DSN=MHLRES1.INDXCISZ.TERSED
//DD1U   DD  DISP=(NEW,CATLG,KEEP),SPACE=(CYL,(2,1),RLSE),UNIT=SYSDA,
//             DSORG=PS,
//             DSN=MHLRES1.INDXCISZ
//FTP     EXEC PGM=FTP,REGION=4096K,
//             PARM='FTP.SOFTWARE.IBM.COM  (TIMEOUT 720 EXIT'
//SYSMDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//INPUT    DD  *
ANONYMOUS
xxxx@yyy.zzz.com
binary
CD /s390/mvs/tools/
dir
binary
get  INDXCISZ.JCL.TRSD 'MHLRES1.INDXCISZ.TERSED' (replac
quit
/*
//TRSMAIN EXEC PGM=TRSMAIN,PARM='UNPACK',TIME=1440
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//INFILE   DD  DISP=SHR,DSN=MHLRES1.INDXCISZ.TERSED
//OUTFILE  DD  DISP=SHR,DSN=MHLRES1.INDXCISZ
```

Figure 4-6 Job to download the CISIZE tool

The untersed file contains instructions for link editing and executing the tool. It will tell you the current CI size and recommended CI size for your selected VSAM data sets.

4.3 Sharing VSAM data sets

In this section we discuss the sharing of VSAM data sets from one address space throughout a Parallel Sysplex® and what that it implies, such as level of data integrity, MVS enqueues, VSAM mechanisms to protect data integrity, and SHAREOPTIONS.

To protect the integrity of your VSAM data sets, VSAM uses internal locks and issues ENQs in SYSVSAM major name, which invokes the MVS Enqueue

Manager routine to serialize the resource locally. To serialize the full cluster across systems, it is required to have the Global Resource Serialization (GRS) or an equivalent product with the caution of *not placing* SYSVSAM resource name in RNL exclusion list.

Some of the consequences of placing SYSVSAM in the exclusion list include: loss of records, overlaid records, duplicate records, invalid index structures, invalid catalog records, invalid data set dumps, backups, restores, or migrations, incorrect extends of the data set, and incorrect release of space in the data set.

The major name SYSVSAM is used to serialize lots of resources in VSAM, Here are some examples:

- Enforce VSAM SHAREOPTIONS.
  - VSAM no longer determines whether any user has a data set open on other systems.
- Serialize OPENS, CLOSES, or EOVs.
  - Multiple OPENS, CLOSES, or EOV requests can now occur at the same time on multiple systems.
  - Multiple extends of a data set can now occur on multiple systems.
- Serialize dynamic string addition.
  - It is no longer possible to serialize dynamic string additions with other dynamic string additions and with OPEN/CLOSE/EOV.
- Serialize catalog updates.
  - A user can no longer determine if the data set is open on another system so that the catalog can be correctly updated.
  - Users cannot correctly serialize catalog updates of data set compression tokens.
- You cannot correctly determine the status of a data set for DELETE and RENAME processing.
- HSM cannot correctly serialize migrations with VSAM OPEN/CLOSE/EOV.
- Serialize DFDSS dump/restores. DSS cannot correctly serialize DUMP/BACKUP/RESTORE with VSAM OPEN/CLOSE/EOV.
- Partial release. VSAM CLOSE cannot determine when the last user for a data set is closing the data set in order to correctly implement PARTIAL RELEASE.
- Recognize close failures in order to warn the user.
  - VSAM OPEN cannot determine whether the data set was improperly closed.
  - May not warn users of a close failure.

  – May not attempt to determine correct end of the data set.

- Support VSAM Record Level Sharing (RLS). Unable to enforce RLS/non-RLS sharing rules.

Returning to sharing, you can share VSAM data sets between:

- Many task programs running in different address spaces, in a single operating system. Some tasks may be running the same program, accessing the same VSAM data set. For example, data set Teka in Figure 4-7.
- Many task programs running in the same address space. Multiples ACBs pointing to same VSAM data set, like data set Tita, in Figure 4-7.
- One ACB shared in a task or different subtasks, in the same address space, as an example, with multistring processing.
- Different tasks in different z/OS images. As an example, data set Prica in Figure 4-7.

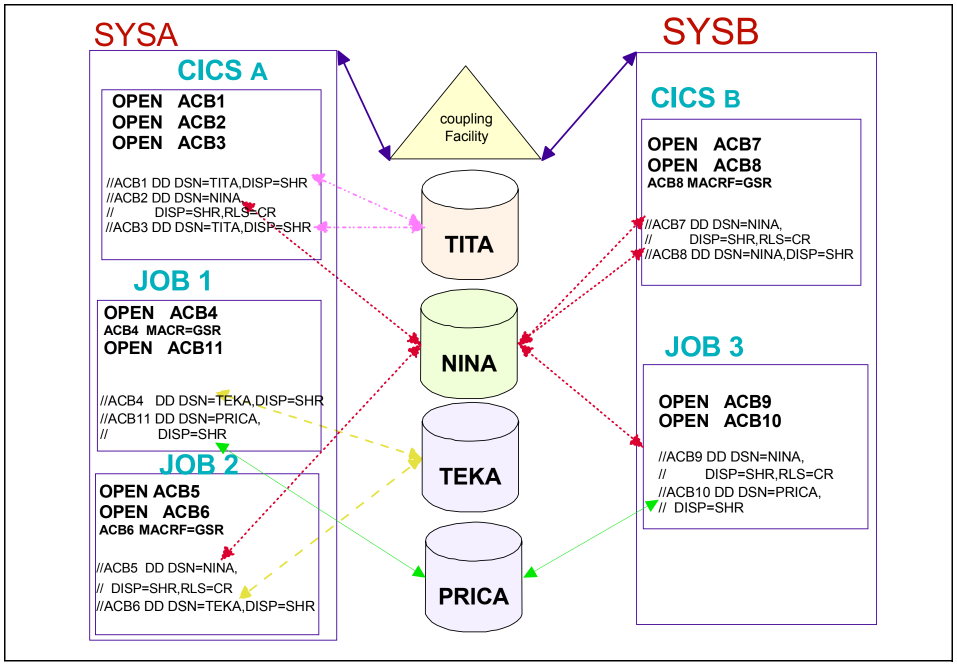


Figure 4-7 Sharing VSAM data sets in the Parallel Sysplex

The Enqueue Manager is an MVS component. It does not care about the nature of the resource but only about its symbolic name, that is formed by its Major (also known as Qname) and Minor name (Rname). The serialization is done in the resource symbolic name and the type of serialization indicates the protection of the resource:

- When SHARED, any other shared access is permitted and any exclusive access is denied or queued until the resource becomes available.
- When EXCLUSIVE, the Enqueue Manager guarantees only the task owning the resource can use it. Any other request to the same resource, no matter if it is exclusive or shared, it is denied.

The scope of the enqueue indicates if the resource is serialized at address space level (scope STEP), at system image level (scope SYSTEM) or sysplex wide (scope SYSTEMS), which implies GRS.

Before you define the level of sharing for a VSAM data set, you must evaluate the consequences of reading incorrect data (a loss of read integrity) and writing incorrect data (a loss of write integrity). These are situations that can result when one or more of the data set's users do not adhere to guidelines recommended for accessing shared data sets. On the other hand, it is important to avoid the unnecessary use of certain serialization functions which may cause a performance degradation.

4.3.1 Write and read integrity

When you share a VSAM data set in the same task program, or among task programs running in same or different address spaces, from the same or different z/OS images, there is a need to inform what type of integrity (read and/or write) is required.

Write integrity

Write I/O operation should guarantee integrity for non-atomic writes (the writes executed in multiple I/O operations), for example:

- Logical record update in-place, where a record is read, updated in memory and written back
- CI index updates due to CI data insertions and deletions (KSDS/VRRDS)
- CI and CA splits, which involves several write I/O operations
- Data set and catalog (usually VVDS) synchronous updates
- Debit and credit writes program

To have write integrity, when several tasks are accessing the same data set, VSAM (and you) must guarantee that all the non-atomic writes are executed without being pre-empted. This means that no other related intervening write I/O operation should be executed.

Read integrity

Read integrity guarantees that the record you read is the most current copy available. It implies:

- Within non atomic writes, the related read I/O operation is suspended.
- The read I/O operation is able to find the most current copy of the logical record, meaning:
  - If the data is in the buffer pool, VSAM must guarantee the most current copy to satisfy the read (also called buffer pool coherency). In a shared environment with different z/OS images accessing the data set, there are two ways of doing this: through Parallel Sysplex RLS cross invalidation or by VSAM refreshing the buffer at every read request (specified with SHAREOPTION 4 which we discuss later).
  - The write operation must send the current copy to where the next read can access it, even in a multiple z/OS image system (to shared DASD or with Parallel Sysplex RLS, to the coupling facility).

4.3.2 VSAM sharing mechanisms

Before talking about the sharing mechanisms, let us introduce the concept of the VSAM control block structure.

The VSAM control block structure is formed by the input/output buffers and a set of VSAM control blocks, *created at open time*, that holds informations related to the data set being opened, like space available, type of VSAM data set, buffering technique, and so on. The control block structure is deleted when the last close of the data set is issued on this system.

A VSAM control block structure can be shared by more than one OPEN of the same VSAM data set. To be shared, the structure must be *compatible and accessible*.

To understand the meaning of *compatible*, let's see what happens when a VSAM data set is opened. To open a VSAM data set, an ACB is used and some information about the ACB goes to the VSAM control block structure. For a new ACB to use an existing VSAM control block structure, the new ACB must be compatible with the existing control block structure. That means it must be consistent in its specification of the following processing options:

- Data set specifications: For example, a VSAM control block structure created when opening an index of a KSDS data set as an ESDS is not consistent with a control block structure as when opening a KSDS data set such as a KSDS.
- The ACB MACRF options: DFR, UBF, ICI, CBIC, LSR, and GSR must be consistent.

For more information about ACB, refer to “ACB control block” on page 235.

Now, let us see how a VSAM control block structure can be *accessible*. To access a VSAM control block structure depends on it is located:

- When using NSR or LSR, the VSAM control block structure is located in the address space private area and can be accessed by tasks running programs in the same address space.
- For GSR, the VSAM control block structure is located is Common Storage Area (CSA) and can be accessed by task running programs in other address spaces in the same system image.

  For more information about NSR, LSR, GSR and RLS refer to Chapter 4, “Managing your VSAM data sets” on page 203.

At open time, before creating a control block structure, VSAM verifies if already an *accessible* and *compatible* VSAM control block structure of the data set exists:

- If so VSAM uses the same control block structure and the serialization to guarantee data integrity in done at CI level.
- If one does not exist, VSAM creates the control block structure for the data set. Also VSAM uses the SHAREOPTIONS value to determine the level of data set sharing to implement protection through enqueue (ENQ) in SYSVSAM.
- If it already exists in the Parallel Sysplex, it is not compatible or not accessible, VSAM use the SHAREOPTIONS value to determine whether creating a new control block structure would violate the SHAREOPTIONS rules. If so, the OPEN fails, otherwise the OPEN is successful.

VSAM verifies if a control block structure exists in the Parallel Sysplex testing enqueue in SYSVSAM to resources representing input and output control blocks VSAM data set components.

VSAM uses two basic implementations to control data sharing:

- VSAM lock mechanism controls data sharing in a single control block structure. VSAM has two lock mechanisms according to the mode being used: RLS and non-RLS.
- Enqueue in SYSVSAM name, issued by VSAM, for resources representing input and output VSAM control blocks for the data set components, based on data set SHAREOPTIONS.

The VSAM control block structure plays a decisive role in VSAM data sharing. When the sharing of a VSAM data set is done using a *single and only* VSAM control block structure, VSAM guarantees *read and write integrity* at CI level, independent of the SHAREOPTIONS or DISP specifications.

When the data set is accessed throughout different VSAM control block structures at the same time, VSAM *cannot ensure* who has exclusive use of the data because the *control block structure is not the same*. In this case VSAM applies SHAREOPTIONS rules, serializing at data set component level.

When the SHAREOPTIONS being used do not protect the data set, the user application programs must provide data set integrity by the use of ENQUEUE or RESERVE macros.

There are five important ways to implement VSAM data set integrity according to how the data set is shared:

- Internal locks for non-RLS access, when using a single VSAM control block structure.
- Global lock for RLS access, a very sophisticated mechanism using the coupling facility. RLS can be used when sharing a data set throughout the Parallel Sysplex. SHAREOPTIONS does not apply to VSAM data sets in RLS mode because it uses a single control block structure through the sysplex. About sharing VSAM data sets in RLS mode refer to Chapter 5, "VSAM Record Level Sharing" on page 243.
- SYSVSAM ENQ issued by VSAM, according to the data set SHAREOPTIONS, for non-RLS access having multiple VSAM control block structures.
- The ENQ serialization function issued by the data set allocation routine, according to data set disposition informed. The Qname is SYSDSN and the Rname is the data set name. The scope of the serialization is sysplex wide and the resource Status is shared for SHR data set disposition and exclusive for NEW, OLD or MOD disposition.
- The ENQ serialization function can also be implemented by the user application program.

4.3.3 Sharing data in a single VSAM control block structure

When the sharing of a VSAM data set is done using a *single and only* VSAM control block structure *throughout the Parallel Sysplex*, VSAM guarantees *read and write integrity* independent of the SHAREOPTIONS or JCL DISP specifications.

Table 4-1 presents the way you can implement VSAM data sharing with read and write integrity in the Parallel Sysplex.

Table 4-1 One VSAM Control Block Structure

Buffering Technique	**One Control Block structure in the Parallel Sysplex**
NSR/LSR	All OPENs to data set issued in the same address space
GSR	All OPENs to data set issued in the same system image
RLS	Anywhere

With NSR or LSR, many jobs concurrently OPEN to same data set can use only one VSAM control block structure, since issued by task(s) in the *same address space*.

The ability to perform multiple OPENs to the same data set within a task or from different tasks in a single address space sharing *a single VSAM control block structure* is called inter-address space sharing, also referred as subtask sharing. Subtask sharing allows many logical views of the data set while maintaining a single VSAM control block structure. With a single control block structure, you can ensure that you have exclusive control of the buffer when updating a CI. An example of the use of single VSAM control block structure is in Figure 4-7 on page 214, the data set “Tita”, in CICS-A, is accessed by ACB1 and ACB3, both pointing to same data set.

When using GSR, the same VSAM control block structure can be used by *address spaces in the same system image*, because the structure is in the CSA. In Figure 4-7 on page 214, JOB 1 and JOB 2 are using GSR in the same system image, accessing the same data set. They are using the same VSAM control block structure for access the data set “Teka”, that is the only VSAM control block structure in the Parallel Sysplex. Before using GSR, refer to “Global Shared Resources (GSR)” on page 80, for its disadvantages. We strongly recommend you to use RLS instead of GSR.

Let us see how VSAM guarantees read and write integrity, in multiple string processing. In this processing environment there can be multiple independent and concurrently requests (RPLs) for the same data set, using a single VSAM control block structure:

- When your program issues a GET nonupdate request, VSAM first tries to locate the control interval in buffers attached to its string and obtain a copy of the CI (with NSR) or shares the same copy in the buffer (LSR). If VSAM does not find in buffers, it reads the control interval from DASD.
- For GET update requests, the CI is obtained in exclusive control, through VSAM lock mechanism, and then read the control interval from the device for the latest copy of the data. If the buffer is already in exclusive control of another string, the request fails with an exclusive control feedback code. The rules for exclusive control are:

a. If a given string obtains a record with a GET for update request, the control interval is not available for update or insert processing by another string.

b. If a given string is in the process of a control area split caused by an update with length change or an insert, that string obtains exclusive control of the entire control area being split. Other strings cannot process insert or update requests against this control area until the split is complete.

An exclusive control conflict happens when a request is made for a resource that is in exclusive control of an other request. VSAM treats in two different ways, according buffering mode being used:

- With LSR or GSR, VSAM handles the request according to the information in the MACRF parameter in ACB:
  - LEW: The request is queued, placing the requiring task in wait until the resource become available. This is the default.
  - NLW: VSAM returns exclusive control error return code X'14' to the application program, then the application program is able to determine the next action.
- With NSR, VSAM does not queue requests that have exclusive control conflicts. VSAM returns a logical error return code, and you must stop activity and clear the conflict. If the RPL that caused the conflict had exclusive control of a control interval from a previous request, the application program issue an ENDREQ before attempting to clear the conflict and avoid a dead lock. The Conflict can be solved in one of three ways, at application program level:

  a. Queue until the RPL holding exclusive control of the control interval releases that control and then reissue the request.

  b. For Assembler language, issue an ENDREQ against the RPL holding exclusive control to force it to release control immediately.

  c. When using LSR and Assembler source program, release the buffer issuing MRKBFR MARK=RLS.

  If the RPL includes MSGAREA and MSGLEN, the address of the RPL holding exclusive control is provided in the first word of the MSGAREA. The RPL field, RPLDDDD, contains the RBA of the requested control interval.

Sometimes this locking in a CI basis done by VSAM may cause contention and even deadlocks. There are no deadlock detection and prevention algorithms implemented in VSAM, except in an RLS environment. If you are facing these drawbacks a recommendation is to have less logical records in a data control interval, or in other words to have better lock granularity.

Using a single VSAM control block structure

The three methods of achieving a single VSAM control block structure for a VSAM data set while processing multiple concurrent requests are:

1. A single access method control block (ACB) and a STRNO>1. Refer to *z/OS DFSMS Using Data Sets,* SC26-7410, to get more information about STRNO.
2. DD name sharing: When multiple ACBs, all and located in the same address space, pointing to a single DD statement. You have this when multiple tasks, in the same address space, are executing the same program concurrently or as in the example below, in assembler language, multiple tasks executing different programs, but pointing to same DD:

Example 4-1 DD name sharing pointing to a single DD statement

```
JCL:
//DD1 DD DSN=ABC

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD1
```

3. Data set name sharing, with multiple ACBs pointing to multiple DD statements with different DDnames, but *with the same DSname.* The data set names are related with an ACB open specification (MACRF=DSN). This MACRF option means that subtask shared control block connection is based on common data set names. For example:

Example 4-2 DD name sharing with different DD names

```
JCL:
//DD1 DD DSN=ABC
//DD2 DD DSN=ABC

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1,MACRF=DSN

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD2,MACRF=DSN
```

Now, talking about data sets with AIX, VSAM connects an ACB to an existing VSAM control block structure for data set name sharing *only* when the base of the sphere is the same for both ACBs. In Example 4-3, supposing no VSAM block structures exist:

- When task A issues OPEN to the cluster, VSAM creates block structure for the cluster and for the alternate index. VSAM knows by the catalog the name of the AIX.
- When task B issues OPEN to the PATH, VSAM adds to existing structure for the cluster and the alternate index, because the base of the sphere is the same, the cluster.
- When task C issues OPEN to the alternate index, VSAM creates new control block structure for the alternate index. VSAM does not add to existing structure as the base of the sphere is not the same. SHAREOPTIONS are enforced for AIX_A data set name since multiple control block structures exist.

Example 4-3 Two VSAM control blocks structure to same data set

```
JCL:
//DD1 DD DSN=CLUSTER_A
//DD2 DD DSN=PATH CONNECTING AIX_A TO CLUSTER_A
//DD3 DD DSN=AIX_A

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1,MACRF=DSN

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD2,MACRF=DSN

task C executing PGM3:
PGM3:      OPEN ACB3
      ACB3 ACB DDNAME=DD3,MACRF=DSN
```

To a new ACB use an existing VSAM control block structure, the new ACB must be compatible with the existing control block structure, if compatibility cannot be established, OPEN tries (within the limitations of the share options specified when the data set was defined) to build a new control block structure. If it cannot, OPEN fails.

When implementing intra-address space sharing, it is necessary that when attaching new subtasks, the *subpool zero must be shared* by mother and daughter tasks in order to shared the RP control blocks (SZERO=YES in the ATTACH macro).

A subtask has an opened ACB which shares a control block structure that can have been previously used. If this subtask now issues the POINT macro to obtain the position for the data set, it should not be assumed that positioning is at the beginning of the data set, as in a more normal situation.

When VSAM can not use an existing control block structure for a data set, VSAM uses the SHAREOPTIONS. Refer to "VSAM SHAREOPTIONS" on page 223.

4.3.4 Sharing data with many VSAM control block structures

In this section we discuss about sharing data sets throughout the Parallel Sysplex when more than one VSAM control block structure can exist. This happens when an OPEN to a data set cannot use an existing VSAM control block structure.

VSAM SHAREOPTIONS

VSAM SHAREOPTIONS play an important role in VSAM integrity. They specify whether and to what extent data is to be shared among tasks in one or multiple z/OS task programs the Parallel Sysplex. When you define VSAM data sets, you specify how the data is to be shared through the use of SHAREOPTIONS parameter of DEFINE command.

The share options are specified as `SHAREOPTIONS(x,y)`. Improperly stated, *x* and *y* are referred as cross-region and cross-system, respectively. However, *share options are sysplex wide.* We keep using the names cross-region, cross-system to avoid confusion, but keep in mind that they do not mean that.

The sharing is controlled throughout VSAM control block structures. VSAM uses the SHAREOPTIONS values to determine whether creating a new control block structure would violate the SHAREOPTIONS rules. If so, the OPEN fails, otherwise the OPEN is successful.

VSAM uses enqueue to implement the level of sharing requested in SHAREOPTIONS, when creating a new control block structure at OPEN time. VSAM issues enqueues to the major name SYSVSAM.

When a cluster has an alternate index:

- If the data set opened is the *path or the cluster*, VSAM creates control block structure for:
  - The cluster components. For example, if the data set is KSDS, VSAM creates control block structure for the data and for the index component.
  - The alternate index components.
- If the data set opened is the alternate index, VSAM issues enqueue *only* to the alternate index components (for example, data and index).

For each VSAM control block structure created, VSAM issue enqueue to enforce SHAREOPTIONS.

When the VSAM data set is empty and is being open for OUTPUT, no matter the SHAREOPTIONS, *only one* VSAM OUTPUT control block structure can exist throughout the Parallel Sysplex, VSAM process the data set as having SHAREOPTIONS(1,x).

When a data set is allocated with OLD disposition, VSAM treats the data set as having SHAREOPTIONS(1,x).

Cross-region options

The meanings of the options listed below are valid also in the Parallel Sysplex.

(**1**): This SHAREOPTIONS means that can be *only one* VSAM OUTPUT control block structure throughout the Parallel Sysplex *or any* VSAM INPUT control block structures in the Parallel Sysplex.

Except for those OPENs using the same VSAM control block structure, VSAM ensures only *one* OPEN *for output or many* OPENs *for input only*. Any other OPEN fails with a return code in ACB. That means, if the data set is open for input, other OPEN creating a new control block structure can open the same data set for input successfully. If the intention to open the data set is for output, the open fails with a return code in ACB. In other words, VSAM is insuring total *read and write integrity*. The intention of input or output is declared in the ACB MACRF parameter, not in the OPEN input or output options.

For *output*, VSAM issues exclusive enqueue for the input and output VSAM control blocks structure for each VSAM component, guaranteeing just one control block structure in the Parallel Sysplex. If an other OPEN is issued against the same data set, not using the same control block structure, VSAM does not queue the open, but fails it.

For *input*, VSAM issues shared enqueue only for input VSAM control blocks structure for each VSAM component. OPENs for input are allowed. If an OPEN for *output* is issued against the same data set, not using the same control block structure, VSAM does not queue the open, but fails it.

(**2**): VSAM ensures that only *one* VSAM control block structure for *output* in the Parallel Sysplex *and many* VSAM control block structures for *input*. In other words, VSAM is insuring write integrity. If you require read integrity (with a better performance due to higher granularity than cross-regions (1), it is your responsibility to use the ENQ and DEQ macros appropriately to provide read integrity for the data the program obtains.

VSAM fails the OPEN OUTPUT when already exist in the Parallel Sysplex an output VSAM control block structure to the same data set.

(**3**): VSAM allows *many* OPENs to the data set for *output and/or input*. The user application programs must ensure both read and write integrity through their own ENQs (including Open and Close processing on them).

(**4**): VSAM allows *many* OPENs to the data set for *output and/or input*. The user application programs must ensure both read and write integrity through their own ENQs (including Open and Close processing on them). VSAM refreshes the *data and index* components buffer pools for direct processing (the sequence set is not refreshed), to guarantee the coherency of the data in the buffer pool. Coherency in this case means that the task gets the most updated contents of the requested record.

Cross-system options

As we said before, the name cross-system is improper stated, since SHAREOPTIONS is sysplex wide. Cross-system options provide the same capability as cross-region 3 and 4. So, they are just a way to provide data sets using cross-region options 1 and 2 with the capabilities of 3 or 4. That means, you can use cross-systems options to have SHAREOPTIONS 1 or 2 with (4) or without (4) buffers refreshment. The cross-system value can be:

(**3**): VSAM does not refresh the buffer pools for direct processing. When `SHAREOPTIONS(3,3)` or `SHAREOPTIONS(4,3)` is used, VSAM provides a Control Block Update Facility (CBUF).

CBUF is active, whenever a data set is opened with `DISP=SHR`, and `SHAREOPTIONS(3,3)` or `SHAREOPTIONS(4,3)`. In this case, VSAM record management maintains a copy of the critical control block data in z/OS common storage. Obviously, this common storage is available only to address spaces within your operating system. Passing this information to another operating system is your responsibility. CBUF eliminates the restriction that prohibits control area splits. However, under share options 4 these restrictions still exist.

Cross-systems sharing with CA splits can be accomplished (with integrity) by sending the VSAM shared information (VSI) blocks to the other host at the conclusion of each output request. Every time a data set is opened on a system for CBUF processing, a VSI is built for the data set and added to the VSI chain. This control block is then updated by the user to communicate information from one address space to another. Generally, the VSIs sent to other z/OS images has not changed and only a check occurs. Refer to the Appendix , “Accessing the VSAM Shared Information (VSI)” on page 366, where there is an example about how to get the VSI.

About sending the VSI to the other z/OS image, you may choose:

- XRC APIs, as: IXCCONN, IXCMSGO, IXCMSGI
- APPC VTAM® LU 6.2, as: RECEIVE_AND_WAIT and SEND_DATA

Remember that you still must continue to provide read and write integrity. Although VSAM ensures that tasks have correct control block information if serialization is done correctly. Also, the option 3, does not cause buffer pool invalidation as in option 4.

Because programs in many regions can share the same data set, an error that occurs in one region can affect programs in other regions that share the same data set. If a logical error (register 15=8) or physical error (register 15=12) is detected, any control block changes made before the error was detected will be propagated to the shared information in common storage.

The section “Techniques of Data Sharing” in *z/OS DFSMS Using Data Sets,* SC26-7410, contains examples about how to implement VSAM sharing with integrity.

4.3.5 General share options: Considerations

User tasks running user programs that ignore the write integrity guidelines in share options 3 and 4 can cause VSAM program checks, lost or inaccessible records, uncorrectable data set failures, and other unpredictable results. This option places responsibility on each user application program sharing the data set. Refer to 3.3.4, “Broken VSAM data set” on page 163.

User programs that ignore the read integrity guidelines in share options 2, 3 and 4 results in down-level data records and erroneous no-record-found conditions.

As stated before in cross-region option 4 and cross-system option 4, buffers for direct processing are refreshed by VSAM for each request (or in other words, buffering is not saving I/O operations). Here are more details on this:

- Each PUT request results in the appropriate buffer being written immediately into the VSAM object's DASD. VSAM writes out the buffer in the user's address space that contains the new or updated data record. The data and sequence-set control interval buffers are marked invalid following the I/O operation to DASD.
- Each GET request results in all the user's input buffers being refreshed. The contents of each data and index buffer used by the user's program is retrieved from the VSAM object's direct access device.
- GRS serialization

  Open/Close/EOV routines use ENQ/DEQ in SYSVSAM.dsn.catname.I|O|B, to implement serialization when processing a VSAM data set, as well as to ensure proper sharing based on share options. To avoid integrity exposures, *never* add SYSVSAM Qname to the GRS RNL exclusion list.

- Pay attention that during load mode processing, you cannot share data sets. Share options are overridden during load mode processing to (1 3). Refer to 2.6.11, “Initial load option” on page 59.

  Data and sequence sets buffers for direct processing (for reads and writes) are refreshed for each request (index buffers are not). Output processing is limited to update and add processing that does not change either the high-used RBA or the RBA of the high key data control interval. Then, control area splits and the addition of a new high-key record for a new control interval that results from a control interval split are not allowed. VSAM returns a logical error to the user's program when this condition occurs. If the task running your program does not satisfy the requirements described above, you require cross-system option 3, where due to CBUF, CA splits and addition of a new high-key record are allowed.

Table 4-2 contains a summary of the share options.

Table 4-2 Relationship between share options and VSAM functions

Share Options (DISP=SHR)	VSAM function provided
(3 3)	CBUF and no buffer refresh
(3 4)	Data/Seq Set buffers invalidate. No CA splits
(4 3)	Data/Index buffers invalidate. CBUF
(4 4)	Data/Seq Set/Index buffers invalidate. No CA splits

4.3.6 Protecting VSAM data set through DISP parameter

When a data set is allocated, the data set disposition provokes an enqueue in the major name SYSDSN, minor name data set name. The scope of the enqueue is throughout the Parallel Sysplex (unless the data set name is the GRS exclusion list). The enqueue is SHARED for shared data set disposition and EXCLUSIVE for OLD, MOD or NEW data set disposition.

If you intend to protect your VSAM data sets using its disposition, be aware of:

- The enqueue is done only for the data set name, not for the sphere components.
- When OLD data set disposition is used, VSAM treats the data set as having SHAREOPTIONS(1,x).

In Example 4-4, the enqueues in SYSDSN issued due to allocation are in two different resource names: PATH_OF_CLUSTER_A, and CLUSTER_A. The OLD disposition does not avoid allocating CLUSTER_A in JOB 02. But with the OLD disposition in JOB 01, VSAM treats as SHAREOPTIONS(1,x). Then, while JOB 01 is running with DD1 opened, if JOB 02 starts and tries to open the data set, it receives and OPEN error, because already exists one RP control block for OUTPUT.

Example 4-4 VSAM data set allocated with DISP=OLD

```
SPHERE: CLUSTER_A (Base of the sphere)
        AIX_OF_CLUSTER_A
        PATH_CLUSTERA (Connecting the alternate index to the base cluster)

JOB 01 OPEN FOR OUTPUT:
//DD1 DD DSN=PATH_OF_CLUSTER_A,DISP=OLD

JOB 02:
//DD2 DD DSN=CLUSTER_A,DISP=SHR
```

Before using DISP=OLD to protect VSAM data sets, refer to “Using a single VSAM control block structure” on page 221, to understand how VSAM creates control block structure according to the component being opened. If you intend to use DISP=OLD, do not forget to allocate the VSAM data set components with DISP=OLD too.

4.4 Extended addressability (EA)

Extended addressability (EA) was introduced in DFSMS/MVS 1.3, for KSDS data sets. Since DFSMS/MVS 1.4, EA is supported in record level sharing (RLS). With DFSMS/MVS 1.5, support for extended addressability is extended to all other VSAM record organizations.

With EA, the 4G architectural limit for data set size imposed by using the 4-byte field for the relative byte address (RBA) was eliminated.

It is important to state that extended addressability and extended format are not the same concept. Extended format is a way of storing data in a 3390/3380 logical volume. Extended addressability is the ability of allowing larger VSAM data sets. However, extended format is a pre-prerequisite for extended addressability.

Using EA, the size limit for a VSAM data set is determined by either:

- CI size multiplied by 4 GB
- The volume size multiplied by 59

A 4K CI size yields a maximum data set size of 16 TB, while a 32 KB CI size yields a maximum data set size of 128 TB. A 4K CI size is preferred by many applications for performance reasons. No increase in processing time is expected for extended format data sets that grow beyond 4 GB.

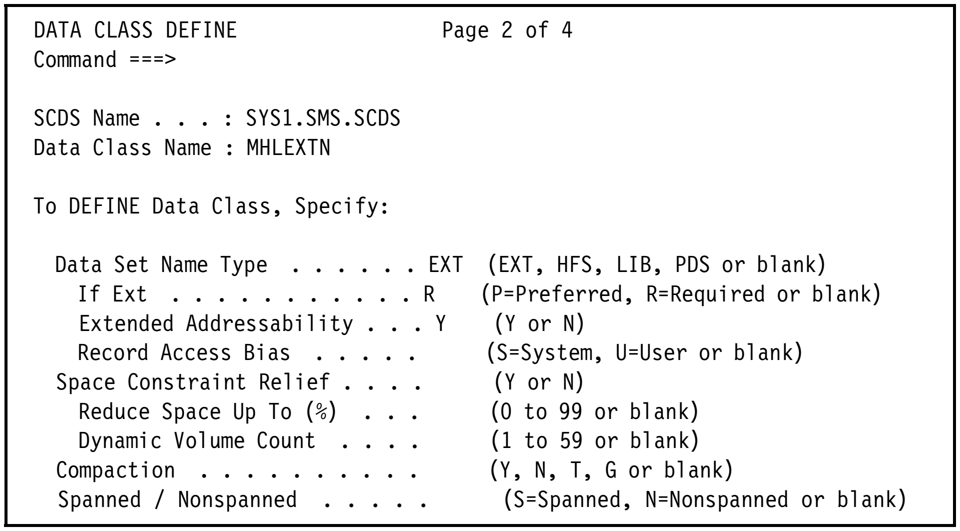
To use EA, the data set *must* be:

- SMS-managed
- Defined as extended format

EA is available to a data sets associated to a data class defined with:

- DSNTYPE=EXT
- EXTENDED ADDRESSABILITY=Y

Figure 4-8 shows the ISMF panel where you specify EF and EA when you define or alter a data class.



```
DATA CLASS DEFINE                 Page 2 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : MHLEXTN

To DEFINE Data Class, Specify:

  Data Set Name Type  . . . . . . EXT  (EXT, HFS, LIB, PDS or blank)
    If Ext  . . . . . . . . . . . R    (P=Preferred, R=Required or blank)
    Extended Addressability . . . Y     (Y or N)
    Record Access Bias  . . . . .      (S=System, U=User or blank)
  Space Constraint Relief . . . .      (Y or N)
    Reduce Space Up To (%)  . . .      (0 to 99 or blank)
    Dynamic Volume Count  . . . .      (1 to 59 or blank)
  Compaction  . . . . . . . . . .      (Y, N, T, G or blank)
  Spanned / Nonspanned  . . . . .      (S=Spanned, N=Nonspanned or blank)
```

Figure 4-8 DATA CLASS DEFINE ISMF panel

After creating a data class with the attributes above, users can code the DATACLAS value on their DD statements or let the ACS routines assign the appropriate data class for their eligible data. Another method in which JCL can be used to create a KSDS with extended addressability is through the DD statement keyword LIKE.

Applications that access a VSAM extended format KSDS through a user-specified key can take advantage of having very large VSAM components without making JCL or code changes.

To support EA, many DFSMS macros and commands have changed. To take advantage of extended addressability, new macro parameters and sub-parameters related to RBA have been added.

- RPL macro, added the XRBA subparameter to the OPTCD parameter to indicate that extended addressability is be used. It must be used when processing a data set by its RBA. For example, OPTCD=(ADR,DIR,XRBA) instead of OPTCD=(ADR,DIR,RBA).
- TESTCB macro, added XADDR as a possible value to ATRB. It can be used to test if the data set is in extended addressability format.
- SHOWCB macro, added:
  - XAVSPAC parameter to obtain the amount of available space in the data component or index component, in bytes.
  - XENDRBA, to obtain the high-used RBA (HURBA)
  - XHALCRBA, to obtain the high-allocated RBA (HARBA)

  The fields above use 8 bytes to return the information, instead of 4 bytes used for non-EA data sets.

Applications that process an EF data set by RBA must provide an 8-byte RBA when the data set is defined for EA. Special provisions are allowed for certain types of requests (for example GET SEQ,ADR or GET ADR,DIR,LRD).

A major idea in the EA design is to make applications, such as backup, transparent to the function. That is, we do not require an extended field (8-bytes) XRBA, unless it is a positioning request. In the case of a backup when the data set is to be just read sequentially from the beginning, requiring no positioning, the extended field is not required. Also, RLS processing does not support any use of RBA or XRBA access to an EA data set.

With DB2 V1.6 and later, EA can be used for very large DB2 table spaces.

A VSAM EF KSDS defined for EA must not be shared with any system running a release prior to DFSMS/MVS 1.3. Toleration maintenance is available for DFSMS/MVS 1.2 systems which support VSAM extended format data sets without EA. A DFSMSdfp toleration PTF allows systems at this level to issue an error message if any attempt is made to open a VSAM KSDS with extended addressability. DFSMSdss provides a toleration PTF so that an error message is issued when a logical DUMP, RESTORE, or COPY operation is attempted on a KSDS with extended addressability.

DB2 data warehousing projects that require table spaces larger than 4 GB can use the EA support for linear data sets provided in DFSMS/MVS 1.5. To do this, assign a data class with the extended addressability attribute to the data set when it is defined. The data class should have the following attributes specified for it:

```
Recorg = LS
Data Set Name Type = Extended
IF Extended = Required
Extended Addressability = Yes
```

Then make sure your data class ACS routine for DB2 permits the use of the data class created.

The message `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS 116` is produced when you try to go beyond 4 GB in a VSAM data set without EA.

4.5 Catalog Search Interface

The Catalog Search Interface (CSI) is shipped as a component of base since DFSMS/MVS 1.5. As its use is not wide spread, we take this opportunity to remind you of its availability and to point out its advantages over other methods of obtaining catalog information. The following section describes:

- CSI setup
  - CSI programming considerations
  - IBM supplied sample program

4.5.1 CSI setup

The CSI is a general-use programming interface for obtaining information from ICF catalogs. It provides great flexibility in specifying the selection criteria for the data that is to be returned. The CSI may be invoked by assembler programs, high-level language programs and REXX execs. See the appendix in *Managing Catalogs*, SC26-4914, for a complete description of the interface. We present an sample REXX using CSI in “SMFLSR sample program” on page 380.

Much of the information you can obtain from the CSI you could also obtain using an IDCAMS LISTCAT command. However, there are some advantages using CSI that you may want to consider when accessing catalog information:

- Using a Generic Filter Key:

  When requesting information from the CSI for specific catalog entries, you may specify a generic filter key. This key can contain the following symbols used to filter the entry names:

*****	A single asterisk represents one or more characters within a qualifier
******	A double asterisk represents zero or more qualifiers
%	A percent sign represents one alphanumeric or national character

%%...	Up to eight characters can be specified in one qualifier

- Using selection criteria fields

  When requesting information from the CSI for specific catalog fields, you may specify a list of field names. For example, if you were only interested in the volume and the file sequence number for specific data sets, you could specify the catalog field names VOLSER and FILESEQ in the field name list when calling the CSI. Obtaining this information from IDCAMS would require you to use the IDCAMS LISTCAT ALL command and to scan the output to retrieve the desired information.

- Performance benefits

  Using the CSI generally results in significantly better performance compared to using IDCAMS LISTCAT, which does a catalog call for each entry processed. The flexibility in requesting only the information that you are interested in using the CSI results in additional performance improvements, since you eliminate the retrieval of unneeded information.

- CSI programming considerations

  The CSI is distributed as load module IGGCSI00 in SYS1.LINKLIB. It is reentrant and reusable, can be invoked in 24-bit or 31-bit addressing mode, in any PSW key, and in either problem or supervisor state.

  CSI requires three parameters to process your request:

  – A 4-byte *reason area* used to return error or status information
  – A variable length *selection criteria list (for input)*
  – *Work area* used to return the requested catalog data

- IBM supplied sample programs

  IBM provides three sample assembler programs and one REXX exec in SYS1.SAMPLIB. Here we provide a short summary of their functions:

- IGGCSILK produces output similar to that of an IDCAMS LISTCAT CAT(catname) command.
- IGGCSIVG identifies unused space at the end of VSAM data sets defined in a given catalog. This is calculated as the difference of the high-allocated and the high-used relative byte address (HARBA-HURBA)
- IGGCSIVS produces a list of data set names defined in a given catalog that reside on a specific volume. Such a list might be helpful in a volume recovery situation.
- IGGCSIRX is a REXX exec that produces a list of data set names matching a generic filter key. When you call it from a TSO/E session, it will prompt you for the filter key, and return matching data set names, their type, and volume definition.

4.6 Major sources of VSAM processing options

The VSAM cluster processing options and characteristics come from different sources, for example, JCL, SMS data classes, or macro parameters. The source of the processing options can be confusing since the same parameter can be specified in more than one source. In this case, there is an order of precedence that, when not understood, may give logically unexpected results. Table 4-3 lists the processing options and the characteristics and their sources, indicating the precedency by the highest number. The underlined value in MACRF indicates the default. The *Type* explains if the parameter is related to a data set characteristic (C) or processing option (P).

Table 4-3 VSAM Data Set Parameters and Processing Options

Parameter	**Type**	**ACB**	**DD**	**Defi ne**	**SMB DC**
ACCBIAS	P	-	2	-	1[a]
AVGREC (RECORDS)	C	-	2	2	1
BUFSP	P	2[b]	3b	1[c]	-
BUFND,BUFNI	P	2	3	1	-
BWO	P	-	-	2	1
CATALOG	C	-	-	1	-
COMPACTION	C	-	-	-	1
CISZ	C			2	1
DATACLAS (DATACLASS) [d]	C	-	1	1	-
DDNAME	P	1	-	-	-
DYNAMIC VOLUME COUNT	P	-	-	-	1
ERASE/NOERASE	P	-	-	1	-
EXCEPTIONEXIT	P	-	-	1	-
EXLST	P	1	2[e]	-	-
EXPDT	C	-	2	2[f]	1,3[g]
EXTENDED ADDRESSABILITY	C	-	-	-	1
EXTENDED FORMAT	C	-	-	-	1
FREESPACE	C	-	-	2	1

Parameter	Type	ACB	DD	Define	SMB DC
FRLOG	P	-	3	2	1
LIKE	C	-	1	1[h]	-
KEYLEN	C	-	2	2	1
KEYOFF	C	-	2	2	1
LOG	P	-	-	2	1
LOGSTREAMID	P	-	2	2[i]	1
LRECL (RECORDSIZE) [j]	C	-	2	2	1
MGMTCLAS (MANAGEMENTCLASS) d [k]	P	-	1	1	-
MACRF=[NSR I LSR I GSR I RLS]	P	1	[l]	-	-
MACRF=[ADR,CNV,KEY]	P	1	-	-	-
MACRF=[CFX I NFX]	P	1	-	-	-
MACRF=[DIR][,SEQ][,SKP]	P	1	-	-	-
MACRF=[DFR I NDF]	P	1	-	-	-
MACRF=[DDN I DSN]	P	1	-	-	-
MACRF=[IN,OUT]	P	1	-	-	-
MACRF=[ICI I NCI]	P	1	-	-	-
MACRF=[LEW I NLW]	P	1	-	-	-
MACRF=[NIS I SIS]	P	1	-	-	-
MACRF=[NRM I AIX]	P	1	-	-	-
MACRF=[NRS I RST]	P	1	-	-	-
MACRF=[NUB I UBF]	P	1	-	-	-
OWNER	C	-	-	1	-
RECATALOG/NORECATALOG	C	-	-	1	-
RECORG	C	-	1	1	-
RETPD	C	-	2	2[m]	1,3[n]
REUSE/NOREUSE	P	-	-	2	1

Parameter	Type	ACB	DD	Defi ne	SMB DC
RLS CF Cache Value	P	-	-	-	1
RLSREAD={NR ICR NORD}]	P	1	l	-	-
RMODE31	P	1	2	-	-
SHAREOPTIONS	C	-	-	2	1
SPACE (CYL, KB, MB, REC, TRK)	P	-	2	2	1
SPACE CONSTRAINT RELIEF	P	-	-	-	1
SPANNED/NONSPANNED	C	-	-	2	1
SPEED/RECOVERY	P	-	-	2	1
STORCLAS (STORAGECLASS) d	P	-	1	1	-
STRNO	P	1	2	-	-
VOLUME	P	-	2[o]	2	1
WRITECHECK	P	-	-	1	-

a. Correspond to Record Access Bias in Data Class
b. Specifies the *maximum* space for buffers
c. Specifies the *minimum* space for buffers
d. Can be overridden by the ACS routines
e. Through the AMP, SYNAD parameter
f. TO(*x*) FOR(*x*) of DEFINE command
g. When specified in MANAGEMENT CLASS, can not be overridden
h. Correspond to MODEL in DEFINE command
i. LGSTREAM JCL parameter
j. LRECL does not apply to LINEAR data sets
k. Overrides Data Class parameters EXPDT RETPD
l. RLS=(NRI or CR) when in ACB RLSREAD=NORD or does not specifies RLS.
m. TO(*x*) FOR(*x*) of DEFINE command
n. When specified in MANAGEMENT CLASS, can not be overriden
o. Only when Storage Class with Guaranteed Space= yes

4.6.1 ACB control block

The ACB is a control block within the application program. The ACB is to VSAM as the DCB is to other access methods. In its fields, the ACB contains logical information about the VSAM cluster. The ACB information is used by Open, Close and VSAM routines. The way that the ACB is created depends on the source language of the application program:

- In Assembler through the ACB macro at assemble time or GENCB at execution time.
- In Cobol through the File Description.
- In through DCL statement.

The source of ACB fields can be:

  - From keywords in macro ACB at compile or assembler time.
  - At allocation, with ACB parameters coming from AMP parameter, in the DD statement.
  - From keywords in macro GENCB at execution time by the application program.

Here is a list of the ACB fields. The ones starting with an asterisk indicate that they can also be entered in the JCL DD statement:

  - DDName, which is the link with the DD statement. The DD statement besides its optional complementary ACB information, describes the physical characteristics of the cluster, as VOLSER, device type.
  - The size (or the maximum size) of the I/O buffer virtual storage space and/or the number of I/O buffers to process data and index records.
  - The chosen buffering technique, such as: NSR, LSR, GSR, RLS.
  - The defer write option, that is to delay the buffer destage.
  - The processing options that you plan to use:
    - Keyed, RBA, RRN
    - Sequential, direct, or skip sequential access, or a combination. This field is just an intention
    - Retrieval, storage, or update (including deletion), or a combination
    - Shared or nonshared resources
  - The address of an exit list for your exit routines. Use the EXLST macro to construct the list.
  - What to do when lock contention arises.
  - Read integrity options for RLS GETs.
  - If you are processing concurrent requests, the number of requests (STRNO).
  - The address and length of an area for error messages from VSAM.
  - RMODE of the buffers and VSAM control blocks.

- Address of the VSAM GET/PUT routine. This field is filled by the OPEN macro function.

4.6.2 DD statement keywords

Please refer to the *MVS JCL Reference*, SA22-7597-04, to get information about all the generic DD Statement keywords which apply to VSAM. Here are VSAM specific parameters not present in the ACB:

- Options for controlling System Managed Buffering (SMB) named ACCBIAS in the AMP keyword
- Logstream option for RLS and Transactional VSAM

4.6.3 Catalog BCS and VVDS entries

The catalog entry for a VSAM cluster or a component has the attributes usually needed to create this VSAM entity. Those attributes are entered in the catalog through the IDCAMS DEFINE/ALLOCATE/ALTER commands:

- Primary and secondary space information
- CI size
- Free space percentages
- Logical record size
- Cluster organization data set (KSDS, RRDS,ESDS, LDS, VRRDS)
- Key length and key offset
- Name of SMS constructs, as DC, SC, M C
- Retention period
- Share options: cross region and cross system
- Spanned records
- Backup while open (BWO) options
- The minimum buffer space size
- Expiration date
- Initial load options (SPEED and RECOVERY)

4.6.4 SMS constructs

Here are the SMS constructs that effect VSAM attributes.

Data Class construct

The data class construct has almost the same options of the IDCAMS define command. The practical difference is that, with a data class, you do not need to define all the IDCAMS keywords. You just point to the data class name. Also, these options are centralized and controlled through the SMS storage administrator. The Following options in data class are not present in the IDCAMS DEFINE, and are not stored in the catalog:

   - To use or not the primary size in a secondary volume
   - Maximum number of volumes
   - Extended format
   - Extended addressability
   - Options for controlling System Managed Buffering (SMB) named RECORD_ACCESS_BIAS
   - Space constraint relief
   - Data Compression
   - RLS use of the CF cache structure

Storage Class

The storage class SMS construct has parameters to control:

- Performance as:
   - How the I/O operation uses the controller cache
   - Are the CIs stripped?
   - CF structure name to be used as a cache by VSAM RLS
- Availability as:
   - Guarantee to the VSAM cluster a controller with RAID
   - Guarantee to the VSAM cluster a controller with T0 copy and remote copy

Management Class

The management class SMS construct has parameters to control VSAM clusters expiration, partial release, GDG, backups and migration.

4.7 Media Manager, Open, Close, EOV in VSAM

Media Manager is the I/O driver code. It stands between the access method and the Input Output Supervisor. VSAM has two I/O drivers depending on the required function:

- Block Processor (SVC 121), that is old and there is a SOD for being inactivated. Since z/OS V1R4 DFSMS Block Processor is being used only for Improved Control Interval processing (ICI).
- Media Manager, which is modern. There is a trend to all access methods to use Media Manager. Media Manager has the I/O channel program support for implementing Extended Format.

Media Manager executes all these functions:

- Creates Channel Programs with virtual addresses (this was done before by VSAM)
- Page-Fix (or Page-Free) buffers
- Verifies that buffers are accessible in user key
- Translates virtual addresses to/from real addresses in the channel program
- Validates that RBA is within data set
- Re-drives Channel Programs to IOS (through STARTIO macro)
- Provides for DCME statistics
- Invokes SMFIOCNT

4.7.1 OPEN macro

Before an application program can access a data set, it must first issue the OPEN macro to open the data set for processing. The OPEN is issued against a user Access method Control Block (ACB). Opening a data set causes VSAM to:

- Verify that the data set matches the description specified in the ACB or GENCB macro. For example, MACRF=KEY implies that the data set is KSDS.
- Construct the internal control blocks and buffer pools that VSAM needs to process your requests for access to the data.
- Load the access method (based in the ACB information) placing the address in the ACB for the next GET or PUT.
- Check your program security authorization (RACF).

4.7.2 CLOSE macro

The CLOSE macro disconnects your program from the data set. VSAM does the following during CLOSE:

- Writes any unwritten data or index records whose contents have changed.
- Writes SMF records if using SMF.
- Updates the catalog entry for the data set. Updates the data set's high-used RBA (HURBA).
- Restores control blocks to the status they had before the data set was opened.
- Releases virtual storage obtained during OPEN processing for additional VSAM control blocks and VSAM routines.

- For extended format KSDS, releases all space between HURBA and HARBA if partial release was specified at OPEN.

If an abend happens during CLOSE, the HURBA which is updated during CLOSE processing, may be incorrect because the catalog was not updated.

The next VSAM OPEN performs an implicit VERIFY. If the VERIFY is not successful, VSAM OPEN passes a return code and reason code.

You can issue a CLOSE TYPE=T or a temporary CLOSE. This causes VSAM to complete any outstanding I/O operation, update the catalog, and write any required SMF records. Then processing can continue without issuing an OPEN macro.

4.7.3 End-of-Volume (EOV)

EOV function is invoked by VSAM Record Management, when a VSAM data set requires additional space. The lack of this additional space is perceived by:

- High-used RBA (HURBA) = High-allocated RBA (HARBA)
- RBA of next CI/CA greater than HARBA during create
- No extent in the current volume contains a specific searched RBA

Then, EOV acquires new extents interfacing with DADSM, updates the VSAM control block structure for the data set with the new extent information, and updates the critical control block data in common storage and in the catalog, so that this new space is accessible by all regions using this VSAM data set.

4.8 VSAM and 64 bits

Since z/OS V1R4 DFSMS, VSAM supports 64 bits of *real* storage. Some VSAM functions was modified to can use real storage address:

- Media Manager:
  - Now performs I/O to all VSAM data sets, except when using ICI processing.
  - Full support for ESS-2105 (SHARK), allowing Read Track Data, Write Track Data and Write Full Tack.
  - Support for 64-bit real addresses used to back 31 bit virtual addresses.
- OPEN routine:
  - Obtains buffers in 64-bit *real* storage in all cases.

  - Eliminates all control blocks and data from CSA and SQA, except when GSR and ICI is being used. The data and control blocks now use private storage area.
  - Initializes a protected field for the high allocated CI, for being later updated by the EOV routine when extending the data set.
- Record Management:
  - Uses the information of the high allocated CI, stored in a protected page, to search the index. It avoids loops when a pointer is corrupted.
- VSAM Hiperbatch:
  - Now supports VSAM extended format data set
- Checkpoint/Restart:
  - Now supports VSAM extended format data set
- EOV routine:
  - EOV does not support ORDERED parameter. If space is not available on the next candidate volume, a different candidate volume is used.

4.9 Special considerations for COBOL users and SMB

This was added to provide some insight for COBOL users.

4.9.1 COBOL users take note

Note: This information was provided by Helen Witter.

Try to keep in mind is that COBOL is written for NSR processing. If anything is done to switch the processing to LSR (whether OEM, SMB or BLSR), the program or COBOL may not be able to handle it. The good news is that SMB can be disabled for a particular program by coding an override on the DD card and letting other files that access the dataset continue to use SMB. The most common problem is a positioning error. With NSR processing, implicit positioning by VSAM to the first record in the file is done on the first sequential GET. Many programs (especially older ones) take advantage of this and do not explicitly position to the first record of the file. With LSR, there is no implicit positioning done so if the program does not position, the same GET that worked with NSR will be returned with a rc58. This translates into a COBOL SK30. This should be minimal since SMB won't decide to do DO processing on a file that is opened with sequential. The other very common problem is waits/hangs. This is also due to the inherent differences between NSR and LSR. With NSR you may have multiple requests in the pool accessing the same data CI, because a copy of the CI will be given to each request. In LSR, you can have multiple requests reading the same CI (they will all share the same buffer rather than get their own copy), but a request for update of that CI must wait until all the readers have finished with it. This can cause hangs. There are a number of possible circumvention for this. It depends on what the program is doing and in what order. If, for example, if the program has two ACB's, it could open the sequential ACB first then DO would not be selected. Another possible problem could be very old programs that were written for ISAM datasets. If the customer has been using the VSAM/ISAM interface to continue to use these programs, they must continue to use NSR. Again, the impact should be minimal as the ACB probably has sequential specified. Some SMB users, not just COBOL, are finding that they need to increase their region size when running SMB because of the extra storage required to optimize the buffering. This is especially true on large files or programs that open many SMB VSAM files. In general, SMB is pretty "smart" and the user should see a good deal of performance benefits from using SMB, however, it still comes down to some applications might need minor JCL or coding changes to use SMB.

5

VSAM Record Level Sharing

This chapter introduces VSAM Record Level Sharing (RLS) concepts and describes how to implement RLS.

The topics cover:

- Introducing RLS
- RLS terminology
- Planning for RLS
- Implementing RLS
- MVS commands for RLS
- RLS problem determination and recovery
- RLS enhancements
- RLS performance considerations

5.1 Introducing VSAM RLS

In this section we introduce the basic concepts of VSAM RLS mode.

5.1.1 What is VSAM RLS?

VSAM RLS is another mode of managing buffer pools, which allows any number of users within your Parallel Sysplex to share your existing VSAM spheres. It provides full data integrity (read and write). The serialization is at record level. However to implement recoverable spheres the user must have its own backout log, as CICS has.

VSAM RLS does not introduce new types of VSAM clusters; rather, it introduces a new way of accessing existing data sets. Apart from the need to open data sets in RLS mode, the same VSAM record management interfaces (get, put, point, erase) are used.

You can specify the RLS mode in the MACRF parameter of the ACB macro that you use to open the data set in your program. You can also specify the RLS mode, in the new keyword ‘RLS’ in your DD card that points to your VSAM data set in your JCL.

RLS mode maybe used with KSDS, RRDS, VRRDS, and ESDS VSAM spheres. PATH access for KSDS and ESDS's is allowed with RLS. Extended format, extended addressibility, spanned, and compression are also supported with RLS.

RLS and non-RLS VSAM data sets can co-exist.

5.1.2 Why RLS?

RLS allows concurrent access, in a sysplex, to your VSAM data sets at record level, while maintaining data integrity.

VSAM RLS also addresses other VSAM issues, mainly in the CICS environment. Refer to “CICS and VSAM RLS” on page 246.

5.1.3 How does RLS work?

RLS logic is implemented in programs running in a VSAM address space named SMSVSAM, these programs gain the control from the RLS requesters through cross memory (PC instruction). The major difference between RLS and non-RLS modes is that the VSAM sphere control block structure is located in a CF cache structure and it is shared by all string task programs (also called users) in RLS mode. Refer to “VSAM sharing mechanisms” on page 216 for details of control block structures.

Figure 5-1 shows in a very simplified form how a VSAM spheres is shared between various address spaces (CICS and Batch jobs) across a Parallel Sysplex. Each z/OS system in the sysplex has an SMSVSAM address space to co-ordinate the sharing. Shared Control Data Set (SHCDS) contains critical information that is used by various SMSVSAM address spaces for RLS. The coupling facility contains:

- Lock structure IGWLOCK00 which contains global locks to serialize at record level and to serialize functions as splits.
- Cache structures used to hold shared data and control block structures.

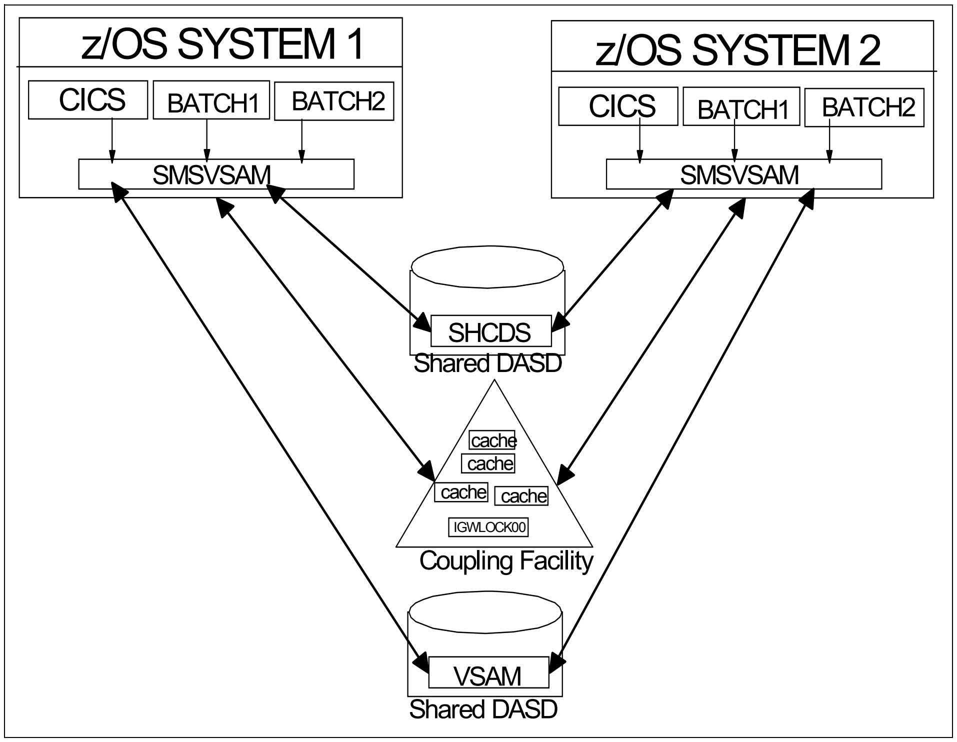


Figure 5-1 RLS in Sysplex

5.1.4 RLS in a single system (monoplex)

You may access VSAM sphere in RLS mode by users running in just one z/OS image. However, even in a monoplex environment a Coupling facility is required. Figure 5-2 shows an example of such a setup.

The reason for doing that is to exploit the better serialization provided by RLS. For example, in a non-RLS mode two jobs trying to access for update a cluster with total read and write integrity, that is, SHAREOPTIONS (1,3), just one can open the cluster. The other needs to wait for the first to close the cluster. The granularity of the serialization is at cluster level.

If the two jobs access the cluster in RLS mode, then both the OPENs will succeed, and both in parallel can update the cluster. The serialization is at logical record level.

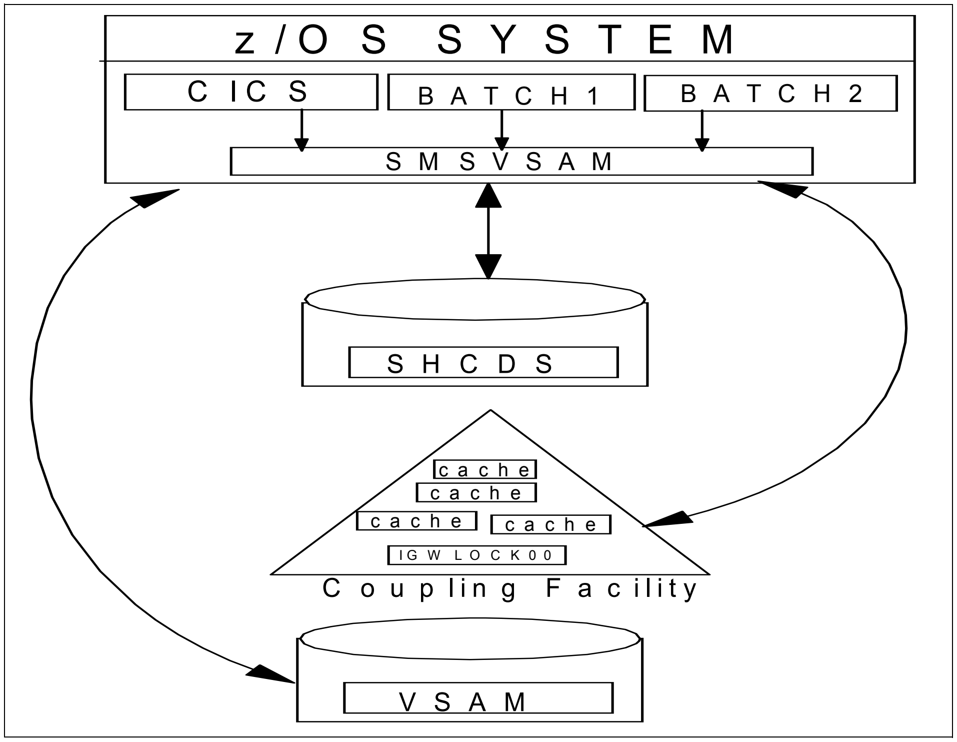


Figure 5-2 RLS in Monoplex

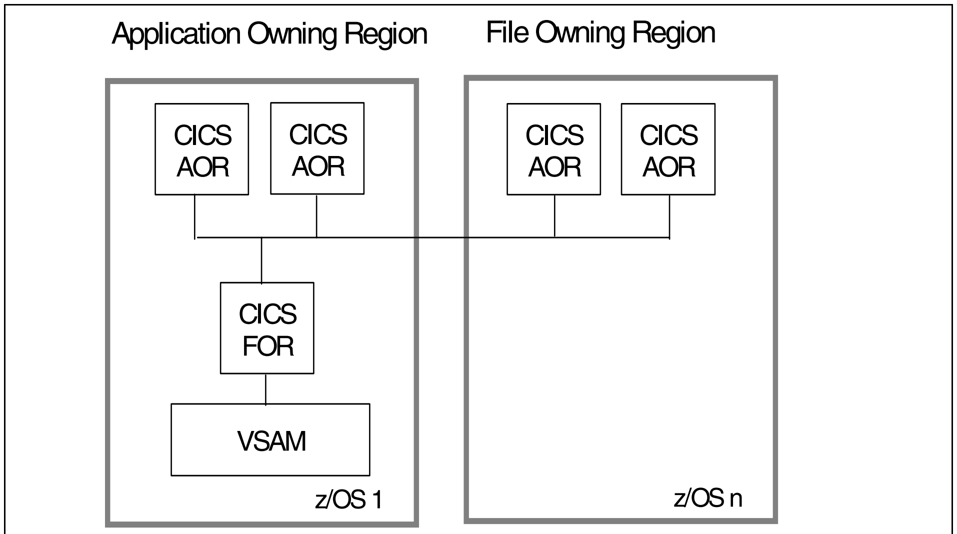
5.1.5 CICS and VSAM RLS

Without VSAM RLS, CICS uses a process known as *function shipping* to share spheres between CICS systems. Function shipping is implemented by using a single CICS region known as a File Owning Region (FOR) to own a data set. All access to the data set was through the corresponding FOR, a focal point for VSAM access. The role of the FOR is simply to provide file access; the CICS transactions run elsewhere in Application Owning Regions (AORs). These regions run programs which, when they want to access a shared VSAM sphere, ship the access request off to the FOR for that sphere and await the reply from the FOR. This is illustrated in Figure 5-3.

This implementation has several problems as:

- Function shipping between AORs and FORs in distinct systems, consuming resources.
- The FOR is a single point of failure and can also be a CPU bottleneck
- Locks are held by CICS/FOR. An in-doubt lock held by a CICS task can cause an integrity problem if FOR fails. In this case, there is no way to prevent other applications from accessing uncommitted data. With RLS, locks are held by SMSVSAM (not by CICS/FOR). Then the locks held by an in-doubt CICS task can be held if a CICS fails, preventing other applications from accessing uncommitted data until CICS is restarted or the locks are released by an operator command.
- Need of committing remotely.
- HURBA/HARBA are updated only when CICS closes the data set, causing conflicts along shared updates.

These problems are fully addressed by VSAM RLS.



Problems:

- CICS FOR is a single point of failure
- Multiple system performance is no acceptable (uses VTAM or XCF cross system)
- No exploitation of System/390 Parallel Sysplex

Figure 5-3 CICS before VSAM RLS

CICS with VSAM RLS is shown in Figure 5-4. The RLS data sharing environment is designed to support sharing of VSAM spheres among multiple cloned AORs. With VSAM RLS, we no longer need file-owning regions (FOR). Each copy of z/OS contains an SMSVSAM address space which manages locks within a coupling facility. Coupling facility configurations can be designed to be highly available. As we no longer have file owning regions, we have removed the single point of failure. We can also scale by adding z/OS systems to the parallel sysplex, with each new system running a SMSVSAM address space.

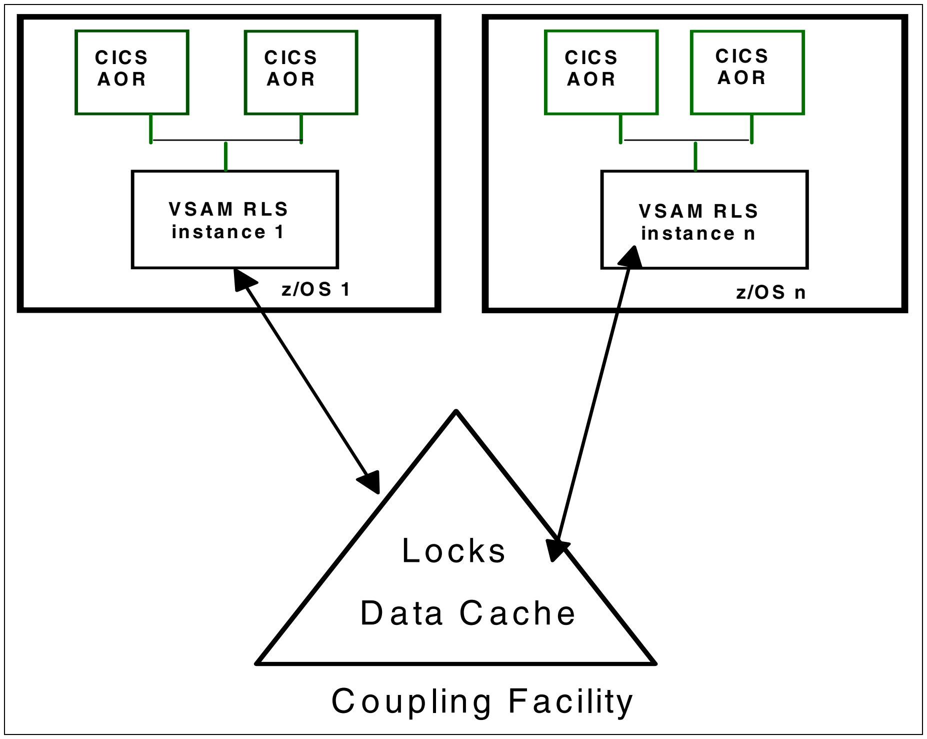


Figure 5-4 CICS after VSAM RLS

Attention: The version of CICS supporting RLS is CICS/TS.

For more details about VSAM RLS and CICS refer to the following publications:

- *CICS and VSAM Record Lavel Sharing: Implementation Guide*, SG24-4766
- *CICS and VSAM Record Lavel Sharing: Planning Guide*, SG24-4765
- *CICS VSAM Recovery User's Guide and Reference*, SH26-4127

5.1.6 RLS restrictions

There are restrictions that need to be considered before implementing RLS.

- To share a VSAM sphere under RLS, the sphere must be SMS managed
- IDCAMS currently does not have access to the control blocks in SYSVSAM address space. You will not be able to use IDCAMS to obtain information pertaining to RLS.
- Linear, keyrange, and temporary data sets are not supported under RLS.

- Because the control block structure is shared, RLS does not use share options.
- RLS does not support data striping.

5.2 RLS terminology

The terminology used in RLS books is sometimes different from the terms used more widely in z/OS and SMS. Therefore, it is worthwhile to define the RLS terms to avoid confusion.

SMSVSAM

This is the z/OS jobname of the RLS address space. This is created on each z/OS image according to PARMLIB specifications. It is responsible for centralizing in one z/OS all processing necessary for cross system sharing, that is, the access to the CF (with data, control block structures and locks) and connect among its peers through XCF.

VSAM sphere

A sphere is the name used to represent the components (data/index) and all the AIX of several related VSAM clusters.

RLS client

This is any address space which invokes a RLS function which results in a PC to the SMSVSAM address space. Examples of RLS functions are: OPENs, CLOSEs, GETs, PUTs, DELETEs, and so on. Examples of RLS client spaces are CICS, "batch jobs", CATALOG, DSS, HSM, and so on.

Subsystem

This is an RLS client space that "registers" with the SMSVSAM address space as an address space that provides recovery (that is, forward/backward logging). CICS is an example of recoverable subsystem.

Batch

This is an RLS client address space that does not first register with SMSVSAM as a recoverable subsystem (that is, does not provide logging). Examples of batch address spaces are: non-CICS VSAM applications, Batch, CATALOG, DSS, HSM, and so on).

Record lock

This is an XES lock resource obtained by SMSVSAM on behalf of a user and associated with a logical record. The lock resource name is based on a 16 byte hashed version of the record's key (or RBA, or RRN), as well as the sphere name and component name. There are also other locks to serialize splits, for example.

True contention

This is when two different "users" attempt to access the same record lock at the same time. Reported as true contention on the D SMS,CFLS command and in RMF reports. You need to tune your application if you have high true contention rates, or use NRI for reads.

False contention

This is when VSAM RLS assigns locked resources to an entry value in the lock table, and uses this entry value to quickly check whether a resource is already locked. If the lock structure (and thus the lock table) is too small, many locks can be represented by a single value, making "false" lock contention possible. False lock contention occurs when two different locks on different resources attempt to use the same lock entry. The second lock requester is suspended until VSAM RLS determines that there is no real lock contention on the resource.

This is reported as false contention on the D SMS,CFLS command. For high false contention rates need to create a larger lock structure.

True/False or False/False contention

This occurs when either true contention or false contention occurs, and the holder releases the record before the RLS contention exit runs. At this point RLS cannot tell between true or false contention. This type of contention is reported as false contention on the D SMS,CFLS. See Figure 5-13.

Deadlocks

This occurs when two different transactions, each holding a record lock, attempt to obtain the other transactions record lock. SMSVSAM abnormally terminates one of the waiting lock requests (RPL RC=8 RSN=21), based on the Deadlock_detection value in your IGDSMSxx. Refer to 3.2.16, “Deadlocks” on page 160.

Read integrity

SMSVSAM allows the user to decide about read integrity through parameters in DD card or ACB macro. If the option is consistent read (CR), logical records locks are required in shared mode for reads. If the option is no read integrity (NRI), no locks are required along reads.

Retained lock

This is a record lock obtained by a recoverable subsystem such as CICS, which was still held at the time of a failure in: CICS, IGWLOCK00, CF, z/OS, other SMSVSAM, or Indoubt transactions. A job can be cancelled without retained locks because this job dose not register, as subsystem with SMSVSAM.

Lost Locks

A sphere is said to be in "Lost Locks" if the sphere was being accessed by a recoverable subsystem when a failure to the lock structure occurs at the same time as a failure in at least one of the SMSVSAM address spaces (double failure scenario). Use the IDCAMS SHCDS LISTSUBSYS(ALL) command to list CICS subsystems holding retained or lost locks. This situation can be avoided by duplexing or failure isolation.

Recoverable sphere

This is a VSAM sphere which is accessed by a recoverable subsystem (that is, CICS) which provides logging for the sphere. A recoverable sphere is defined by the LOG(UNDO/ALL) attribute in the catalog.

Non-recoverable sphere

This is a VSAM sphere for which no logging is required. A non-recoverable sphere is defined with the LOG(NONE) attribute in the catalog.

Quiesce a sphere

There is a special interface provided by RLS for CICS to close a sphere across the sysplex. QUIESCE = YES is set in the catalog following the quiesce request. There is a command “D SMS,SMSVSAM,QUIESCE” on page 275, that shows the quiesce state of a sphere.

Quiesce transactions

There is a special interface provided to CICS and DSS, to temporarily halt transactions for a given sphere in order for DSS to take a sharp copy (also called a T0 copy).

Quiesce a volume or a cache

There is a command to SMSVSAM to stop new opens for RLS to a particular volume or cache. The formats are as follows:

```
V SMS,CFVOL(volid),QUIESCE

V SMS,CFCACHE(cachename),QUIESCE
```

Sharing control

This is a set of routines executing in the SMSVSAM address space which open, format, read and write to the Sharing Control data sets (SHCDS). SHCDS - Linear data sets, accessed by SMSVSAM in non-RLS mode which contain "state" type information about RLS client spaces and VSAM spheres accessed by SMSVSAM.

RLS mode

A sphere is said to be in RLS mode when it was last accessed by RLS. RLS mode is indicated by the RLS-IN-USE indicator in the catalog.

5.3 Planning for RLS

This topic covers planning activities needed to implement RLS.

5.3.1 Hardware requirements

VSAM RLS uses the coupling facility (CF) to perform sphere level locking, record locking, and data caching. If your system is running in a sysplex you already have the necessary hardware to implement VSAM RLS.

If you are going to implement system-managed duplexing for RLS lock structure for improved availability, then you would require additional CF storage, CF processor capacity, and z/OS-to-CF link capacity in order to accommodate duplexed VSAM RLS lock structure instances, and the duplexed VSAM RLS lock structure operations to them from each participating z/OS image.

Note: If you want to implement VSAM RLS on a standalone z/OS system, to share VSAM across multiple address spaces, then you must not run in XCFLOCAL mode. A coupling facility is still needed for the lock structures even though there is only a single z/OS image.

5.4 Implementing VSAM RLS

We describe the step-by-step procedure to implement VSAM RLS.

5.4.1 Define Sharing Control Data Set (SHCDS)

Here we give some considerations and examples to define SHCDS.

What is its function?

The SHCDS is critical to maintaining data integrity in the event of the failures of the Parallel Sysplex, SMSVSAM address space or the CF lock structure.

The SHCDS contains the name of the CF lock structure in use, a list of subsystems, the status of the subsystems, a list of open spheres using the CF and other information.

How do I define it?

You must use the following naming convention when defining your SHCDSs:

`SYS1.DFPSHCDS.`*qualifier*`.`*Vvolser*

Where:

qualifier is a 1 to 8 character qualifier.

volser is the volume serial number. The V prefix allows you to specify numeric volume serial numbers.

Important: The SHCDS naming convention depends on the volume to match the SHCDS name. So do not move SHCDS across volumes.

How much space to allocate for SHCDS?

Use the following formula to calculate the size of your SHCDS:

```
S = (16 + (N * (16 + C/10))) kilobytes
```

Where:

S is the space required for the SHCDS.

N is the number of systems.

C is the number of concurrent OPEN requests that you expect.

Sample JCL to allocate SHCDS

The SHCDS may be defined using IDCAMS. See Example 5-1.

Example 5-1 Allocate SHCDS by IDCAMS

```
//ALLOCLD1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.PRIMARY.VSMS001) LINEAR -
          STORCLAS(GSPACE)                             -
          SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS001) )
    DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.SECONDARY.VSMS002) LINEAR -
          STORCLAS(GSPACE)                             -
          SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS002) )
```

```
   DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.SPARE.VSMS003) LINEAR -
         STORCLAS(GSPACE)                              -
         SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS003) )
 /*
```

Example 5-2 shows how you can also use DD statements in JCL to allocate these spheres. Use a data class defined with a cross-region share options value of 3 and a cross-system share options value of 3.

Example 5-2 Allocating SHCDS using JCL DD statements

```
//PRISHCDS DD DSN=SYS1.DFPSHCDS.PRIMARY.VSMS001,SPACE=(1,(10,10)),
//             RECORG=LS,STORCLAS=GSPACE,VOL=SER=SMS001,AVGREC=M,
//             DISP=(NEW,CATLG)
//SECSHCDS DD DSN=SYS1.DFPSHCDS.SECONDRY.VSYS002,SPACE=(1,(10,10)),
//             RECORG=LS,VOL=SER=SYS002,AVGREC=M,
//             DISP=(NEW,CATLG)
//SPRSHCDS DD DSN=SYS1.DFPSHCDS.SPARE.VSMS003,SPACE=(1,(10,10)),
//             RECORG=LS,STORCLAS=GSPACE,VOL=SER=SMS003,AVGREC=M,
//             DISP=(NEW,CATLG)
/*
```

SHCDS considerations

- At a minimum, define and activate two SHCDSs and at least one spare SHCDS for recovery purposes.
- Because the contents of these data sets are highly dynamic, we do not recommend backup and restore functions for these data sets.
- The SHCDS is a VSAM linear data set.
- The CISIZE for SHCDS must be 4096.
- When defined, the SHCDS does not need to be catalogued on all systems in the sysplex. If it is cataloged, it must be in a catalog available when SMSVSAM initializes.
- If the SYS1.DFPSHCDS.qualifier.Vvolser cannot be catalogued into the MCAT, use the multi level alias (MLA) facility (2 levels at least) and define a SYS1.DFPSHCDS as alias in the MCAT.
- Place the SHCDSs on volumes with global connectivity.
- The share options for SHCDSs must be set to (3,3) so that each system in the Parallel Sysplex can properly share the data sets.
- Use storage classes defined with the guaranteed space attribute.
- Avoid placing SHCDSs on volumes for which there might be extensive volume reserve activity.

- Secondary space definition is strongly recommended. All extents for each data set *must* be on the same volume.
- SMSVSAM remembers the SHCDSs from one SMSVSAM recycle to the next. Do not delete or redefine an active or spare SHCDS without first telling SMSVSAM. Use the command **V SMS,SHCDS(shcdsname),DELETE.**
- SMSVSAM must be RACF authorized to update SYS1.DFPSHCDS.* data sets. Refer to “Security definitions” on page 263.
- For some SHCDS errors, an IPL would NOT help because SHCDS is remembered from one IPL to another. FALLBACK procedure documented in “SHCDS FALLBACK procedure” on page 267 is the only way to correct the problem. If you get repeated abends with the prefix '67' in the RSN code, then you know it's time to do a FALLBACK.

Important: Use the FALLBACK procedure only as the last resort when everything else fails.

5.4.2 Define CF cache structures

VSAM RLS uses the Coupling Facility (CF) which is accessible from all the z/OS systems in the sysplex to maintain cache and lock structures to administer VSAM RLS. You can define several cache structures but just one lock structure.

What is its function?

The CF cache structure maintains RLS local buffer pool consistency at the control interval (CI) level of all SMSVSAM instances in a Parallel Sysplex. CF cache structures contains the control block structure, being used as a system buffer pool for VSAM RLS, providing a level of storage hierarchy between local storage buffers and DASD cache. That is, RLS uses the cache structure for store-through caching; along writes data is written from local buffer pool to DASD as well as to the cache structure.

How to define it?

You use the IXCM2APU utility program to define the cache structures in the CFRM couple data set. You should coordinate this activity with your z/OS systems programmer. See the sample JCL in Example 5-3 to define the cache structures.

Example 5-3 JCL to define RLS cache structures

```
//STEP01   EXEC PGM=IXCM2APU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=H
//SYSIN    DD   *
```

```
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(yourpolicy) REPLACE(YES)
       STRUCTURE NAME(CACHE01)
                 SIZE(4000)
                 INITSIZE(3000)
                 PREFLIST(yourCF1)
       STRUCTURE NAME(CACHE02)
                 SIZE(4000)
                 INITSIZE(3000)
                 PREFLIST(yourCF2)
/*
```

How to determine the cache size?

The total size of the cache structures should ideally be equal to the sum of the local VSAM LSR buffer pool sizes. After you get that figure, you should divide this size among the structures that will be defined. You can use the sizing tool available at the following IBM site.

http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html

Multiple cache structures

Once you have determined the total CF cache required, you may need to break this down into individual cache structures. You can define multiple SMSVSAM cache structures on different CFs and assign one or more CF cache structures to each *cache set* associated with a storage class.

To make it simple, (1) the sphere points to a storage class; (2) the storage class has the cache set name; (3) the cache set (in base SMS ACDS) has the name(s) of the structure(s).

Having multiple cache sets allows you to provide different performance attributes for spheres with distinct performance requirements. When more than one CF cache structure is assigned to a cache set, spheres within that storage class are cached in each CF structure in turn, in an effort of SMSVSAM to balance the load.

Considerations

Cache structures are built at open time and they remain even after the close of a sphere due to performance reasons (kept attribute). RLS uses LRU algorithms to release the structures.

Cache must be large enough for the CF cache directories to contain an entry for each of the VSAM RLS local buffers across all instances of the RLS server.

Enhancements allow VSAM spheres with a CI size greater than 4 KB records to be cached.

5.4.3 Define CF lock structures

Here we explain how to define the lock structures.

What is its function?

The CF lock structure maintains the record-level locks and sphere-level locks to enforce global serialization. The CF lock structure includes two parts:

- The lock table, which is used to determine whether there is R/W interest among systems on a particular resource
- Record table space to keep track of information for retained locks and spheres which have been processed by VSAM RLS

How do I define it?

The CF lock structure is named IGWLOCK00. You can use the same IXCMIAPU utility that you used for CF cache structure to define the lock structure in the CFRM couple data set, as well. See Figure 5-5 for sample jcl.

```
//STEP01   EXEC PGM=IXCMIAPU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=H
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(CONFIG1) REPLACE(YES)
              STRUCTURE NAME(IGWLOCK00)
                 SIZE(4000)
                 PREFLIST(yourCF01)
```

Figure 5-5 An example of defining a CF lock structure

How to determine the lock structure size?

You can use the following formula to estimate an initial size for the lock structure.

```
Size in Mega Bytes=10 * number_of_systems * lock_entry_size
```

The lock_entry_size depends on the MAXSYSTEM value defined for the sysplex couple data set when it was initially defined by the IXCL1DSU format utility. The MAXSYSTEM value represents the maximum number of systems that can be exist in the Parallel Sysplex.

The lock_entry_size is 2 if the MAXSYSTEM is 7 or less. It is 4 if MAXSYSTEM >= 8 and < 24 and will be 8 if MAXSYSTEM >= 24 and <= 32.

You can issue the following command to determine the MAXSYSTEM value.

DISPLAY XCF,COUPLE,TYPE=CFRM. See Figure 5-6.

```
-D XCF,COUPLE,TYPE=CFRM
 IXC358I  15.22.48  DISPLAY XCF 780
 CFRM COUPLE DATA SETS
 PRIMARY    DSN: SYS1.XCF.CFRM04
            VOLSER: SBOX67     DEVN: 3F39
            FORMAT TOD         MAXSYSTEM
            11/29/2001 13:36:18        4
            ADDITIONAL INFORMATION:
             FORMAT DATA
             POLICY(5) CF(4) STR(100) CONNECT(32)
             SMREBLD(1) SMDUPLEX(1)
 ALTERNATE  DSN: SYS1.XCF.CFRM05
            VOLSER: SBOX68     DEVN: 3B39
            FORMAT TOD         MAXSYSTEM
            11/29/2001 13:36:19        4
            ADDITIONAL INFORMATION:
             FORMAT DATA
             POLICY(5) CF(4) STR(100) CONNECT(32)
             SMREBLD(1) SMDUPLEX(1)
             CFRM IN USE BY ALL SYSTEMS
```

Figure 5-6 Displaying MAXSYSTEM

You can use the sizing tool available at the following IBM site.

http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html

Considerations on lock contentions

- You must monitor false contention to adjust the lock structure size. False contention can be monitored via RMF or via the DISPLAY SMS,CFLS command.
- The amount of real lock contention is application-dependent; it depends on record access patterns. False lock contention is almost entirely determined by the size of the lock table. That is, with a larger lock table having less false lock contention than a smaller one.

5.4.4 SMS definitions

This section defines which CF cache structures can be used to assign a VSAM sphere.

Define the CF cache structures names to SMS

SMSVSAM uses the SMS storage class construct, that you define, to determine which CF cache structures (varying sizes and performance levels) can be used to assign a VSAM sphere to a CF cache structure. Refer to Figure 5-7 and Figure 5-8.

Based on this you would then group similar (1 to 255) cache structures into cache sets. The storage classes SC54GRT and SCRLS have a pointer to a “cache set” named CSERLS.

These cache sets are defined in the BASE SMS construct and contains a list of CF cache structures, in the example: RLS_CACHE1 and RLS_CACHE2.

A VSAM sphere opened in RLS mode must have a storage class assigned to it.

The first time an RLS VSAM sphere is opened in the Sysplex, SMVSAM picks the *best* cache structure to use for the requested VSAM sphere and attempts to connect with it.

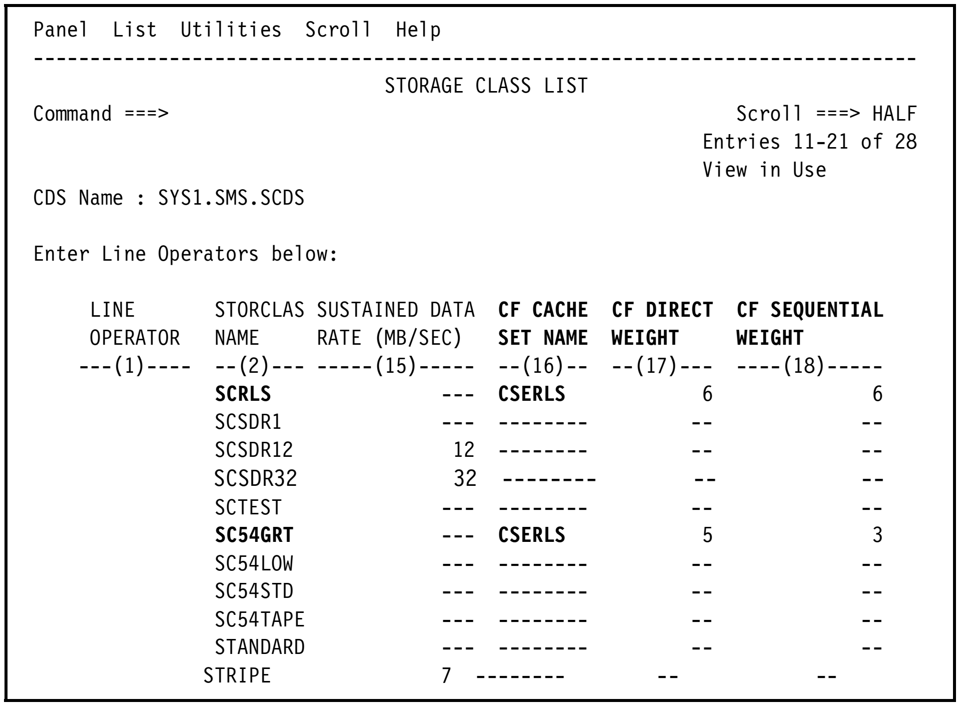

```
Panel  List  Utilities  Scroll  Help
------------------------------------------------------------------------------
                              STORAGE CLASS LIST
Command ===>                                                   Scroll ===> HALF
                                                            Entries 11-21 of 28
                                                            View in Use
CDS Name : SYS1.SMS.SCDS

Enter Line Operators below:

     LINE       STORCLAS SUSTAINED DATA  CF CACHE  CF DIRECT  CF SEQUENTIAL
     OPERATOR   NAME     RATE (MB/SEC)   SET NAME  WEIGHT     WEIGHT
    ---(1)----  --(2)--- -----(15)-----  --(16)--  --(17)---  ----(18)-----
                SCRLS               ---  CSERLS           6              6
                SCSDR1              ---  --------        --             --
                SCSDR12              12  --------        --             --
                SCSDR32              32  --------        --             --
                SCTEST              ---  --------        --             --
                SC54GRT             ---  CSERLS           5              3
                SC54LOW             ---  --------        --             --
                SC54STD             ---  --------        --             --
                SC54TAPE            ---  --------        --             --
                STANDARD            ---  --------        --             --
               STRIPE              7   --------       --            --
```

Figure 5-7   CF cache set in Storage Class

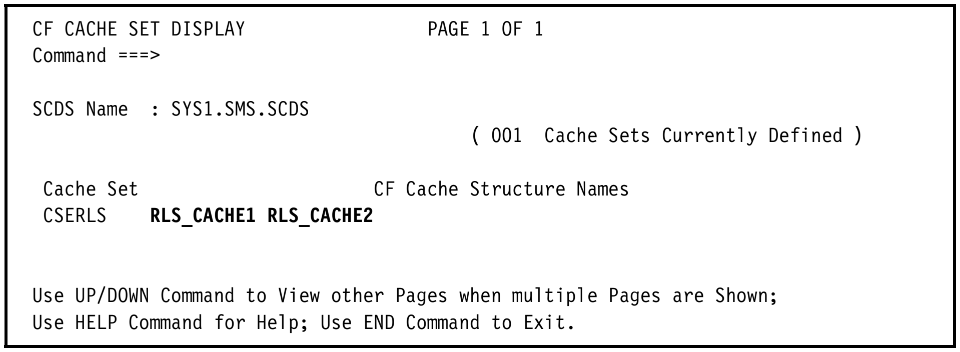

```
CF CACHE SET DISPLAY                PAGE 1 OF 1
Command ===>

SCDS Name  : SYS1.SMS.SCDS
                                        ( 001  Cache Sets Currently Defined )

 Cache Set                     CF Cache Structure Names
 CSERLS    RLS_CACHE1 RLS_CACHE2


Use UP/DOWN Command to View other Pages when multiple Pages are Shown;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-8   Cache Set assignment

There is also a weight storage class attribute indicating the data's relative importance in the CF DIRECT WEIGHT or the CF SEQUENTIAL WEIGHT fields. Use the CF DIRECT WEIGHT field for direct data; use the CF SEQUENTIAL WEIGHT field for sequential data. The default is a weight value of 6. You can specify a value from 1 to 11, with 11 indicating the highest relative importance.

The greater the weight value, the more important it is that the data be assigned more cache resources less stealing), thereby improving cache hit rates for the data.

Attention: SMSVSAM at z/OS 1.3 DFSMS only supports the default value. All data is assigned a weight value of 6 regardless of the value you specify.

5.4.5 Modifying the PARMLIB IGDSMSxx Member

In IGDSMSxx member you specify options for SYSVSAM.

Note: There is one SMSVSAM address space in each image of z/OS.

Example 5-4 shows the RLS parameters in bold.

Example 5-4 IGDSMSxx parmlib member

```
SMS ACDS(SYS1.SMS.ACDS)
    COMMDS(SYS1.SMS.COMMDS)
    INTERVAL(15)
    DINTERVAL(150)
    DEADLOCK_DETECTION(15,4)
    SMF_TIME(YES)
    CF_TIME(1800)
    RLSINIT(YES)
    RLS_MAX_POOL_SIZE(100)
    REVERIFY(NO)
    ACSDEFAULTS(NO)
    PDSESHARING(EXTENDED)
    TRACE(ON)
    SIZE(128K)
    TYPE(ALL)
    JOBNAME(*)
    ASID(*)
    SELECT(ALL)
    OAMPROC(OAM)
```

RLSINIT

To use VSAM RLS, the SMSVSAM address space should be up and running. If you specify RLSINIT=YES, the SMSVSAM address space starts as part of your system initialization at IPL. If you specify NO, you need to start it later via the VARY SMSVSAM,ACTIVE command. The default is NO.

CF_TIME

Here you specify at what interval you want to collect SMF type 42 records containing information and statistics on CF cache and lock structures. The default is 3600 (one hour). Refer to “SMF record 42” on page 321.

DEADLOCK_DETECTION

This parameter specifies how frequently the dead lock detection routine is invoked to check for local and global deadlocks. The the local deadlock detection interval default is 15 seconds.

It also specifies the number of local deadlock cycles that must expire before global deadlock detection is run, as a one to four digit numeric value in the range 1-9999. The default is 4 cycles. Refer to “What is a deadlock?” on page 160.

RLS_MAX_POOL_SIZE

This is the maximum buffer pool size SMSVSAM can use in MB. The default is 100 MB.

SMF_TIME

For systems running DFSMS/MVS Version 1.3 or later, this keyword specifies whether DFSMS is to use SMF timing; that is, whether SMF type 42 records are to be created at the expiration of the SMF interval period, synchronized with SMF and RMF data intervals.

5.4.6 Security definitions

This section provides security definitions for access to the SHCDS data set and access to use VARY SHCDS command.

Access to SHCDS

You should provide update access for SMSVSAM to the SCHCDS data set. SMSVSAM should be authorized to update SYS1.DFPSHCDS.* data sets. If you protect SYS1.* data sets, be sure that the userid associated with SMSVSAM is able to access SYS1.DFPSHCDS.* for update.

You can issue the RACF command **RLIST STARTED SMSVSAM,ALL** to find out the userid associated with SMSVSAM.

Use the following RACF command to give SMSVSAM access to SHCDS:

```
PERMIT SYS1.DFPSHCDS.* ACCESS(UPDATE) ID(smsvsam_userid)
```

Figure 5-9 shows the error messages you will get due to SHCDS access failure.

Attention: If you do not provide the necessary RACF access authority to SMSVSAM it will fail to initialize.

Access to use VARY SHCDS command

To use the SHCDS command you should have should have access to the facility class **STGADMIN.IGWSHCDS.***. You can use the following RACF command to provide this access.

```
PERMIT STGADMIN.IGWSHCDS.* CLASS(FACILITY) ID(youruserid)
```

```
 ICH408I JOB(SMSVSAM ) STEP(SMSVSAM ) 744
    SYS1.DFPSHCDS.WTSCPLX2.VSBOX52 CL(DATASET ) VOL(SBOX52)
    INSUFFICIENT ACCESS AUTHORITY
    FROM SYS1.DFPSHCDS.* (G)
    ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
  IEF196I IEC161I 040(056,006,IGG0CLFT)-002,IEESYSAS,SMSVSAM,SYS00
  IEC161I 040(056,006,IGG0CLFT)-002,IEESYSAS,SMSVSAM,SYS00001,,,
  IEF196I IEC161I SYS1.DFPSHCDS.WTSCPLX2.VSBOX52
  IEC161I SYS1.DFPSHCDS.WTSCPLX2.VSBOX52
IEA794I SVC DUMP HAS CAPTURED: 760
  DUMPID=001 REQUESTED BY JOB (SMSVSAM )
  DUMP TITLE=COMPID=DF122,CSECT=IGWXSI20+0490,DATE=04/16/00,MAINT
          ID= NONE   ,ABND=0F4,RC=00000024,RSN=67510404
  RSN67510404 67510404
```

Figure 5-9 SMSVSAM errors due to SHCDS access failure

5.4.7 Using a VSAM sphere in RLS mode

When all the above actions are implemented, the system is ready for VSAM RLS processing.

To enable a VSAM sphere for RLS the following actions need to be taken:

- LOG parameter (NONE, UNDO or ALL) must be specified on the DEFINE CLUSTER or the ALTER CLUSTER command for the sphere. Or you may select one of the data classes containing such parameter.
- Option RLS in MACRF keyword in ACB (refer to “ACB control block” on page 235) needs to be set. There is also the option RLSREAD keyword in ACB where you indicate the type of read integrity (NRI or CR). Another way is using the keyword RLS= (NRI / CR) in the DD card defining the sphere.

Attention: Your RLS specification about read integrity on the ACB (RLSREAD) overrides the RLS specification on your JCL, if you specify in both places.

Figure 5-10 shows the messages presented in the MVS console when a job using RLS starts.

```
IEF403I MHL#64C5 - STARTED - TIME=22.10.03 - ASID=0027 - SC64
ICH70001I MHLRES2  LAST ACCESS AT 21:53:13 ON FRIDAY, MARCH 14, 2003
$HASP373 MHL#63C5 STARTED - INIT 2    - CLASS A - SYS SC63
IGW500I DFSMS CACHE CHARACTERISTICS FOR 705
VSAM COMPONENT NAME: MHLRES2.VSAM.RLSEXT.INDEX
DFSMS CF CACHE STRUCTURE NAME: RLS_CACHE
CI SIZE: 1536
CF CACHING SIZE: 2048
DFSMS DATACLASS NAME: RMMCDS
DFSMS RLSCFCACHE DATACLASS KEYWORD VALUE: ALL
IGW500I DFSMS CACHE CHARACTERISTICS FOR 706
VSAM COMPONENT NAME: MHLRES2.VSAM.RLSEXT.DATA
DFSMS CF CACHE STRUCTURE NAME: RLS_CACHE
CI SIZE: 4096
CF CACHING SIZE: 4096
DFSMS DATACLASS NAME: RMMCDS
DFSMS RLSCFCACHE DATACLASS KEYWORD VALUE: ALL
```

Figure 5-10 MVS messages for a JOB using RLS

5.4.8 RLS Recoverable spheres

A recoverable sphere has an associated log and in a problem appearance it can be processed for undo and also for redo. This attribute set in the catalog entry, by DEFINE/ ALTER LOG keyword, is referred to a VSAM cluster.

The LOG keyword has the following options: NONE, UNDO or ALL (UNDO plus REDO), with the following properties:

NONE means that the sphere is non-recoverable implying no logging. When this option is used neither backout nor forward recovery logging is performed. If either software or hardware damages the cluster, it will not be possible to recover the cluster.

The other two options (UNDO or ALL) imply that the accessing application uses a commit protocol allowing a recovery function through a log, in this case the CICS log.

UNDO means that transactional recovery is required. To support this option, the accessing application must support a two phase commit and backout protocol. During the life of a transaction, changes are protected by record level locks which prevent the changes from being seen by other applications that also support locking.

REDO means a forward recovery capability in the case of a lost or overlaid sphere. If you do not have an application that can perform forward recovery of the VSAM sphere, such as CICS/VR, there is no point in specifying LOG(ALL). It generates the overhead of creating the redo images of the records, but no ability to do anything with that information.

5.5 RLS problem determination and recovery

In this section we describe common problems with RLS and suggest some actions. There are a number of information APARs in IBMLINK that describe common problems in VSAM and what you can do resolve them. These APARs also tell you what documents you should collect before reporting the problem to IBM. Please review APAR II12927 as a starting point.

5.5.1 Problems with SHCDS

Please review “SHCDS considerations” on page 255 to ensure that you have taken care of those issues when you defined your SHCDS. Refer to information APAR **II13326** (“II13326 - Common problems with SHCDS” on page 444) for guidelines to diagnose SHCDS problems. Also review IBMLINK items **BDC000010564** (“BDC000010564” on page 451) and **BDC000007033** (“BDC000007033” on page 459).

5.5.2 Problems with SMSVSAM

You can get into SMSVSAM initialization problems due to various reasons. One of the most common causes is when SMSVSAM does not have access to your SHCDS. Information APAR **II12603** (“II12603” on page 465) gives you some tips to diagnose and recover from SMSVSAM problems. APAR **II12243** (“II12243” on page 467) tells you how to properly terminate SMSVSAM.

5.5.3 Problems with locks

Please remember that if an SMSVSAM is holding a lock only that SMSVSAM can release it. If you get into locking problems, even an IPL may not resolve the problem if the lock is held by another SMSVSAM on another system in the sysplex. IBMLINK has some information that will help you with lock problems.

IBMLINK entry **BDC000022923** (“BDC000022923” on page 468) describes a looping problem if you define a very large size for your lock structure. Item **RTA000141603** (“RTA000141603” on page 475) show how resolve IGWLOCK00 rebuild problems due to loss of connectivity to the sysplex.

5.5.4 SHCDS FALLBACK procedure

For some SHCDS errors, FALLBACK is the only way to correct the problem and to get the SMSVSAM to initialize successfully again. If you get repeated abends with the prefix “67” in the RSN code, you may have to rely on FALLBACK as the last resort. The following errors might require FALLBACK:

- Abend0F4 RC24 RSN675D0355
- Abend0F4 RC24 RSN67260989
- Abend0F4 RC24 RSN675F0398

Attention: If you have serious problems with SHCDS an IPL would NOT fix the problem because SHCDS name is remembered from one IPL to another

FALLBACK procedure is documented in z/OS V1R3.0 DFSMSdfp Storage Administration Reference. In a nutshell, FALLBACK reformats the SHCDS and therefore clears all the errors.

FALLBACK involves:

- Terminate all SMSVSAMs in the sysplex by issuing using the command: VARY SMS,SMSVSAM,TERMINATESERVER
- Then issue the command: VARY SMS,SMSVSAM,FALLBACK
- Reply Yes, to the WTOR that you are sure you want to FALLBACK
- Reactivate SMSVSAM using the command: VARY SMS,SMSVSAM,ACTIVE

  On the first system, you will be asked to specify 2 ACTIVE and 1 SPARE SHCDS.

To add an active SHCDS issues the command:

```
VARY SMS,SHCDS(SHCDS_name),NEW
```

To add a spare SHCDS issues the command:

```
VARY SMS,SHCDS(SHCDS_name),NEWSPARE
```

5.5.5 RLS rules

RLS enforces some rules when you have sphere open in RLS and non-RLS modes.

RLS open rules

- RLS OPENs for input/output fails, if the sphere is already opened for non-RLS output.
- RLS OPENs for input/output fails, if the sphere is already opened for non-RLS input, except if the sphere has been defined as SHAREOPTION(2,x). The non-RLS inputter does not have read integrity.
- RLS OPENs for output of a recoverable sphere by a batch client fails.
- RLS OPENs for a sphere which is either quiesced (or is quiescing) will be failed (QUIESCE=YES in catalog).
- RLS OPENs for a sphere that has not been assigned a CF cache via the SMS STORCLAS construct fails.
- Empty KSDSs - RLS allows opening of an empty KSDS without first loading the data set. In other modes (NSR, RLS) this is not possible.
- Positioning - RLS does not do implicit positioning to the beginning of the data set for SEQ processing. An explicit POINT is required.

Close and delete rules

- A RLS CLOSE is “not successful” if the SMSVSAM address space has recycled since the ACB was opened. One of the following messages are presented: IEC251I 16-0608 or IEC251I 16-0609.
- A Catalog DELETE deletes all retained or lost locks, if the owner is not an RLS subsystem. Refer to “RLS terminology” on page 250 to see the definition of an RLS subsystem. The DELETE does not delete CICS log records, because CICS is a subsystem.

5.5.6 MVS commands for RLS

Here are some commands to become familiar with. They have been changed or added to support RLS.

D XCF,STR,STRNAME=IGWLOCK00

Use this command to display the lock structure. See Figure 5-11 and Figure 5-12 for sample output of the command.

```
D XCF,STR,STRNAME=IGWLOCK00
IXC360I  19.59.09  DISPLAY XCF 097
STRNAME: IGWLOCK00
 STATUS: REASON SPECIFIED WITH REBUILD START:
           OPERATOR INITIATED
         DUPLEXING REBUILD
           METHOD        : SYSTEM-MANAGED
             AUTO VERSION: B8FC8279 72B01005
         REBUILD PHASE: DUPLEX ESTABLISHED
 POLICY INFORMATION:
  POLICY SIZE    : 28600 K
  POLICY INITSIZE: 14300 K
  POLICY MINSIZE : 0 K
  FULLTHRESHOLD  : 80
  ALLOWAUTOALT   : NO
  REBUILD PERCENT: 75
  DUPLEX         : ALLOWED
  PREFERENCE LIST: CF01     CF02
  ENFORCEORDER   : NO
  EXCLUSION LIST IS EMPTY
DUPLEXING REBUILD NEW STRUCTURE
 -------------------------------
  ALLOCATION TIME: 02/15/2003 11:16:05
  CFNAME         : CF02
  COUPLING FACILITY: 002064.IBM.02.000000010ECB
                     PARTITION: D   CPCID: 00
  ACTUAL SIZE    : 14336 K
  STORAGE INCREMENT SIZE: 256 K
  PHYSICAL VERSION: B8FC827A A4E7CF02
  LOGICAL  VERSION: B8FC73F0 F2A13441
  SYSTEM-MANAGED PROCESS LEVEL: 8
  XCF GRPNAME    : IXCL0001
  DISPOSITION    : KEEP
  ACCESS TIME    : NOLIMIT
  NUMBER OF RECORD DATA LISTS PER CONNECTION: 16
 MAX CONNECTIONS: 4
 # CONNECTIONS  : 3

 CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
 ---------------- -- -------- -------- -------- ---- ----------------
 SC63             01 0001006E SC63     SMSVSAM  0009 ACTIVE NEW,OLD
 SC64             03 00030079 SC64     SMSVSAM  0009 ACTIVE NEW,OLD
 SC65             02 0002007A SC65     SMSVSAM  0009 ACTIVE NEW,OLD
```

Figure 5-11   Display the lock structure

```
DUPLEXING REBUILD OLD STRUCTURE
-------------------------------
 ALLOCATION TIME: 02/15/2003 10:11:03
 CFNAME         : CF01
 COUPLING FACILITY: 002064.IBM.02.000000010ECB
                   PARTITION: E   CPCID: 00
 ACTUAL SIZE    : 14336 K
 STORAGE INCREMENT SIZE: 256 K
 PHYSICAL VERSION: B8FC73F0 F2A13441
 LOGICAL  VERSION: B8FC73F0 F2A13441
 SYSTEM-MANAGED PROCESS LEVEL: 8
 XCF GRPNAME    : IXCLO001
 ACCESS TIME    : NOLIMIT
 NUMBER OF RECORD DATA LISTS PER CONNECTION: 16
 MAX CONNECTIONS: 4
  # CONNECTIONS  : 3
```

Figure 5-12 Display the lock structure (continued)

D SMS,CFLS

This command displays the following information about the CF lock structure:

- Size
- Status
- Contention rate
- False contention rate

See Figure 5-13 for a sample output of the command.

```
-D SMS,CFLS
 IEE932I 236
 IGW320I 14:52:52 Display SMS,CFLS
 PRIMARY STRUCTURE:IGWLOCK00 VERSION:B8FC73F0F2A13441 SIZE:14336K
 RECORD TABLE ENTRIES:36258 USED:3
 SECONDARY STRUCTURE:IGWLOCK00 VERSION:B8FC827AA4E7CF02 SIZE:14336K
 RECORD TABLE ENTRIES:36258 USED:3
 LOCK STRUCTURE MODE: DUPLEXED
 System   Interval   LockRate   ContRate  FContRate   WaitQLen
 14:52:52 Display SMS,CFLS
 SC64     1 Minute        3.3     25.969      0.000       0.33
 SC64      1 Hour        84.0      4.797      0.416       0.10
 SC64      8 Hour        17.8      1.004      0.092       0.01
 SC64       1 Day         7.7      0.354      0.044       0.00
   (03)   1 Minute        2.2     17.714      0.000       0.19
   (03)    1 Hour        56.0      3.220      0.282       0.05
   (03)    8 Hour        11.4      0.671      0.059       0.00
   (03)     1 Day         4.5      0.236      0.024       0.00
 ***************** LEGEND ******************
   LockRate = number of lock requests per second
 ***************** LEGEND ******************
   LockRate = number of lock requests per second
   CONTRATE = % of lock requests globally managed
   FCONTRATE = % of lock requests falsely globally managed
   WaitQLen = Average number of requests waiting for locks
```

Figure 5-13 Sample output of D SMS,CFLS

D SMS,CFCACHE(structurename or *)

This command displays information about cache structures in the CF. See Figure 5-14 for a sample output.

```
-D SMS,CFCACHE(*)
 IEE932I 035
 IGW530I DFSMS CF STRUCTURES

 DFSMS CF CACHE STRUCTURE TO SYSTEM CONNECTIVITY

 SYSTEM         ==>  00000000011111111112222222222333
 IDENTIFIER     ==>  12345678901234567890123456789012

 RLS_CACHE           ++..............................
 ................    ................................

 SYSTEM  1 = SC64     SYSTEM  2 = SC65     SYSTEM  3 = ........
 SYSTEM  4 = ........ SYSTEM  5 = ........ SYSTEM  6 = ........
 SYSTEM  7 = ........ SYSTEM  8 = ........ SYSTEM  9 = ........
 SYSTEM 10 = ........ SYSTEM 11 = ........ SYSTEM 12 = ........
 SYSTEM 13 = ........ SYSTEM 14 = ........ SYSTEM 15 = ........
 SYSTEM 16 = ........ SYSTEM 17 = ........ SYSTEM 18 = ........
 SYSTEM 19 = ........ SYSTEM 20 = ........ SYSTEM 21 = ........
 SYSTEM 22 = ........ SYSTEM 23 = ........ SYSTEM 24 = ........
 SYSTEM 25 = ........ SYSTEM 26 = ........ SYSTEM 27 = ........
 SYSTEM 28 = ........ SYSTEM 29 = ........ SYSTEM 30 = ........
 SYSTEM 31 = ........ SYSTEM 32 = ........

 DFSMS CF CACHE                  TOTAL SPHERES   ACTIVE CACHE
 STRUCTURE STATUS:               CONNECTED:      FEATURE LEVEL:
 RLS_CACHE       = CF_ENABLED                -   -NONE-
 ................ = ............             -
 ................ = ............             -

 DFSMS CF CACHE FEATURE STATUS:
   A = SMS RlsCfCache Data Class values honored
```

Figure 5-14 Sample of CFCACHE display

Attention: If there is the name of the structure without the system names, it means that the structure is defined in SMS but does not exist in the CF.

D SMS,CFVOL(volid)

This command will display a list of coupling facilities cache structures that contain data for the specified volume (volid) and the status of the volume.

D SMS, MONDS(specmask or *)

This command will display the sphere specifications eligible for coupling facilities statistics monitoring. This monitoring is activated by the command; see “VARY SMS,MONDS(dsname{,dsname...}),ON|OFF” on page 276. You can specify a full or partial sphere name (specmask) to view a subset of the sphere specifications. You must specify at least one high level qualifier. A wildcard in the sphere name cannot be followed by additional qualifiers. Specify ‘*’ to display all the sphere specifications eligible for coupling facilities statistics monitoring.

D SMS,SHCDS

This command will display the following information about the sharing control data sets.

- Name
- Size
- Amount of free space for the active and spare SHCDS
- Whether the data set is usable

See Figure 5-15 for a sample output. Please note that it displays only the last two qualifiers of the SHCDS names. You should read it as *SYS1.DFPSHCDS.WTSCPLX2.VSBOX48* and so on.

```
-D SMS,SHCDS
 IEE932I 045
 IGW612I 19:01:43    DISPLAY SMS,SHCDS
 Name                      Size    %UTIL Status  Type
 WTSCPLX2.VSBOX48        10800Kb     4%  GOOD    ACTIVE
 WTSCPLX2.VSBOX52        10800Kb     4%  GOOD    ACTIVE
 WTSCPLX2.VSBOX49        10800Kb     4%  GOOD    SPARE
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
 -----------------           0Kb     0%  N/A     N/A
```

Figure 5-15 Sample output of D SMS,SHCDS

D SMS, SMSVSAM [,ALL]

This will display the status of the SMSVSAM server on this system or all the SMSVSAM servers and lock table connection status.

.

```
-D SMS,SMSVSAM,ALL
 IEE932I 049
 IGW420I DISPLAY SMS,SMSVSAM,ALL
 DISPLAY SMS,SMSVSAM - SERVER STATUS
   SYSNAME:  SC64       AVAILABLE ASID: 0009 STEP: SmsVsamInitComplete
   SYSNAME:  SC65       AVAILABLE ASID: 0009 STEP: SmsVsamInitComplete
   SYSNAME:  SC63       AVAILABLE ASID: 0009 STEP: SmsVsamInitComplete
DISPLAY SMSVSAM     - JOB STATUS
   SUBSYSTEMS CONNECTED:        0 BATCH:        3
DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)
   CONNECT STATUS:
     SYSNAME:  SC64     ACTIVE            RSN: 02010407 RbldNotActive
     SYSNAME:  SC65     ACTIVE            RSN: 02010407 RbldNotActive
     SYSNAME:  SC63     ACTIVE            RSN: 02010407 RbldNotActive
COMPOSITE STATUS:
     ORIGINAL STRUCTURE: NOT VOLATILE       NOT FAILURE ISOLATED
     NEW      STRUCTURE: NOT VOLATILE       NOT FAILURE ISOLATED
STRUCTURE STATUS:
     SYSNAME: SC64      Duplex
     SYSNAME: SC65      Duplex
     SYSNAME: SC63      Duplex
DISPLAY SMS,SMSVSAM - SMF RECORD 42 STATUS
                     SMF_TIME  CF_TIME ----- SUB-TYPE  SUMMARY -----
                                             15  16  17  18  19
   SYSNAME: SC64      YES- 3    1800- 3     YES YES YES YES YES
   SYSNAME: SC65      YES- 2    1800- 2     YES YES YES YES YES
   SYSNAME: SC63     *YES- 1    1800- 1     YES YES YES YES YES
   SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
*SMSVSAM SMF 42 RECORDS ARE WRITTEN FROM THIS SYSTEM.
    RECORDS ARE WRITTEN WHEN SMF GLOBAL INTERVAL EXPIRES.
DISPLAY SMS,SMSVSAM - GLOBAL CACHE FEATURE PARMLIB VALUES
   MAXIMUM CF CACHE FEATURE LEVEL   = Z
DISPLAY SMS,SMSVSAM - CACHE FEATURE CODE LEVEL VALUES
   SYSNAME:  SC63
    CACHE FEATURE CODE LEVEL        = A
   SYSNAME:  SC65
    CACHE FEATURE CODE LEVEL        = A
   SYSNAME:  SC64
    CACHE FEATURE CODE LEVEL        = A
   SYSNAME:  ........
    CACHE FEATURE CODE LEVEL        = .................................
CACHE FEATURE LEVEL DESCRIPTION:
   Z = No Caching Advanced functions are available
   A = Greater than 4K Caching code is available
```

Figure 5-16 Output of D SMS,VSAM,ALL

D SMS,SMSVSAM,QUIESCE

This will display the status of all active VSAM record-level sharing (VSAM/RLS) sphere quiesce events on the system that the command is entered. (This is not a Sysplex-wide command.)

SETSMS RLS_MAXCFFEATURELEVEL({A|Z})

This command specifies the method that VSAM RLS uses to determine the size of the data that is placed in the CF cache structure. If you specify A, caching proceeds using the RLSCFCACHE keyword characteristics that are specified in the SMS data class that is defined for the VSAM sphere. If you do not specify a value, or if you specify Z, then only VSAM RLS data that have a Control Interval (CI) value of 4K or less are placed in the CF cache structure. The default is Z. Refer to "Modifying the PARMLIB IGDSMSxx Member" on page 262 for more information.

SETSMS BMFTIME(nnnnn)

This command specifies that SMS is to allow nnnnn seconds (1 to 86399) to elapse between the production of SMS BMF records. The default is 3600 seconds.

SETSMS LRUCYCLES(cycles)

This will specify the maximum number of times (5 to 240) that the buffer management facility (BMF) least recently used (LRU) routine will pass over inactive buffers before making them available for reuse. While this parameter sets the maximum value, BMF will dynamically change the actual number of times it passes over inactive buffers.

SETSMS LRUTIME(seconds)

This command specifies the number of seconds (5 to 60) that the buffer management facility (BMF) will wait between calls to the BMF data space cache least recently used (LRU) routine. That routine releases inactive buffers in the BMF data space that are used to cache partitioned data set extended (PDSE) directory data. LRUTIME is related to LRUCYCLES. A change to the LRUTIME value introduced by this parameter will take effect on the next execution of the LRU routine. Most installations should use the default value. In some very high data rate situations you may want to tune this value. You should monitor the SMF 42 type 1 record to determine the amount of caching activity in the BMF data space. See z/OS MVS System Management Facilities (SMF) for information about the buffer management statistics recorded in SMF record type 42. The default value is 15 seconds.

SETSMS CF-TIME(nnn or 3600)

This command specifies the number of seconds between recording SMF 42 records related to SMSVSAM coupling facility use (subtypes 15, 16, 17 and 18). SMF_TIME(YES) overrides this parameter.

SETSMS DEADLOCK_DETECTION(iiii,kkkk)

This command specifies the deadlock detection intervals used by SMS.

iiii is a 1 to 4 digit numeric value in the range 1-9999 that specifies the length in seconds of the local deadlock detection interval. The default for iiii is 15 seconds.

kkkk 1 to 4 digit numeric value in the range 1-9999 that specifies the number of local deadlock cycles that must expire before global deadlock detection is performed. The default for kkkk is 4 local cycles.

VARY SMS,CFCACHE(structurename)

Issue this command to change the state of a cache structure and specify the name of the cache structure.

If you specify ENABLE, VSAM RLS data can be stored in cache structure. This is the normal state of operations and the state the coupling facility cache structure is in after sysplex IPL. If you specify QUIESCE, you cannot store any VSAM RLS data in the cache structure. The QUIESCE operation is not complete until the state of the volume is quiesced. Use the D SMS,CFVOL to determine the state of the volume.

VARY SMS,CFVOL(volid)

Use this command to change the state of a volume as it relates to coupling facility cache structures, specify the volume (volid). If you specify ENABLE, data contained on this volume can be stored in a coupling facility cache structure. This is the normal state of operations. If you specify QUIESCE, you cannot store any data from the volume on the coupling facility cache structure.

Note: If you specify QUIESCE, SMS may still select the volume during data set allocation. To stop SMS from selecting this volume, see “Changing the SMS Status of a Storage Group or Volume” on page 575.

VARY SMS,MONDS(dsname{,dsname...}),ON|OFF

This will specify the data set name (dsname) or data set names (dsname{,dsname...}). If you want to be eligible for coupling facility statistical monitoring, specify ON. Now, the commands are generic for the CF structures.

To indicate that the specified data set is no longer eligible for statistical monitoring, specify OFF. Monitoring is tracked through SMF record 42 subtype

16. You can specify a full or partial data set name with at least one high level qualifier. An asterisk cannot be followed by other qualifiers. You can specify up to 16 data set names with each command. This command affects activity for the specified data sets across all systems in the sysplex.

VARY SMS,SHCDS(shcdsname)

Issue this command to add or delete a sharing control data set (SHCDS), specify the name of the SHCDS. If you specify NEW, a new active SHCDS named (shcdsname) will be added. If you specify NEWSPARE, a new spare SHCDS named (shcdsname) will be added. If you specify DELETE, a SHCDS named (shcdsname) will be deleted. This SHCDS can be either an active or a spare SHCDS. **Note**: The sharing control data set (SHCDS) is identified by the dsname SYS1.DFPSHCDS.qualifier.Vvolser. When specifying its name (shcdsname) in this command, do not use the fully-qualified name. Use only *qualifier.Vvolser* as the shcdsname, without the SYS1.DFPSHCDS prefix, when specifying the name of the SHCDS.

VARY SMS,SMSVSAM, xxxxx

Use this command to manage SMSVSAM data sets or the SMSVSAM address space, specify one of the following parameters where xxxxx is:

- ACTIVE. Restarts the SMSVSAM server and re-enables the automatic restart facility for the server. This command will not function if the SMSVSAM address space was terminated with a FALLBACK command.
- SPHERE. Clears the VSAM-quiesced state for the specified sphere. Normally, this operation is done under application program control. This command is required only in rare circumstances.
- FALLBACK. Is used as the last step in the disablement procedure to fall back from SMSVSAM processing. For the SMSVSAM fallback procedure, see z/OS DFSMSdfp Storage Administration Reference.
- TERMINATESERVER. Abnormally terminates an SMSVSAM server. The server will not automatically restart after the termination. After some recovery action is complete, you can restart the SMSVSAM server with the V SMS,SMSVSAM,ACTIVE command. Use this command only for specific recovery scenarios that require the SMSVSAM server to be down and not to restart automatically.
- FORCEDELETELOCKSTRUCTURE. Deletes the lock structure from the coupling facility and deletes any data in the lock structure at the time the command is issued. You must reply to the confirmation message with the response, FORCEDELETELOCKSTRUCTURESMSVSAMYES, before the command takes effect. Use this command only in the event of a volume loss.

5.6 RLS enhancements

What follows is a list of RLS enhancements. The majority of them are introduced in z/OS DFSMS 1.3. Refer to “VSAM functions by release level” on page 2.

5.6.1 RLS/KSDS extended addressability

When VSAM RLS support was announced with DFSMS/MVS V1R3, it did not support VSAM KSDS extended addressability (EA) format. In other words VSAM RLS component retained the 4 GB architectural limit for the component size imposed by using the 4-byte field for the relative byte address (RBA) addressing mechanism. DFSMS/MVS V1R4 removed this restriction by supporting RLS EA for VSAM KSDSs, and therefore VSAM RLS now supports VSAM KSDSs up to the current multivolume limit of 59 DASD volumes.

As in VSAM non-RLS extended addressability, a VSAM RLS extended addressability KSDS must have its data class defined with:

```
Data Set Name Type =EXT

If Ext =P or R

Extended Addressability =Y
```

5.6.2 VSAM RLS CF structure duplexing and rebuild

VSAM RLS goes to lost locks state when the CF lock structure is lost and a system is also lost — a disruptive condition.

This process consists of the lock structure rebuild with all of the active VSAM RLS instances with the steps of allocating a new structure instance, propagating the lock data to the new structure, and switching over to using the new structure instance.

Currently, z/OS supports four types of lock structure rebuild processes. They are

- User-managed rebuild
- User-managed duplexing rebuild
- System-managed rebuild
- System-managed duplexing rebuild

User-managed rebuild

This enhancement to the current user-managed rebuild process will minimize the possibility of running out of record table space. Prior to this enhancement, if record space is exhausted, either an 0F4 abend or a record management error will occur. This enhancement adds a new validation check function during the user-managed rebuild process for the lock structure. When rebuild tries to change a lock structure and the size of the record table is greater than 50%, this new function will determine if the new table has enough space for all existing record table entries, plus enough empty space for future locking processes (120%). If the answer is no, then the rebuild request is rejected and a new IGW457I message will be displayed.

System-managed duplexing rebuild

This support is available from z/OS V1R4 DFSMS. The VSAM RLS duplexing support only applies to the lock structure and not the cache structures.

What is it?

This facility is a hardware-assisted mechanism for VSAM RLS duplexing CF lock structure data, and provides a robust recovery mechanism for failure scenarios such as a loss of a CF, or loss of connectivity to a single CF.

- Provides quick recovery from RLS lock structure failure
- Reduces lost lock conditions
- Minimizes running out of record table space
- Eliminates 0F4 abends

Attention: When the lock structure is in duplex mode, user command rebuild requests are rejected. You must use the alter function to change the lock structure.

How to set it up?

To use the new VSAM RLS system-managed duplexing function the following tasks are required:

Review CF configuration

Evaluate the CF configuration (storage, links, processor capacity) and make any necessary configuration changes to accommodate the new VSAM RLS lock structure instances resulting from system-managed duplexing.

Migrate to z/OS V1 R4 or later

All systems in the sysplex that use this function will need to be upgraded to this level that supports System-managed duplexing.

Install hardware CFCC level 12 or later

CF Duplexing requires CFCC Level 12 for implementation on IBM @server zSeries servers. For G5/G6 Servers and the stand alone CF 9672-R06, CF Duplexing function requires CFCC Level 11.

Attention: The lock structure (IGWLOCK00) should not be defined in the CFRM policy as duplexed before all of the VSAM RLS instances on all system have been upgraded to the level that supports System-Managed duplexing. (Once the lock structure is duplexed, down-level VSAM RLS instances that do not provide the support for System-managed duplexing will be unable to connect to duplexed lock structure).

Format the CFRM couple data set

Format this as system-managed, duplexing-capable, and bring that CFRM couple data set into use in the sysplex, to enable CFRM for system-managed duplexing.

Modify the CFRM policy

Modify this to control the sizing, placement (through the CF preference list) and the DUPLEX parameter indication for the VSAM RLS lock structure, which is duplexed via system-managed duplexing, and activating this CFRM policy.

5.6.3 RLS CF caching enhancements

This support is available in z/OS V1 R3.

What is it?

There are two RLS caching enhancements, respectively:

- Dynamic cache reassignment
- For the CF structure rebuild process to work, extra space in a CF must be reserved, if not it fails. Then all I/O requests for those VSAM spheres fail.
- DFSMS dynamic cache reassignment allows the customer to better utilize the CF resources.
- If the rebuild fails, SMSVSAM Dynamic cache reassignment code examines each VSAM sphere that is assigned to the failing cache. Those spheres that are assigned to a storage class that point to 'cache set' where there are more than one cache structure names are reassigned. The same algorithm used in the original allocation of the sphere is used. For the VSAM sphere that were able to be reassigned, all requests continue to work. For the VSAM spheres in which the reassignment failed, those requests are failed with a reason code “cache not available”.
- Data CIs larger than 4KB in cache structure. The initial version of SMSVSAM support did not cache RLS data CI that was greater than 4 KB, for performance reasons. However, since 9672 G4 machines, due to the faster CF links and CF CPU, there are performance gains in keeping larger than

4 KB data CIs in the CF cache structure. With this support you can cache all, some, or none of the VSAM RLS data in the CF cache structures defined to SMSVSAM — through the RLSCFCACHE keyword. The VSAM index structure is ALWAYS cached. The directory entry for a CF data entry also is always cached, regardless of what option is specified.

How to set it up?

Here are the steps to take.

Migrate to z/OS V1 R3 or later

You must be at this level or higher.

Update SYS1.PARMLIB member IGDSMSxx

There are two new keywords in IGDSMSxx. They are:

- RLS_DynamicCfCacheReassign(YES|NO).

  This keyword controls, at the sysplex level, whether RLS dynamic cache reassignment is active or not.

- RLS_MAXCFFEATURELEVEL({Z|A})keyword

  This specifies the method that VSAM RLS should use to determine the size of the data that is placed in the CF cache structure. You can use the RLS_MAXCFFEATURELEVEL keyword to limit the connect level when the sysplex has a mixed level of releases and maintenance. If you do not specify a value, or if you specify Z, then only VSAM RLS data that has a Control Interval (CI) value of 4K or less is placed in the CF cache structure. If you specify A, caching proceeds using the RLSCFCACHE keyword characteristics that are specified in the SMS data class that is defined for the VSAM sphere.

  RLS_MAXCFFEATURELEVEL is a sysplex wide value. The first system activated in the sysplex will set the value; all other systems will use the value set by the first system. Default: Z.

  – RLS_MaxCfFeatureLevel(A) indicates that VSAM RLS caching uses the characteristics specified in the SMS DATACLASS defined for the VSAM sphere.

  – RLS_MaxCfFeatueLevel(B) indicates that DynamicCfCacheReassignment is installed. This feature is active if the RLS_DynamicCfCacheReassign(YES) is specified in the active SMS parmlib member.

  – RLS_MaxCfFeatureLevel(AB) indicates that both features may be activated.

Update data class

The new caching characteristics are defined in the DATACLASS SMS construct. A new field is added to the ISMF DATACLASS construct called RLSCFCACHE. The values are NONE, UPDATESONLY, and ALL(default).

5.7 RLS performance

Even knowing that the major reason for implementing RLS is availability, in this topic we discuss the performance considerations caused by such migration, that is, from non-RLS to RLS mode.

Performance is a matter of subjectivity and relativity. In each performance evaluation task, we need to have very clear objectives and to what objectives we should relate the measured numbers. Here in this topic, we have two types of goals:

- Compare theoretically — although supported by IBM experiences — a CICSPlex® without RLS data sharing with the same environment but with RLS. Refer to “CICSPlex RLS performance comparison” on page 299.
- Compare practically (with experiments) batch workload accessing VSAM clusters in non-RLS and in RLS mode. Refer to “Batch RLS performance experiments and comparison” on page 300.

Our performance monitor is Resource Measurement Facility together with a hand full of MVS commands and SMF records reports related to RLS performance.

It is clear that one big difference in these two modes (RLS and non-RLS) is the use of the coupling facility (CF) by RLS, which implies performance losses and gains. The losses could be the time the transaction needs to wait for CF link transport and CFCC processing, for guaranteeing integrity, for example. The gains could be the use of CF caching capabilities to avoid an I/O operation, improving throughput.

In this topic we also explain major RMF fields covering RLS and the SMF records created on the subject.

5.7.1 Factors affecting RLS data sharing performance

Here, we cover all the factors introduced by RLS data sharing, which could affect the performance of a VSAM I/O request. Some of them are connected to specific ways of implementing data sharing. After the description of the factor, we recommend how to improve performance in the factor realm.

Speed of the CF hardware and software

RLS data sharing performance is directly affected by the speed of the hardware associated with the coupling facility (CF), as follows:

- Here is the relationship between the CF and host CPU speeds:
  - The faster the CF CPU, the less overhead in the host CPU. This is crystal clear, host CPU waits less.
  - The faster the host CPU, the higher the overhead, because for the time waiting for the CF CPU, MIPS are not being productively used by the host CPU.
- Dedicated or shared CF CPUs: Here we admit they are dedicated, as is strongly recommended in production environments.
- As for the number of CF CPUs in the CF logical partition, two is better than one.
- The CF type of link, such as IC, ISC or ICB, will effect the rate (MB/sec).

The other factor is the load submitted to the above mentioned hardware, for example:

- Total demand in the CF and CF links, which may cause high queue time.
- How heavy is the actual RLS request, which may cause high service time.

Refer to “Coupling Facility Usage Summary report” on page 311, and “Coupling Facility structure activity report” on page 314 to learn how to evaluate the performance of your CF hardware.

Recommendations

This recommendation is very easy to make. If you see performance problems in the CF through RMF analysis, buy the faster hardware for this CF:

- Faster and more than one (if needed) dedicated CF CPUs
- Faster links: ICs or ICBs, if possible
- Do not overload the CF and CF links capacity

DASD performance of the SHCDS data sets

The DASD performance of the SHCDS data sets is important, to have a good general RLS performance. Refer to “Define Sharing Control Data Set (SHCDS)” on page 253 to get more about the role played by this data set in SMSVSAM.

To be sure that the I/Os towards SHCDS is performing well, go through the explanations in “Performance scenario using RMF reports” on page 115.

Recommendation

Allocate them in faster controllers with lots of cache, small capacity disks, with high RPM, accessed by FICON channels

CF synchronous and asynchronous requests

A key XES question effecting performance, is what to do with the MVS CPU when CFCC is processing the request. There are two possibilities:

- With *synchronous processing*, the MVS CPU is placed in a loop by XES, waiting for the completion of the CF request. The synchronism referred to here is between the requiring task (not placed in wait state) and the execution of the request in the CF. Some observations on this are:
  - It is efficient because the requesting CP spins until response is received from CF, saving overhead of task being undispatched, saving status, swapping registers, being dispatched again, and so on.
  - This can burn MIPS if CF takes a long time to respond.

  This should be used for short requests.
- *Asynchronous processing*: Instead of holding the MVS CPU, XES releases it once the CF requests have been sent. Here are some observations on this approach:
  - This will consume less CPU time than very long running synch requests.
  - This frees up CPU for other work while the request is processed.
  - This uses instructions in the dispatcher component because the task must be undispatched then later re-dispatched.

  This should be used for long requests.

Usually the CF requests for the IGWLOCK00 are synchronous.

In z/OS 1.2 the *CF Synch/Asynch heuristic algorithm* is introduced, where XES selects the most efficient way (in terms of *used* machine cycles) of handling *each* request. This is based on:

- The current actual synch response times for each CF request type. For duplexed structures, XES keeps the time from the start of the first operation to the end of the last one.
- The speed of the CPU that z/OS is running on, which determines the number of instructions that could be executed while CP spins on synch request.

Recommendations

- Install z/OS 1.2 to have the new *CF* Synch/Asynch heuristic algorithm.

- Learn RMF reports covering synchronous and asynchronous requests, and refer to “Speed of the CF hardware and software” on page 283, to improve the performance of Synch/Asynch CF requests. By the way, you cannot influence the decision between one type and the other.

SMSVSAM buffering and caching

An application program GET can be served by SMSVSAM in three places: local buffer pools, SYSVSAM CF cache structure, or DASD I/O.

- *Local buffer pool*: The needed data CI is there and valid. No I/O is necessary.

Each buffer in this buffer pool has the size of a CI. It is possible to have 16 different types of buffer pools varying the CI size, from 2 KB to 32 KB.

These SMSVSAM buffers CI contents are managed by Buffer Management Facility (BMF) an SMSVSAM component, using a least recently used (LRU) algorithm. By the way, there is not a sequential look-ahead for any type of RLS access. LRU keeps in the buffer pool the CIs most referenced, or the best selection of CIs, as direct access is concerned. The number of buffers in the buffer pool varies dynamically, it increases when a buffer is needed by an incoming CI and decreases when a CI is deleted or stolen by the LRU algorithm. Time driven — the least referenced CIs are stolen by an LRU buffer aging routine. This routine runs with a dynamic frequency. At a higher frequency more CIs are stolen, less exact is their measured unreferenced pattern, and more CPU cycles are spent.

The full LRU algorithm is controlled by the Parmlib member IGDSMSxx, where the installation sets a goal limit for the local buffer pool size, through the RLS_MAX_POOL_SIZE keyword (defaults of 100 MB and limit of 1.5 GB). This parameter when you are migrating to RLS, should have the same magnitude of the size of the CICS/FOR (LSR and NSR) buffers. Also at IGDSMSxx (LRUTIME keyword), the installation defines the frequency of the LRU cycle routine.

At every invocation of the LRU cycle routine, it verifies if the local buffer pool size is:

- Equal or below the goal, then the buffer aging routine is slowed down.
- Over the goal, then the buffer aging routine is accelerated increasing the stealing of CIs. This state is named accelerated.
- Five times over the goal, or reaches the 1.5 GB limit, BMF energetically starts stealing (reclaiming) the buffers without waiting for the aging routine. This state is named reclaimed.
- If the local buffer pool appears as over the goal (status Accelerated or Reclaimed), you could adapt the goal in Parmlib member IGDSMSxx by increasing the RLS_MAX_POOL_SIZE value at a price of increasing the use

of central storage. If you do not have enough central storage, the option is to decrease the level of RLS activity. Also, this case guarantees that the CF cache structure is able to keep the records that overflow from local buffers in order to avoid DASD I/O.

In conclusion, we say that for direct access, RLS BMF manages its buffers more efficiently than local shared resources (LSR) pools. Then, the buffer pool hit ratios are better for the RLS managed buffers than for LSR pools, in direct accesses.

However, because the local buffer pools managed by BMF does not implement look ahead along read sequential access, avoid batch sequential RLS processing. If there is a reading JOB accessing the CICS/RLS spheres, it is recommended to use SHROPT(2,n) and no RLS mode.

- As for the SYSVSAM cache structure in the CF, it is used when the data CI is not in local buffer pool or is there but invalid, due to cross invalidation. No I/O is necessary.

To use the cache structure with data CIs, you may use at data class (DC) the RLSCFCACHE keyword to control who is going to the cache:

- For the ALL keyword, all data CIs read into or updated in local buffers are copied in CF cache structure. This is the default.
- For the UPDATESONLY keyword, only data CIs updated in local buffers are copied in CF cache structure.
- For the NONE keyword, no data CIs are copied to the CF cache structure. The CF cache structure is only used, through its directory for cross invalidation.

Note that this keyword is per DC, consequently each cluster can have a different value.

Before z/OS1.3 DFSMS only data CIs less or equal 4 KB were candidates to be located in the CF cache structure.

The CF cache structure is also managed by an LRU algorithm, trying to keep the most referenced CIs. However, there are the DIRECT WEIGHT and SEQUENTIAL WEIGHT keywords fields, designed to protect some clusters relatively from others against stealing. Refer to “Define the CF cache structures names to SMS” on page 260.

To increase the probability of finding the CI in this cache, you may allow big cache structures, at least the sum of all the local RLS buffers. We recommend a few cache sets, with each one pointing to only two CF same size structures in distinct CFs. Refer to “Define CF cache structures” on page 256. It seems that the use of weights is not fully implemented, because all the weights are forced to the value six.

- *DASD device*: The data CI is not in the cache structure.

RMF has a set of reports covering this subject. Refer to “RMF and VSAM RLS” on page 306.

Recommendations

- If the local buffer pool is accelerated or reclaimed, as shown in RMF, increase RLS_MAX_POOL_SIZE, if you have enough central storage. If you do not have such storage or increase the size of the cache CF structure or decrease the RLS load at measured interval. Also decrease the LRU_TIME keyword to the LRU routine to be invoked more often.
- Avoid batch reading sequentially in RLS mode, there is not look-ahead. Change the mode, if you want performance and you do not care about read integrity.
- If READ CF% in RMF is low, increase cache structures size, at least to the sum of all the local RLS buffers. We recommend a few cache sets, with each one pointing to only two CF same size structures in distinct CFs.
- Re-visit the value of RLSCFCACHE keyword, which the default is ALL. Maybe your I/O workload is cache unfriendly for reads. Then, you are polluting the cache with data that is not used later. Think about changing for OPDATESONLY.

Control Interval cross invalidation

Buffer coherency must be maintained when sharing data. There are two things that drive this coherency and they are:

- The SMSVSAM address space must to be updated to reflect that a CI located in its local buffer was updated by other SMSVSAM, that is, that the CI is no longer valid.
- There must be a place where any SMSVSAM can find the most current copy of a CI located in its local buffer pool

To address these problems the following CF logic is implemented; refer to Figure 5-17.

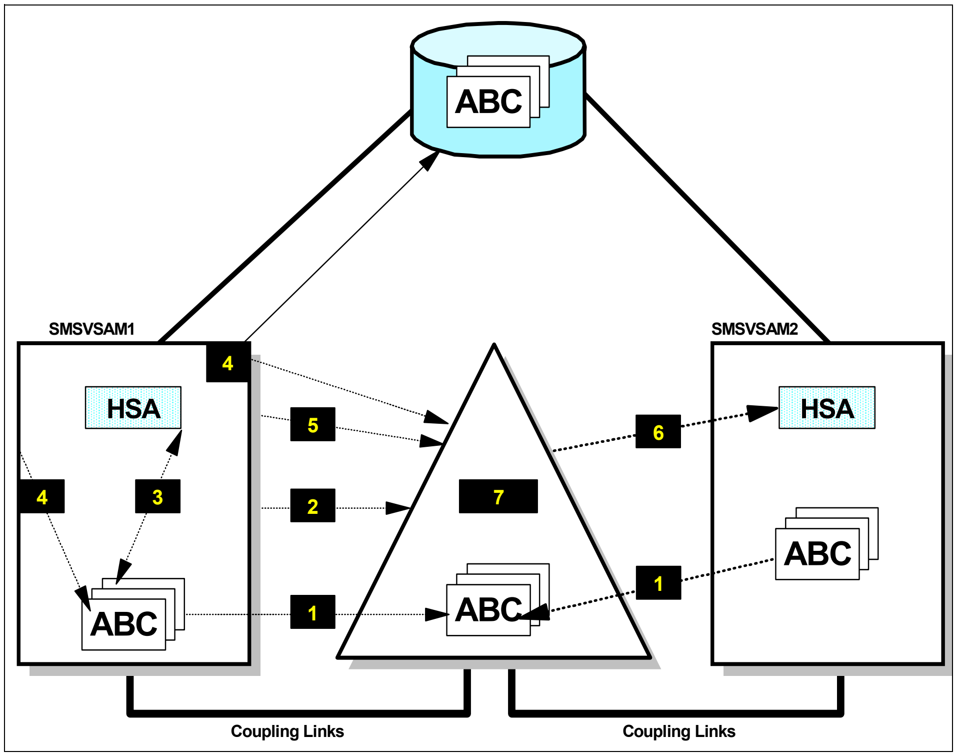


Figure 5-17 Cross invalidation and locks

The numbers below refer to the numbers in the figure:

- **1**. Every time a CI enters a local buffer pool, SMSVSAM indicates its interest on the CI to CFCC. So there are entries in the directory not pointing to a cache element, but to a specific SMSVSAM. The ratio between the number of directory entries and cache elements is determined by the first SMSVSAM to connect to the structure through the IXLCONN macro. The ABC control interval is in both local buffers.
- **2**. A transaction in system 1 is requiring an update to a record located in ABC control interval. The lock is required by SMSVSAM1 to CFCC and granted.
- **3**. SMSVSAM1 verifies that the copy in local buffer pool is valid by inspecting the HSA valid bit.
- **4**. The CI is modified in a local buffer pool, the CI is copied in DASD (store-through) and optionally in a CF cache element.
- **5.** and **6**. CFCC is informed by SMSVSAM1 about the change. By searching the directory, CFCC discovers the other SMSVSAMs that manifested interest in such CI — in other words, the ones that now have an out of date copy of the

ABC updated control interval. CFCC switches on the invalid bit accordingly, using IXLVECTR XES service. This action is called a *cross invalidation* (XI).

- There is an unusual case of XI called *directory reclaims XI.* It happens when a directory entry is needed and there is no one available. To reclaim an entry its previous content is released and the corresponding pair SMSVSAM/CI affected has the HSA invalid bit set to on. This is not because the copy of the affected CI is not valid anymore, but just because CFFC is not able to track it. RMF shows these figures in "Coupling Facility structure activity report" on page 314.

Later an SMSVSAM2 receives a transaction request, to be filled by the modified CI. The old copy is still in local buffer pool, but SMSVSAM2 detects the invalid bit on, using the IXLVECTR local vector service TestLocalCache. We have a cross invalidation and the SMSVSAM searches by the fresh copy in the CF cache elements or in DASD. When found, the new copy is brought to local buffer and made available to the transaction.

Recommendations

The "VSAM RLS Monitor III reports: RLSLRU" on page 306 shows the XI figures. If these figures are too big (above 10%), you may try to:

- Avoid directory reclaims XIs caused by CF cache directory shortages. Verify if the APAR OW23008 is applied. It causes SMSVSAM to constantly adjust the data-to-directory ratio to minimize the number of buffer invalidation due to a shortage of directory entries. In our experiments, it seems that this function is not active due to the huge amount (25%) of directory reclaims XI observed in the RMF report.
- Increase the size of the CF Cache structure to avoid that the directory reclaims XI cause an I/O DASD operation.
- Decrease the size of local buffer pools, this is trade off between buffer hits and cross invalidations.
- When using CICSplex dynamic workload balancing, temporarily make transactions connected to a certain type of data to bias the workload to a specific system.

True and false contention in lock structures

Usually we have many more records (many millions), in a medium to large data set, to serialize through global locks, than the number of entries (many thousands) in the CF structure lock table. The reason for this is the impossibility of having enormous lock structures in the CF.

The solution is to share the same lock entry with several logical records locks. Refer to Figure 5-18. A hashing algorithm applied in the Key, RBA, or RRN is used to map a lock associated with the record to a lock entry. When more than one lock register maps to the same lock table entry (synonyms), there is a false contention.

In the example, it is a false contention because the lock table entry #2 is held by an exploiter in SMSVSAM1 due to a logical record 2723 lock. Another exploiter in SMSVSAM2 requires a logical record 8627 lock. It cannot get the lock because the common entry is held by the first (false contention). In this case there is a performance penalty, the second lock requester is suspended until this SMSVSAM2 determines through XCF conversation (using the XCF group name IXCLO001) with SMSVSAM1, that there is no real lock contention on the resource. If the contention is true the requesting task still stays in wait state; if false it is placed immediately in ready state.

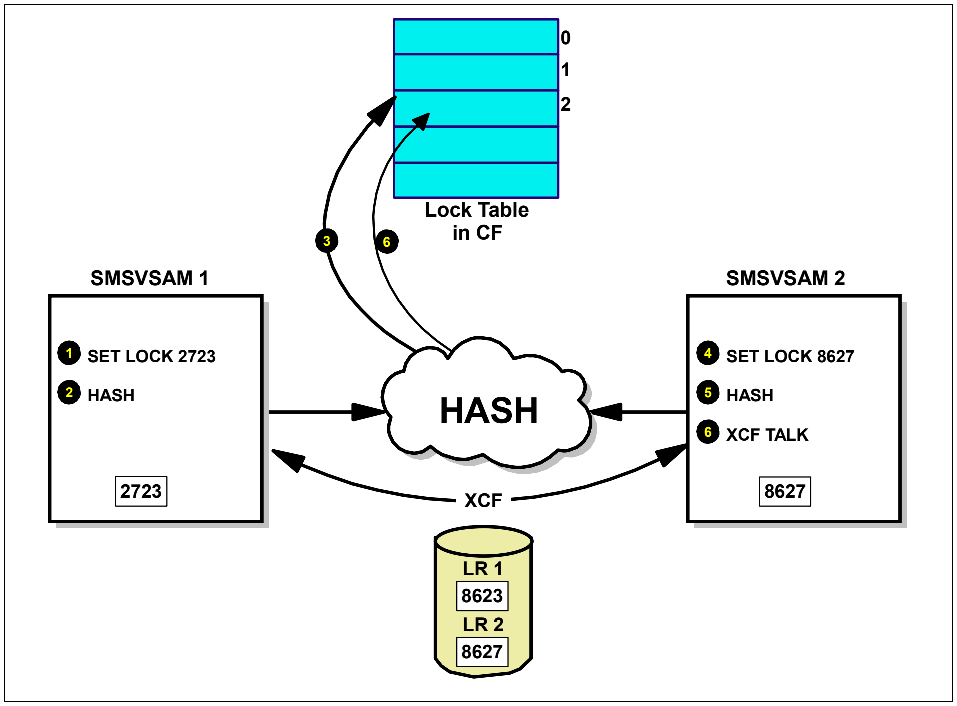


Figure 5-18   Example of two locks synonyms

To avoid such performance degradation due to false contention (should be less than 0.1% for iGWLOCK00), you might decrease the possibility of synonyms. This can be done in two ways:

- Decrease the length of each lock table entry, by decreasing MAXSYSTEMS (parameter used along formatting the Sysplex Couple data set) to a number 7 or less — if you do not have more than 7 systems in your sysplex. If you have more than 7, define MAXSYSTEMS as less than 24.

- Increase the size of the lock table. There is a formula to calculate its size:

  10M *number_of_systems *lock_entry_size

  The the loc_entry_size depends on the MAXSYSTEM value as we saw in the previous topic:

7 or less:	2 bytes
>= 8 and < 24:	4 bytes
>= 24 and <= 32:	8 bytes

Refer also to *OS/390 MVS Parallel Sysplex Configuration, Volume 2: Cookbook*, SG24-2076, for more information on how to derive the size of the structure.

If you are facing real contention (the rule of thumb is no more than 2% of the total locks requested) maybe you should, if possible, use *no ready integrity (*NRI) option in DD card or ACB. It implies that no shared locks are obtained for reads, meaning that the reader can read before the commit of an updater. This is sometimes referred to as dirty read, because the reader might see an uncommitted change made by another transaction. However, the buffer pool coherency is achieved, even with NRI. Another recommendation is to verify the existence of long transactions not committing and them holding many locks, they can flood the CF lock structure.

Refer to “RMF and VSAM RLS” on page 306 and “MVS commands about RLS performance” on page 320, if you are concerned about lock contention.

Recommendations

If you have more than 0.1% of false contention:

- Decrease the length of each lock table entry.
- Increase the size of the lock table.

If you have more than 2% for real or more than 0.1% for false contention:

- If possible, use NRI.
- Avoid long transactions running with scarce commits.
- Avoid rerunning a JOB applying duplicate updates instead of updating from the last commit point.
- Avoid backing out changes made by an application error (or bad/invalid output) in a specific job running in parallel with heavy workload against the cluster.

Serialization at record level

This function has no direct relation with the use of CF, however it is a key modification in the way VSAM serialize its data. In non-RLS mode, the level of serialization used by DFSMS lock manager is:

- At sphere level (at OPEN time), using share options.
- At CI level (at GET/PUT time) using intra-address space sharing.

Refer to "VSAM sharing mechanisms" on page 216, for more information.

Now returning to RLS mode, let us compare the pros and cons of serializing a logical record level with CI level.

The serialization at logical record level (instead at CI level) has the following important advantage:

- Significantly decreases the real contention due to a higher granularity.

And, here are the disadvantages:

- Increases the number of locks and the associated effort to manage them.
- Due to more locks, the number of false contentions should be higher than if the serialization would be done at CI level, because more locks are used.

Generally speaking, the serialization at logical record favors the smaller components.

However, pay attention that the cross invalidation process as described in "Control Interval cross invalidation" on page 287, is still done at CI level. Then the following scenario may happen:

Two different logical records in the same CI could be updated at the same time on two different SMSVSAM (X and Y), and this is good. X updated the first logical record in a CI holding its lock. The same CI is in the local buffer pool of Y. When Y tries to update the second logical record (also holding its lock), it detects that the buffer is now invalid, although the second logical record was not updated by X. The CI is reread by Y from DASD (or CF). The new CI, containing both updated records, is now written by Y to the CF and then to DASD, the invalid bit is set in X.

Please note that a logical record lock is only free at commit time and not at end of the logical processing.

Recommendations

Read the recommendations of "Control Interval cross invalidation" on page 287, if you are facing high level of real and false contentions. Do not forget the possibility of using NRI.

Effect of a write dominant I/O workload

Writes are always a source of performance, integrity, and availability problems in data processing. Nevertheless, because there's no such thing as a full read-only environment, let's explore this some more.

A write dominant I/O workload may cause performance degradation because of the following reasons:

- Excess of CI cross invalidations; refer to "Control Interval cross invalidation" on page 287.
- All the *random* writes against a CI in the local buffer pool are in store-through mode. This means that the DASD I/O operation is always executed, there is no option of a defer write (store-in mode). In CICS FOR there is such an option. On the other hand, the store-through mode favors a faster recovery.

  The performance gets worse, because:

  - The requesting task waits till the end of the DASD I/O operation.
  - If the same CI is updated *N* times, when in the buffer pool, with the store-through mode we have *N I/O* operations; but with defer write we have just *one* I/O operation, therefore saving *N-1 I*/O operations.
- Lots of locks held in exclusive mode (because of the writes), causing more real contention at logical record level.

Recommendations

- If you have to face lots of DASD I/O writes, then improve the performance of your volume, such as: allocate the VSAM cluster it in a fast controller with lots of cache, RAID10 (instead of RAID5), small capacity disks and high RPM disks. Refer to "Performance scenario using RMF reports" on page 115, to improve the performance of your DASD I/Os.
- Refer to "Control Interval cross invalidation" on page 287, for recommendations about how to decrease XIs.

RLS lock structure duplexing

An error in software or hardware may destroy the contents of a CF structure or can make it inaccessible due to the lack of CF links. IGWLOCK00 is sensitive to structure failures and its demands, to keep continuous availability, by implementing one of these features: *Failure Isolation* or *System Managed Duplexing*. Refer to "VSAM RLS CF structure duplexing and rebuild" on page 278*,* for more information.

In this topic, we discuss the performance aspects of System Managed (SM) duplexing the IGWLOCK00 structure.

Cost of SM duplexing differs depending on structure type, usage, and the percent of requests that get duplexed. When duplexing IGWLOCK00 all the writes are duplexed.

Performance impact of implementing SM duplexing is more likely to be a lack of capacity in the CF, than a transaction response time (in MVS CPU time component).

Here is a list of such costs:

- The MVS CPU cost: 3 to 4 times the CPU consumption of a simplex request (but may save CPU associated with DASD I/O). Refer to “CF access time” on page 295, to understand how this figure is derived.
- The subchannel cost: 6 to 8 times that of a simplex request (2 subchannels tied up, each 3-4 times longer than before).
- The link cost: Cannot directly measure, unlikely to be an issue CF CPU cost — 4 to 5 times that of a simplex request (across both CFs).

All of these assume minimal distance between the CPCs and CFs and between the two CFs.

There is no additional cost for simplex requests, because your system has the SM duplexing capability.

Recommendations

- Review the recommendations stated in “Speed of the CF hardware and software” on page 283, to improve the performance of the CF hardware in order to absorb the load generated by duplexing.
- If you do not have CF and CF Links capacity for implementing duplex with acceptable performance, then you may want to implement Failure Isolation instead.

Recoverable VSAM spheres

Recovery does affect performance due to extra work that needs to be done to allow a cluster to be recoverable. We strongly recommend, if you do not have the procedures in place and the recovery products, such as CICS/VR, to implement, for instance, forward recovery (REDO), there is no point in specifying LOG(ALL). This option generates the overhead of creating record images for UNDO/REDO in the forward recovery log, but does not provide the ability to do anything with such information. Refer to “Recoverable sphere” on page 252 for more information.

Recommendations

Do not declare your VSAM cluster recoverable, if you do not have products to enforce this recoverability.

Dealing with deadlocks

Deadlocks are a performance and also a problem determination subject. Refer to 3.2.16, “Deadlocks” on page 160. When in CICS RLS, messages start to pop up telling about transactions time out, one reason could be RLS VSAM deadlocks.

You may try to avoid deadlocks by changing the way programs are written, such as:

- Avoid to lock more than one logical records concurrently.
- Create a hierarchy of locks. Then, when more than one lock is required, they need to be required in a pre-determined sequence, as defined by such hierarchy.

Your installation may activate the deadlock detection routine through the IGDSMSxx keywords: DEADLOCK_DETECTION(nnnn|15,kkkk|4) and RLSTMOUT({nnnn|0}).

The default for DEADLOCK_DETECTION is (15,4). You may want to change these values to (1,1) to minimize the time required to detect a deadlock condition. Obviously there is a trade off between CPU time spent detecting deadlocks and the effect that deadlocks are causing to applications. Carefully monitor these effects in your environment before making changes to the supplied defaults.

Recommendations

- Avoid deadlocks by enforcing the NRI option, or implementing a lock hierarchy, or never asking for more than one lock concurrently.
- Activate the deadlock detection routine more often, if you are facing several deadlock situations.

5.7.2 CF access time

Now that we covered the majority of items affecting the RLS data sharing performance, we introduce a process to measure numerically some of those overheads.

As we said earlier in this chapter, the use of the CF by SMSVSAM implies in losses and gains. Here we start accounting the losses in MVS CPU in RLS caused by the access to the CF in a MIPS scale. Our calculation applies to our test machine, the 2064-1C7 CEC. You must scale to your processor's “per CPU” speed. One of the reasons for such a calculation is to help you to implement some CPU Capacity Planning due to migration from CICS/FOR to CICS/SMSVSAM.

However, before you get concerned about this MVS CPU overhead, we'll make a few comments:

- How much of a 0.5 seconds (500,000 microseconds) CICS response time is spent waiting on the CF? Maybe 500 microseconds (0.1%)?
- Even if you quadruple the CF time component, does anyone notice the extra 1.5 milliseconds?
- Remember that many large CICS customers still use DASD-only logging with response times around 2000 microseconds (2 millisecond) and still provide acceptable transaction response times.

The numeric approach has two parts: for synchronous and asynchronous requests, respectively:

Synchronous requests formula

Refer to "CF synchronous and asynchronous requests" on page 284. Here we state the rule-of-thumb, that the MVS CPU time overhead for such types of requests is made:

- Software CPU time which includes the exploiter (SMSVSAM) plus XES. It varies by structure type and exploiter and follows the rules of thumb:
  - Lock: Average 11 microseconds on 2064-1C7
  - Cache/List: Average 18 microseconds on 2064-1C7
- Hardware dwell CPU time which includes:
  - The time for the MSMG instruction in MVS CPU
  - The CF link transmission time (back and forth)
  - CFCC processing

  This time is reported in RMF CF Activity report as SYNC SERV TIME. Refer to "Coupling Facility structure activity report" on page 314, to see the report.

Asynchronous requests formula

The MVS CPU time overhead for such requests consists of:

- Software CPU time which includes the exploiter plus XES plus the code to execute a task switch:
  - Any type of structure (cache, list or lock): average 54 microseconds on 2064-1C7
- Hardware dwell CPU time is zero because the it asynchronous and the MVS CPU is not placed in a loop.

Lock contention event software CPU time

There is another type of MVS CPU overhead that occurs when we have lock contention. To find out if the contention is real or false, VSAM RLS instances talk through XCF paths. It averages: 360 microseconds on 2064-1C7.

SM Duplexing MVS CPU cost

MVS CPU cost: 3 to 4 times the CPU consumption of a simplex request.

So let's make the calculation for structure IGWLOCK00 as an example. The numbers are based in the report shown in Figure 5-22 on page 315.

Example: MVS CPU Time to access IGWLOCK00

Due to the fact that the requests to IGWLOCK00 are synchronous, we use the numbers shown in "Synchronous requests formula" on page 296. To get the hardware dwell time and the frequency of access, we look at "Coupling Facility structure activity report" on page 314, in the SC63 system.

- **Software time** = 11.0 microseconds.
- **Hardware dwell time** (SYNC SERV TIME) = 50.4 microseconds.
- **Frequency of access** = 215.6 locks /second
- **IGWLOCK00 structure consumption** = 215.6 x (11.0 + 50.4) = 13.23 milliseconds per second of MVS CPU 1C7.
- **Lock Contention using XCF** = 360 x Contention rate =

  360 x 44K/ 1200= 13.20 milliseconds per second of MVS CPU 1C7.
- The **RMF interval duration** where 44K contentions were found equaled 20 minutes or 1200 seconds.
- **Total MVS CPU TIME in IGWLOCK (no duplexing) =** 13.23 + 13.20 = 26.43 ms per second of MVS CPU 1C7, that is, 2.6% of one 1C7 CPU.
- **Total MVS CPU TIME in IGWLOCK (duplexing) =** 26.43 x 3.5 = 92.50 ms per second of MVS CPU 1C7, that is, 9.25% of one 1C7 CPU. Maybe you can convert to MIPS if it suits you.

5.7.3 RLS performance gains

Now here's the good news. The performance gains are coming from data sharing and the use of CF.

Balancing the system through dynamic workload distribution

In the case of CICS/RLS, the major performance objective (besides availability) is to balance the load between several MVS images sharing the same data and consequently to provide better overall response time and throughput.

The queuing theory shows that two images at 80% and 20% busy have a much longer average queue time than two images at 50% busy each. Then, in a controlled test environment, the VSAM “I/O time” (includes here the buffer and cache hits and the real I/O) may get worse (bigger) with RLS than without RLS, due to CF overhead caused by cross invalidation and locking. However, the caching function saves read access to DASD. Nevertheless, if the results are not that bad (after tuning) chances are that in a production environment the bottom line result implies a better performance, due to subsystems balance.

In a batch environment, throughput tends to be better, because a much larger level of parallelism is obtained with RLS. Before RLS data sharing, to guarantee integrity, just one updater could open the same cluster at a time. The serialization was at cluster level. With RLS the serialization is at record level and many updaters can, at same time, access concurrently the same cluster at different logical records.

Using the CF cache structure as a buffer

Due to the better load balance of the systems accessing a VSAM cluster in RLS mode and the inexistence of the FOR bottleneck a major rate of GET/PUT requests per second is expected. These GET/PUT requests can be served out of DASD, in local buffers and in the cache. The discussion here is to address whether or not this large rate causes an increased number of DASD I/Os per second.

The number of DASD I/Os is dependent on the local buffer hit ratio. The buffer pool hit ratio is dependent on the type of workload and the read/write ratio. Remember that each write causes an I/O. To avoid growth in I/Os when adding systems, you should scale the buffer pool and coupling facility cache size to maintain a constant hit ratio in the buffer pool, refer to “SMSVSAM buffering and caching” on page 285. If the cache size is set up correctly for a read load, one should experience a drop in the DASD I/O rate per transaction improving the overall performance. If the Coupling Facility cache is sized large enough to avoid reclaims, the XIs due to updates do not cause an increase in the DASD I/O rate, because the updated record can still be found in the CF. Because the RLS buffer manager manages its buffers more efficiently than local shared resource (LSR) pools, the buffer pool hit ratios are better for the RLS managed buffers than for LSR pools.

5.7.4 CICSPlex RLS performance comparison

The paramount question as far as CICS is concerned is: *What will happen with your transaction response time and throughput (number of transactions per second) when you migrate from a CICS/ VSAM non-shared environment to a VSAM RLS?*

As a general statement, we may say that running transactions under CICS TS in CICSPlex (multiple z/OS) accessing VSAM cluster in RLS mode, did not degrade transaction response time at all. Experiences have shown that response time remained the same and the total throughput was linear from one image, two images, three images, that is, it doubled and tripled from one to two and two to three images. What follows are some theoretical reasons supporting the above statements:

CICSPlex dynamic workload balance

Balancing the load across AORs located in different z/OS images in a Parallel Sysplex is beneficial for CICS performance in general. Read “Balancing the system through dynamic workload distribution” on page 298.

No more CICS function shipping

Let’s review the environment CICSPlex without RLS.

The CICS transactions coming from VTAM are welcomed by a TOR address space and based in WLM suggestion is sent to an AOR, where the transaction logic is executed by a program. If such logic requires a VSAM logical record, through a CICS function shipping mechanism (through XCF, if the FOR is remote), the request is sent to a FOR address space. FOR is in charge of accessing non-RLS VSAM spheres on behalf of CICS transactions.

However, function shipping has a price. IBM benchmarks suggest an increase of about 16% to 20% in the CPU time used by CICSPlex compared with just one system. This increase is due to function shipping between the AOR and the FOR.

With VSAM RLS such overhead vanishes, due to the fact that AOR passes the control to SMSVSAM in its own system. So, this gain may counterbalance the MVS CPU cycles for accessing the CF.

No file owning region bottleneck

Without RLS all the flow of CICS requests to the VSAM cluster is through file owning region (FOR). This address space is mono task and it can be a bottleneck, performance wise, and a single point of failure as availability is concerned. With RLS the FOR is replaced by several SMVSAM address spaces, one per system in the Parallel Sysplex. Then, as you see, RLS addresses both problems of performance and availability.

Avoiding remote two phase commit

Before SMSVSAM, it was possible for a unit of recovery to be formed by updates in VSAM data sets, accessed by different CICS FOR address spaces. In this case the synch point manager (CICS) needs to implement a remote two phase commit, a lengthy process, causing a transaction response time to increase. With SMSVSAM all the writes from the same recovery unit maybe executed in the same SMSVSAM eliminating this performance problem.

Avoiding I/O because of the CF cache structure

The RLS cache structures in the CF, may avoid a large number of read I/O operations. Refer to “Using the CF cache structure as a buffer” on page 298.

5.7.5 Batch RLS performance experiments and comparison

We made several measurements with batch jobs accessing VSAM clusters in RLS mode. The hardware and software description of our lab is in “Our test environment” on page 396. Here, we describe unique aspects related to the RLS experiments:

- A master file implemented in a KSDS cluster with 2M of logical records, freespace is (10,20).
- A sequential file with 200 K logical records not sorted by key. Causing 50% inserts and 50% reads in the master. There are no records with identical keys on purpose. This workload is clearly what the storage people call a “cache unfriendly” workload. The reasons for that are:
  - There is contention caused by locks not connected to logical records, as the locks are serializing CI and CA splits.
  - And the RLS behavior in a cache unfriendly (no re-visits) workload.
- The logic is to:
  - Read a record in the sequential file.
  - Randomly, try a key match in the master, with a GET:
    - If no match, insert the logical record in the KSDS (50%).
    - If it matches, just read the master record and do not update it (50%).

As we said, the keys in the sequential file were prepared to deliver 50% of matches. Then we have 100 K reads and 100 K insert writes. There are not update writes.

- After execution we have:
  - Data component: one extent, about 20800 CI splits and no CA splits
  - Index Component: one extent and no splits at all
- At every run the master is re-created again.

Test case description

1. One job in SMB mode processing 200 K records.
2. Two jobs in SMB mode in the same system processing 100 K records each, with read and write integrity (SHROPT=(1,3)), then one runs serially after the other.
3. Two jobs in RLS mode, in the same system processing 100 K records each No RLS tuning.
4. Two jobs in RLS mode, two systems processing 100 K records each. No RLS tuning.
5. Three jobs in RLS mode, three systems processing 100 K / 3 records each. No RLS tuning.
6. Three jobs in RLS mode, three systems processing 100 K / 3 records each. Some RLS performance tuning, including stopping duplex.

In all the described RLS experiences, the IGWLOCK00 is duplexed, with the exception of run 6. Always, the read integrity (CR option) is used.

Measurements of interest

- Number of EXCPs in the master KSDS cluster: In VSAM this figure is the same as the number of I/O operations or SSCHs.
- Total connect time in the master KSDS cluster: Less total connect time means less I/O operations (for direct access) and more efficiency in the executed I/O (for sequential access).
- Total CPU time: The programs running in the JOBs do not process the data, so the CPU time (TCB plus SRB) is only for GET/PUT processing.
- Elapsed time: The wall clock time to process all the 200 K records. However, we do not have a totally controlled environment, and chances are that the elapsed time may depend on other types of loads of the three logical partitions.
- In run 5, we capture other RMF metrics to tune RLS and prepare run 6. These metrics are described in "RMF and VSAM RLS" on page 306. The values we got are shown in "Tuning run 5 with RLS parameters" on page 304.

Table results

These are the numbers we got from the runs.

Table 5-1 Batch RLS results

Run Description	Number of EXCPs	Total Conn Time (sec)	CPU Time (sec)	Elapsed Time (sec)
1. One Job, one system	**280,276**	**62.518**	**19.3**	**204**
2. Two Jobs, one system, no RLS	**283,880**	**63.285**	**19.3**	**208**
3. Two Jobs, one system, RLS	**263,268**	**58.923**	**158.9**	**212 longest**
4. Two Jobs, two systems, RLS	**270,673**	**60.626**	**149.1**	**203 longest**
5. Three Jobs three systems, RLS	**278,005**	**80.893**	**153.1**	**160 longest**
6. Three Jobs, three systems, RLS, tuning	**270,804**	**78.281**	**113.4**	**145 longest**

Comments about runs 1 to 5

Here are some comments explaining the numbers captured for these runs:

- Run 1 was to calibrate run 2. Run 2 is important because when we move to run 3, we may see the difference caused by a local RLS (just one system). The difference between the numbers in run 1 and run 2 are meaningless and we have no further comments.
- Comparing run 3 with run 2, we see:
  - Less EXCPs and less connect time. For random access (as ours), this means that the savings in connect time are caused only by a decrease in the number of EXCPs. This decrease is caused by better local buffering (compared with SMB) and the existence of CF caching. However, pay attention that these gains are only for reads, because all writes in VSAM RLS forces an I/O operation.
  - Huge increase in the CPU time, that is the price we pay for better I/O. Remember that we do not process the logical record read from master on

purpose (read and forget mode). So, the CPU time (TCB plus SRB) is only consumed for the GET/PUT request. In RLS mode, there are more CPU time per GET/PUT request due to the CF access (refer to "CF access time" on page 295). This explains the high percentage figure in the CPU time increase. Do not forget that the IGWLOCK00 structure is duplexed in run 3, and this feature consumes more CPU time.

  – The elapsed time is almost the same. Here we have a conflict between two forces. One in run 3 due to parallelism and less I/O, and the other in run 2 due to less CPU (however, lots of CPU cycles are available) consumption and no lock contention. The final result, as elapsed time (an important metric for IT managers) is a tied game. Maybe some external effects, such as different external load in each run, may bias such metric. In comparison, we observe that the longest of the two JOBs is the one with more insertions (52841 against 47158) causing more delays due to CI splits. Then, we have a queue effect which makes the comparison a little unfair.

- Comparing run 4 with run 3, we see:
  - More EXCPs and more connect time. This increase is caused by worse local buffering and worse caching. The major contributor is the Cross Invalidation (XI) event, around 12% in run 4. In run 3 there are no XIs because there is just one local buffer pool.
  - Little decrease in the CPU time. The cache and the lock structure are slightly less used. This justifies the observed CPU decrease.
  - The elapsed time is almost the same. We may expect a little higher for run 4 due to slightly worse I/O and caching. Maybe some external effects as different external load in each run may bias such metric. In comparison, both the longest JOB is the one with more insertions, which makes the comparison fair.
- Comparing run 5 with run 4, we see:
  - More EXCPs and more connect time. This increase is caused by more unfavorable local buffering. The cache is used a little more than in run 4. The major contributor for less hits for reads in the local buffer pool is again the Cross Invalidation (XI) event. It is around 20% in run 5. The higher level of XIs in run 5 is caused by having three (instead of two) local buffer pools. Then increasing the chance of XIs.
  - Little increase in the CPU time. The cache and the lock structure are slightly more used. This justifies the observed CPU increase.
  - The elapsed time is less than in run 4. We do not have a clear explanation for such behavior. So, let's go traditional: maybe some external effects, such as different loads in each run, may bias such metric.

Tuning run 5 with RLS parameters

When running run 5 (three JOBs, three systems, RLS, no tuning), we observe other RMF measurements, that were not presented in Table 5-1 on page 302:

- BMF LRU algorithm accelerated at 75% of the measurement interval.
- Maximum BMF pool size was around 145 MB, the RLS_MAX_POOL_SIZE is 100 MB (default).
- Average I/O response time per GET/PUT was 0.001 second, including the requests served in cache and buffers.
- Data reads found in the BMF is 58.0%, index reads 97.0 percent.
- CPU time consumed by BMF to manage the buffers is 387.3 milliseconds.
- Data reads found in the CF is 10.0%, index reads is 2.6 percent.
- Data reads found in DASD is 31.8%, index reads is 0.5 percent.
- DASD device with four static PAV aliases allowing 528 IO/second with an average response time of 0.8 ms and average connect time of 0.5 millisecond.
- Cross invalidation was 20%, without directory reclaims XI.
- Real lock contention is 4.2% and less than 0.1 false contention in IGWLOCK00. This contention is not cause by logical records being accessed by two JOBs at same time (there are no repetition of keys in the sequential file), so the contention is caused by locks used to serialize splits.
- CF performance is outstanding with no queues at all.
- WLM service class had a current execution velocity of 85%, that is very good.
- CI Splits in all runs around 21K, and no CA splits.

Modifications

- Increase RLS_MAX_POOL_SIZE to 180 MB.
- Increase the size of the cache from 38 MB to 45 MB to improve the percentage of hits in the cache. We use the command:

  ```
  SETXCF START,ALTER,STRNM=RLS_CACHE,SIZE=45000
  ```
- Stop duplexing through the use of the following command:

  ```
  SETXCF STOP,RB,DUPLEX,STRNM=IGWLOCK00, KEEP=OLD
  ```
- Do not do anything about cross invalidation.
- Do not do anything to decrease the splits, we are only changing the parameters directly affecting RLS behavior.

Modification results

When performing run 6 (three JOBs, three systems, RLS, RLS tuning), we observe the following aspects in several RMF reports:

- BMF LRU algorithm is not increased.
- Maximum BMF pool size is around 168 MB (it was 145 MB), the RLS_MAX_POOL_SIZE is 180 MB. No paging is observed.
- Average I/O response time per GET/PUT was 0.001 second, including the requests served in cache and buffers. The accuracy of the measurement does not allow comparison with the previous value (also 0.001 second).
- Data reads found in the BMF is 62% (it was 58%), index reads 98% (it was 97%), this is good.
- CPU time consumed by BMF to manage the local buffers is 337.0 ms (it was 387.3 ms).
- Data reads found in the CF is 9.4% (it was 10.0%), index reads 1.7% (it was 2.6%).
- Data reads found in DASD is 28.8% (it was 31.8%), index read is 0.7% (it was 0.5%), this is good.
- DASD device with four static PAV aliases, allowing 451 IO/second (it was 528 IO/second) with an average response time of 0.6 ms (it was 0.8) and average connect time of 0.3 (it was 0.5 ms).
- Cross invalidation is 20% (it was 20%), without directory reclaims XI.
- Real lock contention is 3.8 (it was 4.2%) and less than 0.1 false contention in IGWLOCK00.
- CF performance was outstanding with no queues at all.
- WLM service class had a current execution velocity of 87.8% (it was 85%), that is very good.
- CI Splits in all runs around 21K (it was 21K), and no CA splits.
- Comparing run 6 with run 5, we see:
  - Less EXCPs and less connect time. This decrease is caused by better local buffering. Data reads in the BMF is 62% (it was 58%). Allowing less I/Os, 451 IO/second (it was 528 IO/second). The cache is used in the same amount as in run 5, even with a larger cache. It indicates no contention in the cache. Also remember that all the writes must have an I/O operation.
  - Medium decrease in the CPU time. The major reason for that is the lack of duplexing. Better buffering only decreases the CPU time in BMF code by 50 ms (387 - 337).

– The elapsed time is less because less I/O and less CPU consumption.

Final comments on our experiments

The results show that with a certain security margin, we may state that the introduction of VSAM RLS will not harm your performance. It may produce, even deliver, better performance. Compare the results of run 2 and run 6. Discarding the expected cost of CPU, we have a significant improvement in the elapsed time. Refer to “CICSPlex RLS performance comparison” on page 299 to be reassured that the same scenario could happen with CICS.

Using RLS mode in an ESDS organization

Usually the first implementations of RLS mode in customers are done in KSDS spheres. There are some concerns in exploiting RLS in other VSAM data organizations such as ESDS, RRDS. It was tested by IBM and works as described in manuals.

However, there is a performance recommendation for ESDS, to avoid heavy direct insert rate (at end of the ESDS). Because of integrity reasons, a lock is held during the I/O operation for inserts. Depending on the rate, we may observe the building up of a queue. Before planning to access ESDS or RRDS in RLS mode, we recommend that you refer to APAR OA01932.

5.7.6 RMF and VSAM RLS

Refer to “Resource measurement facility” on page 136 for more information on this product. There are several RMF reports picturing the VSAM RLS performance such as Monitor III: RLSLRU, RLSSC and RLSDS. There are also the traditional CF reports that describe generically the CF structures and their links. There we look for information about the SYSVSAM structures, such as IGWLOCK00 and the RLS cache structures.

We now describe each of these reports, stressing the keys aspects towards performance management of each field.

VSAM RLS Monitor III reports: RLSLRU

The RLSLRU report provides Local Buffer Manager LRU statistics for each system. The data in this report can help you in adjusting the goal / limit for the local buffer pool. Before going through this report, which pictured in Figure 5-19 on page 308, refer to 5.1.3, “How does RLS work?” on page 244.

So first, let’s go to the fields in RLSLRU report, that need additional explaining:

- *Avg CPU Time:* This is the average CPU time spent by BMF LRU processing during each reporting interval in milliseconds. We guess that this field could be used for comparison along different RLS workloads. These workloads can be more or less buffer friendly, or to produce more cross invalidation causing a change in CPU consumption.

- *Buffer Size Goal*: This is the buffer size goal (bytes) as stated in IGDSMSxx RLS_MAX_POOL_SIZE. If goal is greater than 1.5 GB, then the word MAX is displayed.
- *Buffer Size High:* This is the buffer size actual high value (bytes). Cursor-sensitive control on a system line displays a pop-up panel with buffer counts by pool for the selected system. There are sixteen storage pools (2K, 4K,..., 32K) available. Each one is presented with the low, high and average number of buffers.
- *Accel%:* This is the percentage of Buffer Manager LRU cycle routine, when local buffer pool size is over the goal and buffer aging algorithms were accelerated.
- *Reclaim%:* This is the percentage of Buffer Manager LRU cycle routine, when BMF was five times over the goal and buffer aging algorithms were bypassed to reclaim buffers.
- *BMF Read%:* This is the percentage of GET hits in local buffer pool, the CI is in local buffer pool and *valid*. Refer to “Control Interval cross invalidation” on page 287 to understand the meaning of valid. The recommendation here is around 80% for direct. Less than this, you should evaluate the level of cross invalidation, if it is OK, maybe we suggest to increase your local buffer pool maximum size, if have enough virtual storage. If it is not OK refer to the recommendations displayed in “Control Interval cross invalidation” on page 287. Chances are that your workloads be cache unfriendly with re-visits to the same data.
- *Cache Read%:* This is the percentage of GET hits in the CF cache structure. Recommended value around 15%, if less maybe to increase the size of the CF cache structure.
- *DASD Read%*: This is the percentage of GET I/O requests in DASD.

```
 RMF V1R2   VSAM LRU Overview  - SANDBOX        Line 1 of 3
 Command ===>                                                Scroll ===>
CSR

 Samples: 100     Systems: 3    Date: 03/12/03  Time: 13.33.20  Range: 100
Sec

 MVS      Avg CPU  - Buffer Size -  Accel  Reclaim  ------ Read -----
 System    Time      Goal   High      %       %     BMF%   CF%  DASD%

 SC63      13.08      100M    31M     0.0     0.0    33.3   0.4   66.4
 SC64      8.441      100M    23M     0.0     0.0    68.2   0.1   31.6
 SC65     11.95       100M    30M    0.0      0.0    33.6  0.4   66.0
```

Figure 5-19 RLSLRU report

VSAM RLS Monitor III reports: RLSSC and RLSDS

RLSSC and RLSDS reports show the same fields. The difference is the scope. RLSSC groups data in storage classes and RLSDS groups it in VSAM data sets terms. Both reports cover the use of local buffers and CF cache. Refer to “How does RLS work?” on page 244.

To gather the data about data sets the option VSAMRLS must be set in Monitor III ERBRMFxx Parmlib member:

```
VSAMRLS ADD(data set name mask)

DELETE(data set name mask)
```

This option controls the collection of VSAM RLS activity data. When you specify VSAMRLS or allow the default value to take effect, activity data is gathered for VSAM RLS by storage class. In addition, data set masks can be specified to collect data by VSAM spheres, too. To suppress the gathering of VSAM RLS data, specify NOVSAMRLS.

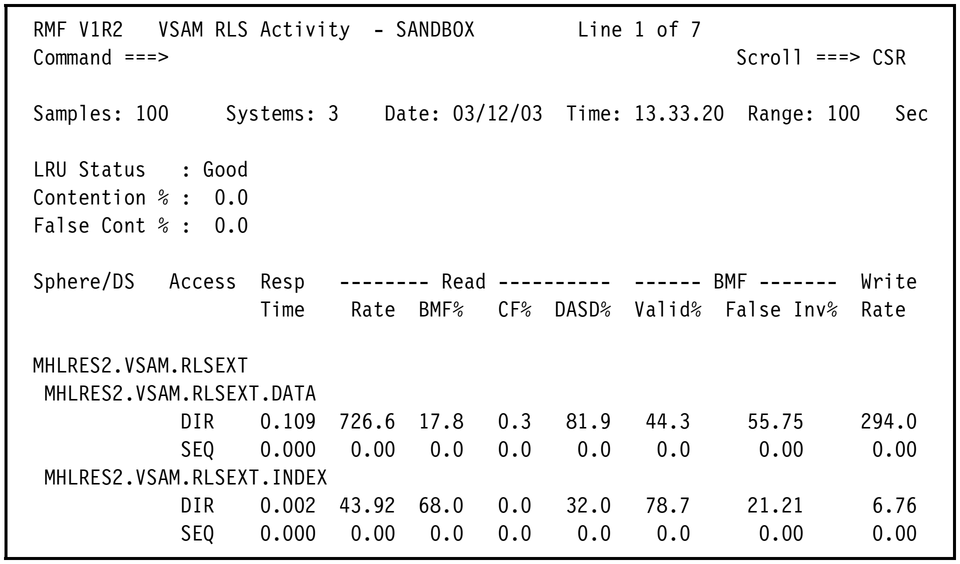

```
RMF V1R2   VSAM RLS Activity  - SANDBOX        Line 1 of 7
Command ===>                                                Scroll ===> CSR

Samples: 100     Systems: 3    Date: 03/12/03  Time: 13.33.20  Range: 100   Sec

LRU Status   : Good
Contention % :  0.0
False Cont % :  0.0

Sphere/DS   Access  Resp   -------- Read ----------  ------ BMF -------  Write
                    Time    Rate  BMF%   CF%  DASD%  Valid%  False Inv%  Rate

MHLRES2.VSAM.RLSEXT
 MHLRES2.VSAM.RLSEXT.DATA
             DIR    0.109  726.6  17.8   0.3   81.9   44.3     55.75     294.0
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
 MHLRES2.VSAM.RLSEXT.INDEX
             DIR    0.002  43.92  68.0   0.0   32.0   78.7     21.21      6.76
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
```

Figure 5-20 RLSSDS

The collection of VSAM RLS activity data by VSAM spheres can be controlled by following sub options:

- ADD: Start collection for all VSAM data sets which are covered by the mask.
- DEL: Stop collection for all VSAM data sets which are covered by the mask.

If by any reason, you stop and start SMSVSAM, then RMF Monitor III Data gatherer (RMGAT) must be restarted to capture VSAM RLS data again.

These reports provide VSAM RLS activity regarding GET and PUT requests accessing the local buffers, the CF cache structures and DASD. This data might help you in answering important questions like:

- Are there problems with Least Recently Used algorithms (LRU) or buffer pool sizes?
- Are the CF cache structures too small?

So now, let's go to the fields presented in Figure 5-20 for additional explaining:

- *LRU status:* This indicates the status of local buffers controlled by BMF. SMSVSAM, where the LRU algorithm is used. The size of this buffer pool is dynamically trimmed by SMSVSAM accordingly with installation parameter RLS_MAX_POOL_SIZE in IGDSMSxx member in Parmlib. Refer to 5.1.3, "How does RLS work?" on page 244.

- We have three statuses:
  - *Good*. The local buffer pool size is below its goal on all systems.
  - *Accelerated*. The local buffer pool size is over the goal on at least one system, and the buffer aging algorithms are accelerated. It means practically that the buffer pool is loosing its efficiency and the CIs are existing less time in such buffers. Maybe it is time to think about increasing the local buffer pool, if you have enough central storage to avoid paging. If you do not have enough central storage, let's hope that at the next level of buffering, the cache structure will do the job of avoiding I/O.
  - *Reclaimed*. The local buffer pool size is over the goal on at least one system, and the buffer aging algorithms were bypassed to reclaim buffers. The same as accelerated, but much more dramatic.
- *Contention%:* This is the percentage of true lock contentions, the percentage of requests issued by exploiters delayed due to real contention on a lock in relation to the total number of requests. It does not include false contention. The recommendation is less than 2.0%. Refer to "Coupling Facility structure activity report" on page 314, to see other RMF reports showing such contention.
- *False Contention%:* This is the percentage of false LOCK contentions, the percentage of requests issued by exploiters delayed due to false contention on a lock in relation to the total number of requests. The recommended value here is around 0.1% (for IGWLOCK00). Refer to "Coupling Facility structure activity report" on page 314, to see other RMF reports showing such contention.
- *Resp Time:* This is average I/O response time of all I/O requests generated by the line portrayed entity, such as: storage class, data set, system (cache structure) in seconds. It includes IOSQ time, pending time, disconnect and connect time.
- *Read:* This describes the behavior of GET I/O operations. Shows the rate and the distribution of where the GET is served: local buffer pool, SYSVSAM cache structure in the CF, or DASD. Follows the sub-headings:
  - Rate: This is the average number of logical records read through a GET, per second.
  - BMF%: This is the percentage of GET hits in a local buffer pool, the CI is in local buffer pool and *valid*. Refer to "Control Interval cross invalidation" on page 287 to understand the meaning of valid. Pay attention to the fact that sometimes the CI is invalid because another logical record (not the one required) was updated by another SMSVSAM. The recommendation here is around 80% for direct. If it's less than this, you should evaluate the level of cross invalidation. If such a figure is normal, we suggest that you

  increase your local buffer pool maximum size, if have enough virtual storage.
  - CF%: This is the percentage of GET hits in the CF cache structure. The recommended value is approximately 15%; if less, increase the size of the CF cache structure.
  - DASD%: This is the percentage of GET I/O requests in DASD.
- *BMF:*
  - *VALID%:* This is the percentage of BMF GETs hits that were valid. If a buffer is found in the local cache and is determined to be valid according to the information in local control blocks, this counts as a BMF valid READ hit.
  - *FALSE INV%:* This is the percentage of READ requests when the copy in the BMF local cache was invalid, because the coupling facility has lost track of the integrity status of the buffer.
- *WRITE RATE:* This is the total number of BMF PUTs requests per second.

Coupling Facility RMF Post Processor reports

These reports are created by RMF monitor III data gatherer in an SMF record format. Later these records are processed by the Postprocessor generating the sysout reports. There are also CF reports online produced by RMF Monitor III data gatherer and shown under TSO Monitor III data reporter. These reports are not covered here, because their fields are a repetition of the Post Processor fields.

The CF Postprocessor reports are generic showing the performance aspects of all the CFs and all the structures. Here, we only covered the fields and the examples about RLS structures. The reports are modified on purpose to highlight such structures only. For a full description of all fields, we recommend reading *OS/390 MVS Parallel Sysplex Capacity Planning*, SG24-4680.

Coupling Facility Usage Summary report

This report shows the use of the CF in number of requests, storage and processor. You can use this section to evaluate the status of all structures and how the CF as a whole is being utilized. Refer to Figure 5-21.

Structure Summary

The structure summary lists all structures ordered by type and name. For each structure the status at the end of the RMF data collection interval is presented:

- ACTIVE: Storage is allocated and there is connection to at least one system. There is also the information about duplexing such as primary (PRIM) or secondary (SEC).

- INACTV: Storage is allocated but there is no connection to any system, these structures are called persistent. IGWLOCK00 is persistent.
- UNALLOC: The structure was active at some point in time during the interval and did not exist at the end of the RMF interval.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                                PAGE   1
        z/OS V1R4              SYSPLEX SANDBOX             START 03/15/2003-15.10.00          INTERVAL 000.20.00
                               RPT VERSION V1R2 RMF        END   03/15/2003-15.30.00          CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF02
    TOTAL SAMPLES(AVG) =   599  (MAX) =   600  (MIN) =   599
   ------------------------------------------------------------------------------------------------------------------------------
                                               COUPLING  FACILITY  USAGE  SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------
    STRUCTURE SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------

                                                     % OF               % OF      AVG     LST/DIR  DATA      LOCK     DIR REC/
           STRUCTURE                        ALLOC    CF            #    ALL       REQ/    ENTRIES  ELEMENTS  ENTRIES  DIR REC
    TYPE   NAME              STATUS CHG     SIZE     STORAGE     REQ    REQ       SEC     TOT/CUR  TOT/CUR   TOT/CUR  XI'S


    LOCK   IGWLOCK00         ACTIVE          14M         1.5   767562    32.7    639.63       36K        0      2097K      N/A
                              SEC                                                               2        0         85      N/A

    CACHE  RLS_CACHE         ACTIVE  X       10M         1.0     1499K   63.8    1249.1       25K      778        N/A     254K
                                                                                              25K      393        N/A     254K
    PROCESSOR SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------

    COUPLING FACILITY         2064       MODEL 1C7      CFLEVEL  12

    AVERAGE CF UTILIZATION  (% BUSY)           3.7      LOGICAL PROCESSORS:   DEFINED   2    EFFECTIVE   2.0
```

Figure 5-21   CF Usage Summary report

For each active structure, the section shows the allocated structure size, the total number of requests (issued by the exploiter to the XES API for the structure), and the average number of requests/second.

There is also the last four columns in the report. The information in these columns can be used to validate the size of various types of structures. The TOT/CUR (total/current) in-use values indicate for each entity described in the column how many of those are allocated (TOT) and how many are in use (CUR) at end of the RMF interval, so it is not a high water mark. Some knowledge of the structure is required to interpret these data.

- **Cache structures (as RLS_CACHE)**
  - *DIR ENTRIES* informs how many cache directory entries are allocated/in-use. In the report you may see that the RLS_CACHE directory is full with 25K entries.
  - *DATA ELEMENTS* informs how many cache elements (data) are allocated/in-use. Here you can see any mis-proportioned in terms of entry-to-element ratio (one never fills up but the other does). In the report

you may see that the RLS_CACHE has 778 elements allocated and 393 used.

- The *DIR RECLAIMS* can be used as an indicator whether the directory of a cache structure is over-committed. Whenever a shortage of directory entries occurs, the CF reclaims in-use directory entries associated with unchanged data. These reclaimed items are used to satisfy new requests for directory entries by the database manager. All users of the data item represented by the directory entry must be notified that their copy of the data item is invalid. When the database manager needs access to the now invalidated data item, the item must be reread from DASD and registered with the CF.

Avoid *Directory Reclaims,* this activity results in increased I/O activity to the cluster to reacquire a referenced data item. To reacquire data items and to re-register them to the CF may result in increased CPU utilization and prolonged transaction response times whenever a read miss occurs in the local buffer pool.

Directory reclaims can be eliminated by increasing the number of directory entries for a particular structure. This can be accomplished by:

- Increasing the size of the structure. For structures as IGWLOCK00 with data elements and directory entries, both increases in the ratio are specified by SMSVSAM.
- Changing the proportion of the structure. Some cache structure exploiters allow the installation to specify the ratio of directory elements to data elements (DB2, IMS). IGWLOCK00 does not have this capability, it is said that it constantly adjusts the data-to-directory ratio to minimize the number of buffer invalidation.

- **Lock structures (IGWLOCK00)**

  LST ENTRIES informs how many lock lists (point to the lock data) are allocated/in-use. In the report you may see that the IGWLOCK00 is almost totally empty with 36K allocated and only 2 used.

  LOCK ENTRIES contains record data entries. In the report you may see that the IGWLOCK00 has 2097K record data allocated and only 85 in use.

  However, the critical data about lock structures you may find at the FALSE CONTENTION data for the lock structure in “Coupling Facility structure activity report” on page 314.

Processor Summary

The processor summary section shows the CF model and version.

Average CF Utilization: This indicates the real CP utilization (discarding the CFCC ethernal loop effect) of the LP that hosts the CF.

Logical processors:

- Defined: How many logical processors are defined to this CF.
- Effective: Number of effective available logical processors in a shared environment. CFCC measures the time of real command execution as well as the time waiting for work. The reported value shows the ratio between the LPAR dispatch time (CFCC execute and loop time) and the RMF interval length.

Coupling Facility structure activity report

The structure activity report in Figure on page 314 presents, for all structures:

- In the REQUESTS columns, the number of synchronous (SYNC), asynchronous (ASYNC) and changed (CHNGD) requests (refer to “CF synchronous and asynchronous requests” on page 284) and the average service times and standard deviation for these requests (SERV TIME(MIC)). The CHNGD line applies to the synch requests changed to asynch because the subchannel was busy.

  The service times of approximately 50 microseconds shown in the report indicates a fast CF hardware.
- In the DELAYED REQUESTS columns, the number requests that were delayed due to the following conditions:
  – NO SCH: Subchannel contention. The number of *asynchronous* requests that were delayed due to subchannel busy conditions.
  – PR WT: Peer subchannel wait contention. Applies to duplexed requests, which always require two subchannels. Number of requests a subchannel for the operation targeted to the peer (secondary) structure was not available. Only in the primary structure. Always about 100% for secondary monitor delay times.
  – PR CMP: Waiting-for-peer-completion contention. Applies to duplexed requests, which always require two subchannels. Number of requests that one of the two duplexed operations has completed, but the completed subchannel remains unavailable for use until the peer operation completes. It shows disparity in response times of CFs.
  – DUMP: Waiting for the DUMP structure serialization. The contents of the other are dumped in this structure, to reduce the serialization time of the structure being dumped.

The /DEL field indicates the average delay time for each delayed request, and /ALL the average delay time taking in consideration all the requests (including the ones which do not suffer delay).

The EXTERNAL REQUEST CONTENTIONS gives information on lock contention for serialized list and lock structures, as well as data access statistics for cache structures. A discussion for the SMSVSAM structures follows:

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                PAGE  13
        z/OS V1R4               SYSPLEX SANDBOX             START 03/15/2003-15.10.00           INTERVAL 000.20.00
                                RPT VERSION V1R2 RMF        END   03/15/2003-15.30.00           CYCLE 01.000 SECONDS

    ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF01
    ------------------------------------------------------------------------------------------------------------------------------
                                              COUPLING  FACILITY  STRUCTURE  ACTIVITY
    ------------------------------------------------------------------------------------------------------------------------------


    STRUCTURE NAME = IGWLOCK00         TYPE = LOCK   STATUS = ACTIVE PRIMARY
              # REQ    -------------- REQUESTS -------------    -------------- DELAYED REQUESTS -------------
    SYSTEM    TOTAL             #     % OF  -SERV TIME(MIC)-    REASON    #    % OF  ---- AVG TIME(MIC) -----   EXTERNAL REQUEST
    NAME      AVG/SEC          REQ    ALL     AVG   STD_DEV             REQ    REQ   /DEL    STD_DEV   /ALL   CONTENTIONS

    SC63        259K   SYNC    259K   33.4    50.4    123.1     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      249K
              215.8    ASYNC   252     0.0   152.7    144.6     PR WT   256K  99.0    3.8        5.3    3.8   REQ DEFERRED    44K
                       CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP  150K  57.9    6.5       22.3    3.8   -CONT           44K
                                                                                                              -FALSE CONT   1439

    SC64        257K   SYNC    257K   33.1    51.0    126.5     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      250K
              214.1    ASYNC   249     0.0   446.6    525.8     PR WT   255K  99.1    3.7        4.7    3.7   REQ DEFERRED    46K
                       CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP  149K  57.9    6.7       23.7    3.9   -CONT           46K
                                                                                                              -FALSE CONT   1869

    SC65        259K   SYNC    259K   33.4    49.6    115.9     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      249K
              215.8    ASYNC   251     0.0   236.1    154.0     PR WT   256K  99.0    3.9       29.7    3.9   REQ DEFERRED    45K
                       CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP  148K  57.2    6.5       19.3    3.7   -CONT           45K
                                                                                                              -FALSE CONT   1594

    ------------------------------------------------------------------------------------------------------------------------------
    TOTAL       775K   SYNC    774K   100     50.3    121.9     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      749K
              645.8    ASYNC   752     0.1   277.8    348.5     PR WT   768K  99.0    3.8       17.6    3.8   REQ DEFERRED   136K
                       CHNGD     0     0.0                      PR CMP  447K  57.7    6.6       21.9    3.8   -CONT          135K
                                                                                                              -FALSE CONT   4902
```

Figure 5-22 CF Structure Activity report (IGWLOCK00)

Lock Structures for External Requests Contentions

RMF reports in REQ TOTAL, the number of requests issued by SMSVSAM to the XES API for the IGWLOCK00 structure, and in REQ DEFERRED the number of requests deferred due to contention. REQ TOTAL must be numerically close to # REQ in the same report, the difference is explained by the way XES account requests for unlocking.

A request can be deferred due to:

- True lock contention, measured by (CONT - FALSE CONT)
- False lock contention measured by FALSE CONT
- XES internal processing delays - if REQ DEFERRED is equal to CONT, there is no such type of delay.

As a result of lock contention you may see increased XCF activity caused by SMSVSAMs negotiating the lock status.

IGWLOCK00 shows 18% (135/749) for true lock contention. This is too much; the rule is 2 percent. True contention is application dependent; therefore, it is necessary to identify the workload using the lock structure. For example, long-running batch jobs that do not commit the locks can cause high true contention on lock structures.

The false lock contention is a little bit more than 1% (4/268) of the deferred requests, but the recommended value for IGWLOCK00 is 0.1 percent. This true contention should be further investigated. Refer to “True and false contention in lock structures” on page 289.

Cache Structures for Data Access

For this analysis, in Figure 5-23 on page 317, review the data about the RLS_CACHE structure.

By the way, the service time is approximately 20 microseconds for this structure, which indicates a fast CF hardware.

DATA ACCESS statistics for cache structures are acquired from counters in the CF; they cannot be broken down into individual systems. Only SMSVSAM knows how efficiently the local buffer and cache structure are being used. Depending on the cache structure implementation, one or more counts can be zero.

- READS and WRITES indicate how often the CFCC returned data to SMSVSAM (read) and how often SMSVSAM placed changed data into the RLS_CACHE structure. If you observe a high number of WRITES and a small number of READS (as our case), it means that the efficiency of the cache is low and something needs to be done. Read “SMSVSAM buffering and caching” on page 285.
- CASTOUTS is a count of the number of times an explorer (demanded by CFCC) retrieves a changed data entry, writes the data to DASD causing the changed attribute to be reset to unchanged in the structure. This happens when the structure occupancy reaches a CFCC internal threshold. As SMSVSAM uses a store-through algorithm no DASD castouts are needed to make room in the structure.
- XIs shows the number of times a data item residing in a local buffer pool was marked invalid by the CF during the RMF interval. The counter reflects the amount of data sharing among the users of the cache structure and the amount of write/update activity against the databases. Refer to “Control Interval cross invalidation” on page 287.

```
STRUCTURE NAME = RLS_CACHE         TYPE = CACHE  STATUS = ACTIVE
               # REQ    -------------- REQUESTS -------------   -------------- DELAYED REQUESTS -------------
     SYSTEM    TOTAL             #     % OF  -SERV TIME(MIC)-    REASON    #    % OF  ---- AVG TIME(MIC) -----
     NAME      AVG/SEC          REQ    ALL     AVG   STD_DEV             REQ    REQ   /DEL    STD_DEV   /ALL

     SC63        527K   SYNC    527K   35.1    21.7    135.5     NO SCH    0    0.0   0.0        0.0    0.0
               439.0    ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0   0.0        0.0    0.0
                        CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0   0.0        0.0    0.0
                                                                 DUMP      0    0.0   0.0        0.0    0.0

     SC64        535K   SYNC    535K   35.7    21.6    123.0     NO SCH    0    0.0   0.0        0.0    0.0
               445.7    ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0   0.0        0.0    0.0
                        CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0   0.0        0.0    0.0
                                                                 DUMP      0    0.0   0.0        0.0    0.0
     SC65        437K   SYNC    437K   29.2    21.4    102.7     NO SCH    0    0.0   0.0        0.0    0.0
               364.4    ASYNC   250     0.0   411.0    342.9     PR WT     0    0.0   0.0        0.0    0.0
                        CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0   0.0        0.0    0.0
                                                                 DUMP      0    0.0   0.0        0.0    0.0

    -------------------------------------------------------------------------------------------------------------------------------
     TOTAL      1499K   SYNC   1499K    100    21.5    122.2     NO SCH    0    0.0   0.0        0.0    0.0    -- DATA ACCESS ---
                1249    ASYNC   250     0.0   411.0    342.9     PR WT     0    0.0   0.0        0.0    0.0    READS       79246
                        CHNGD     0     0.0                      PR CMP    0    0.0   0.0        0.0    0.0    WRITES     572353
                                                                 DUMP      0    0.0   0.0        0.0    0.0    CASTOUTS        0
1
```

Figure 5-23   CF Structure Activity report (RLS_CACHE)

Subchannel Activity report

This report is presented in Figure 5-24. Before we start, here is a recapitulization of CF hardware.

A CF request is first processed by XES, then transferred to the CF, if there is a free link, processed by a CF CP, and sent back to OS/390 via the CF link. This report discusses the possible delays for those requests.

Each CF link in peer mode has seven subchannels, and seven buffers, per each image that it is serving (MIF). Subchannels are busy from the time MVS sends the request until the time MVS processes the response. Links are only busy for the time it takes for the data to travel between CPCs.

CF Links do not support CPMF, so there is no reporting anywhere of actual CF link utilization. Actual link utilization is rarely an issue.

The count used to build this report are not available on a per structure basis. Some of the fields have the same meaning as the ones found in the report shown in Figure 5-23. Here, we only mention the fields with a more complete comprehension:

- PTH BUSY: Measures the number of times that a synchronous request gets a free subchannel but encounters all path busy condition. This situation can occur because the CF link is shared (MIF) between several z/OS images. The request “spins” until the link becomes available. If this figure is far from zero, you should dedicate the CF link to the LP. Our number is zero.

- DELAYED REQUESTS: These columns have the same meaning of the CF Structure report, but organized by the type of the structures.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                    PAGE  11
        z/OS V1R4              SYSPLEX SANDBOX            START 03/15/2003-15.10.00           INTERVAL 000.20.00
                               RPT VERSION V1R2 RMF       END   03/15/2003-15.30.00           CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF02
   ------------------------------------------------------------------------------------------------------------------------------
                                                     SUBCHANNEL  ACTIVITY
   ------------------------------------------------------------------------------------------------------------------------------
             # REQ                                 ----------- REQUESTS -----------   ------------------ DELAYED REQUESTS --------------
    SYSTEM   TOTAL   -- CF LINKS --  PTH                    #    -SERVICE TIME(MIC)-               #     % OF  ------ AVG TIME(MIC) ------
    NAME     AVG/SEC TYPE  GEN  USE  BUSY                 REQ     AVG    STD_DEV                 REQ    REQ   /DEL    STD_DEV     /ALL

    SC63     814343  ICP     2    2     0   SYNC    783206    32.9      132.6  LIST/CACHE    0  0.0    0.0        0.0      0.0
              678.6  SUBCH  28   14         ASYNC    22124   306.4      415.3  LOCK          0  0.0    0.0        0.0      0.0
                                            CHANGED      0  INCLUDED IN ASYNC  TOTAL         0  0.0
                                            UNSUCC       0     0.0        0.0
    SC64     821174  ICP     2    2     0   SYNC    789697    32.9      125.6  LIST/CACHE    0  0.0    0.0        0.0      0.0
              684.3  SUBCH  28   14         ASYNC    21817   232.8      435.0  LOCK          0  0.0    0.0        0.0      0.0
                                            CHANGED      0  INCLUDED IN ASYNC  TOTAL         0  0.0
                                            UNSUCC       0     0.0        0.0
    SC65     741811  ICP     2    2     0   SYNC    693791    33.8      109.2  LIST/CACHE    0  0.0    0.0        0.0      0.0
              618.2  SUBCH  28   14         ASYNC    37460   161.3      313.0  LOCK          0  0.0    0.0        0.0      0.0
                                            CHANGED      0  INCLUDED IN ASYNC  TOTAL         0  0.0
                                            UNSUCC       0     0.0        0.0
```

Figure 5-24   Subchannel Activity report

CF to CF Activity report

This report is presented in Figure 5-25.

This report is introduced because of SM duplexing. In such type of duplexing there is a conversation between the two CFs. The fields pictured in the report show details of the conversation. All of them are already covered in this topic in other reports.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                         PAGE  12
        z/OS V1R4              SYSPLEX SANDBOX            START 03/15/2003-15.10.00           INTERVAL 000.20.00
                               RPT VERSION V1R2 RMF       END   03/15/2003-15.30.00           CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF02
   ------------------------------------------------------------------------------------------------------------------------------
                                                       CF TO CF ACTIVITY
   ------------------------------------------------------------------------------------------------------------------------------
              # REQ                                ----------- REQUESTS -----------   -------------- DELAYED REQUESTS --------------
    PEER      TOTAL      -- CF LINKS --                    #   -SERVICE TIME(MIC)-          #     % OF  ------ AVG TIME(MIC) ------
    CF        AVG/SEC    TYPE       USE                   REQ     AVG    STD_DEV           REQ    REQ   /DEL    STD_DEV     /ALL

    CF01        1536K    ICP          2        SYNC    1536K    2.1        1.4    SYNC     0    0.0    0.0        0.0      0.0
             1279.8

    CF02            0    ICP          2        SYNC        0    0.0        0.0    SYNC     0    0.0    0.0        0.0      0.0
                  0.0
```

Figure 5-25   CF to CF report

XCF Activity Report (XCF Usage by Member)

This report shows the XCF activity caused by SMSVSAM. The majority of this activity is caused by the conversation among SMSVSAMs to verify if a lock contention is true or false. The name of the group used by SYSVSAM is IXCLOxxx. We have the impression that the number in the reports do not need further explanation. For example, the SMSVSAM running in SC63 sends 11890 messages to SMSVSAM running in SC64 (both in IXCLO001 group). See Figure 5-26.

```
 X C F  A C T I V I T Y
                                                                                                                            PAGE    3
             z/OS V1R4                SYSTEM ID SC63             START 03/17/2003-18.30.00  INTERVAL 000.10.00
                                      RPT VERSION V1R2 RMF       END   03/17/2003-18.40.00  CYCLE 1.000 SECONDS
-
                                                        XCF USAGE BY MEMBER
 -------------------------------------------------------------------------------------------------------------------------------------
               MEMBERS COMMUNICATING WITH SC63                                                   MEMBERS ON SC63
 ------------------------------------------------------------------           --------------------------------------------------------
                                                 REQ          REQ
                                                 FROM           TO                                                   REQ         REQ
 GROUP       MEMBER              SYSTEM          SC63         SC63            GROUP       MEMBER                     OUT          IN
IXCL0001    M166                SC64          11,890       11,960            IXCL0001    M167                    31,872      31,897
             M168                SC65          11,493       11,448                                            ----------   ----------
                                           ----------   ----------            TOTAL                               31,872      31,897
 TOTAL                                         23,383       23,408
 IXCL0004    M759                SC64              80           63            IXCL0004    M760                       185         195
             M761                SC65              46           73                                            ----------   ----------
                                           ----------   ----------            TOTAL                                  185         195
 TOTAL                                            126          136
 IXCL0005    M193                SC64               0            0            IXCL0005    M194                         0            0
             M195                SC65               0            0                                            ----------   ----------
                                           ----------   ----------            TOTAL                                    0

0
 TOTAL                                              0            0
 IXCL0026    M132                SC64               0            0            IXCL0026    M133                         0            0
                                           ----------   ----------                                            ----------   ----------
 TOTAL                                              0            0            TOTAL                                    0
```

Figure 5-26   XCF Activity report

5.7.7 MVS commands about RLS performance

There are several commands covering RLS performance aspects. The most important one is the following:

D SMS,CFLS

This command shows details about lock contention in RLS locks. Refer to its output in Figure 5-27.

There is one line per each time interval where the data is averaged.

- *LockRate*: The number of locks required per second (shared and exclusive)
- *ContRate*: % of lock requests globally managed in the CF, if you are running your RLS sphere in just one system, this number is zero.
- *FContRate:* % of lock requests falsely globally managed. It indicates the percentage of false contention in the IGWLOCK00 structure
- *WaitQLen:* The average of the total number of waiters for any lock. We should expect this field to increase when LockRate increases passing from a larger period of observation to a small period (1 hour to 1 minute, for example). If WaitQueue increases just by itself, it indicates a performance problem.

```
System   Interval   LockRate   ContRate  FContRate   WaitQLen
MZ1A     1 Minute     1569.1      0.000      0.000     246.30
MZ1A      1 Hour      1358.8      0.000      0.000      30.38
MZ1A      8 Hour       887.0      0.000      0.000       7.81
MZ1A       1 Day       717.3      0.000      0.000       8.47
(03)   1 Minute      523.0      0.000      0.000      82.10
(03)    1 Hour       452.9      0.000      0.000      10.13
(03)    8 Hour       295.7      0.000      0.000       2.60
(03)     1 Day       239.1      0.000      0.000       2.82
**************** LEGEND *****************
LockRate = number of lock requests per second
CONTRATE = % of lock requests globally managed
00    FCONTRATE = % of lock requests falsely globally managed
WaitQLen = Average number of requests waiting for locks
```

Figure 5-27 D SMS, CFLS command output

5.7.8 SMF records covering VSAM RLS

There are two SMF records covering VSAM RLS statistical performance data: SMF record type 42 and 64. Here we just describe the fields associated with VSAM RLS. Refer to "SMF record types related to VSAM data sets" on page 194 for additional information. Through TDS (grandson of SLR) product you can create graphical reports relating the SMF reported information. Refer to "Tivoli Decision Support (TDS)" on page 137.

SMF record 42

This record can be created on a timed interval or whenever the SMF timer ends. You can specify SMF_TIME in IGDSMSxx to synchronize SMF type 42 data with SMF and RMF data intervals. SMF record type 42 has many subtypes, some of them have RLS cache structure statistics. This data includes information for each system and a sysplex-wide summary:

- Subtype 15 collects data about storage class response time.
- Subtype 16 collects data about data set response time.
- Subtype 17 collects data about coupling facility lock structure usage.
- Subtype 18 collects data about coupling facility cache partition usage.

Because data is collected across the sysplex, it is unnecessary to merge SMF records from all the systems in the sysplex.

SMF record 64

SMF writes a record type 64 when a cluster is closed or processed by EOV. SMF writes one record for each component in the cluster. SMF record type 64 has no subtypes. New sections were added to give CF cache information and some fields were modified to reflect RLS use of the cluster or component. Field SMF64RLS tells you if the VSAM cluster or component was opened for RLS. Refer to Appendix A, “Sample code” on page 363, where you will find a source code to read the SMF 64 selecting the RLS information only. The following report is produced by such code.

```
1         VSAM RLS - CF STRUCTURE STATISTICS SINCE LAST OPEN

  SYSID JOBNAME  START DT/TIME  DDNAME   DATA SET NAME

  SC63  MHLSC63  03071 12:03:11 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       51988 CF CACHE HIT:      20831 I/O DASD:           0
  INS:       32799 DEL:          0 UPDT:      32694 READ:      32781 CI-SPLT:   233 CA-SPLT:     0

  SC64  MHLSC64  03071 11:29:38 VSAMRLSU MHLRES2.VSAM.RLSEXT
  LOCAL BP HIT:         368 CF CACHE HIT:         14 I/O DASD:           0
  INS:          14 DEL:          0 UPDT:          0 READ:          0 CI-SPLT:     0 CA-SPLT:     0

  SC64  MHLSC64  03071 12:03:11 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       46933 CF CACHE HIT:      20162 I/O DASD:           0
  INS:       32427 DEL:          0 UPDT:      22640 READ:      22742 CI-SPLT:   230 CA-SPLT:     0

  SC65  MHLSC65  03071 12:03:11 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       66836 CF CACHE HIT:      20998 I/O DASD:           0
  INS:       33348 DEL:          0 UPDT:      33246 READ:      33384 CI-SPLT:   235 CA-SPLT:     1

  SC63  MHLSC63  03071 13:03:19 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       52135 CF CACHE HIT:      20529 I/O DASD:           0
  INS:       32436 DEL:          0 UPDT:      32341 READ:      32429 CI-SPLT:   229 CA-SPLT:     0

  SC63  MHLSC63  03071 13:33:15 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       52172 CF CACHE HIT:      20462 I/O DASD:           0
  INS:       32419 DEL:          0 UPDT:      32341 READ:      32411 CI-SPLT:   228 CA-SPLT:     0

  SC64  MHLSC64  03071 13:03:19 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       51012 CF CACHE HIT:      20576 I/O DASD:           0
  INS:       32783 DEL:          0 UPDT:      22906 READ:      23004 CI-SPLT:   233 CA-SPLT:     0

  SC64  MHLSC64  03071 13:33:15 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       49596 CF CACHE HIT:      20474 I/O DASD:           0
  INS:       32679 DEL:          0 UPDT:      22826 READ:      22922 CI-SPLT:   231 CA-SPLT:     0

  SC65  MHLSC65  03071 13:33:15 VSAMRLSU MHLRES2.VSAM.RLSEXT.DATA
  LOCAL BP HIT:       61560 CF CACHE HIT:      21085 I/O DASD:           0
  INS:       33270 DEL:          0 UPDT:      33199 READ:      33337 CI-SPLT:   237 CA-SPLT:     0
```

Figure 5-28 Output from SMF64

For more detailed information about SMF records, refer to *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

6

DFSMStvs

In this chapter we cover DFSMStvs. Note that there is an IBM Redbook available titled, *DFSMStvs: Overview and Planning Guide*, SG24-5741. We do not intend to duplicate the material covered in that publication but rather, document our hands-on experiences with TVS.

The topics cover:

- TVS introduction
- Implementing TVS
- Exploiters of TVS
- Changes to system commands, JCL, parmilb member IGDSMSxx
- TVS problem determination and recovery

6.1 Introducing DFSMStvs

In this chapter we look briefly at the recent enhancements to VSAM and show how DFSMStvs is a natural extension to the functions provided by VSAM Record Level Sharing.

The key functions in DFSMStvs are that VSAM will provide locking, two-phase commit, backout, and logging facilities that allow multiple batch update jobs to run concurrently with CICS access to the same data sets while maintaining integrity and recoverability.

We also introduce some of the terms that are used throughout this book.

6.2 Why DFSMStvs?

In the past, it was often acceptable for a CICS system to be available during normal daytime business hours, perhaps for a total of twelve hours. This left plenty of time for the CICS system or application to be shut down and for supporting batch work to be run. There was no requirement to make business data available on the Internet.

CICS typically is active and then shut down in preparation for running batch work for some period of time each day. As soon as CICS has stopped, backups are taken of key data sets as a point of recovery. Then, batch jobs can be scheduled to run. If we have several jobs that update the same data set, they will run sequentially, because they could not share the data set for update. After the batch updates are complete, there may be a job to read the updated data and produce reports. Probably, another backup will be done at this stage. When this is complete, CICS can be restarted and will be active again. DFSMStvs allows us to extend the CICS window of availability.

6.2.1 How to extend CICS availability

There is increasing pressure to extend the availability of CICS systems. This is because of the need for better customer service which in turn requires longer service hours, or because 24 hours per day Internet access to core business applications is desirable to improve competitiveness, extend the range of customers served, or reduce costs.

You will probably start to use DFSMStvs so that you can meet new or changing business objectives. These objectives might include:

- Extending service hours for automated teller machines
- Providing kiosk access to your applications for your customers

- Linking your applications with your suppliers or customers
- Web-enabling applications for direct customer use
- Allowing batch reruns to be done during the online day

In each case, there will be new pressures on availability of your systems. One consequence of the need for greater availability will be that you cannot afford the CICS down-time caused by your existing batch window. The actions that you will need to take will depend on whether you can still tolerate a batch window (although it will be a shorter window) or whether you must aim for continuous availability: 24 hours a day, 7 days a week.

6.2.2 Reducing the batch window

There are several techniques to reduce the batch window, but each has drawbacks.

You can use the External CICS Interface (EXCI), which allows a non-CICS application program to call a CICS program. You would change application programs to use EXCI to pass data to a CICS server transaction, which would then perform the updates with all the facilities available to CICS. However, this needs major application changes, and requires that you provide a server transaction, but it does completely remove work from a batch window.

You can remove a data set from CICS file control while CICS is still running. You use the CEMT transaction to close the data set and re-open it as read-only. You could then copy the file and make batch updates to the copy while still providing read access to the original. After the updates are completed, you deallocate the old version and allocate the new, updated version to CICS. This does not remove work from the batch window, but does offer limited (read-only) access while updates are being done.

You can force VSAM to allow sharing by using SHAREOPTIONS (3) or SHAREOPTIONS(4) but all integrity and recovery issues concerning locking and logging become an application responsibility. The application changes needed to give you complete integrity and recoverability are probably too large to contemplate. For example, you would need to provide logging, locking, and commit, backout protocols such that a CICS transaction backout would not remove updates that were performed correctly by a batch job.

Finally, you can change transactions to collect updates while a data set is being updated in batch and apply the changes later. A memo file is used to collect new records, and transactions are changed to switch between the main file and the memo file when the main file is not available for updates or additions.

Inquiries must also check the memo file first. When the main file update or reorganization is complete, the contents of the memo file are copied into the main file and the switch is reset so that all accesses are now performed solely against the main file. This may have integrity problems depending on the functions that you provide. Adding new records may be safe, but updating existing records opens up the possibility of updating a record that has been updated elsewhere.

These methods provide some relief at the expense of added complication or reduced integrity and recoverability. It is clear that a more general solution to the problems of running batch updates for CICS VSAM data is needed.

6.3 Some definitions

If you are interested in DFSMStvs as a storage administrator, you may not be familiar with some of the terms and concepts that will be used throughout this book. This section offers some brief definitions.

6.3.1 Backward recovery

If a transaction fails, it may leave data in an inconsistent state. Perhaps one update has been requested, but the program failed before requesting a related update.

The process of removing changes which are possibly inconsistent is variously called *backward recovery*, *rollback*, or *backout*. It requires the use of a log that holds images of the records before they were updated. This log is called an *undo log*. In CICS terminology, the records are known as *before* images.

6.3.2 Forward recovery

If a data set is lost, a common way of getting the data back is to recover from a backup copy and then to apply all the changes that were made after the backup copy was taken.

This process is called *forward recovery*. It requires a log of the changes made to a data set together with a date and time stamp. This log is called a *redo* or *forward recovery* log.

6.3.3 Atomic updates

By *atomic updates*, we mean an indivisible group of updates so that either all updates are made or none are made; it would not be possible for some updates to be made but for others to fail and still maintain data integrity. This property is an absolute requirement for integrity. Let us illustrate the need for atomic updates by the simple example of a transfer of money between two bank accounts, as shown in Figure 6-1.

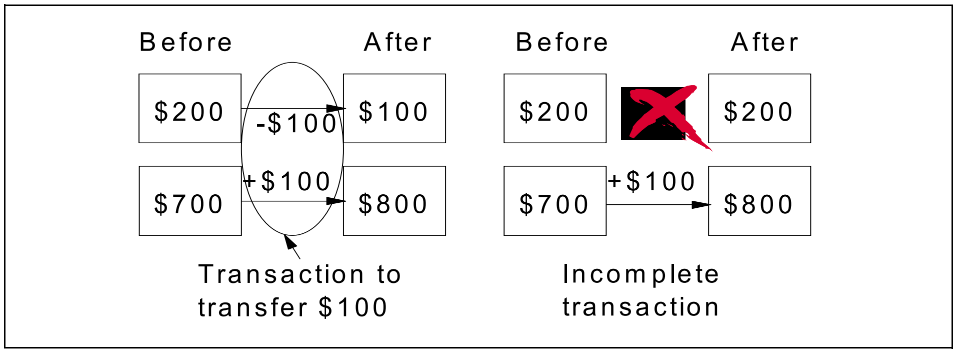


Figure 6-1 An atomic update example

Clearly, either *both* changes must be made, or *neither* can be made. If only the addition or credit is made, the bank has given the customer money; if only the subtraction or debit is made, the bank has taken money from the customer.

Making final changes to the data is called *committing*. Only the application itself knows when data should be committed and it will request that the data be committed at an appropriate point in the processing flow. For example, the application described in Figure 6-1 would request a commit when the updates for both accounts had been done. It would not make sense to commit if only one update had been done, as an integrity exposure would be created.

6.3.4 Unit of work and unit of recovery

A *unit of work* (UOW) is the term used in CICS publications for a set of updates which are treated as an atomic set of changes. The z/OS Resource Recovery Services (RRS) use *unit of recovery* to mean much the same thing. So, a unit of recovery is the set of updates between synchronization points. There are implicit synchronization points at the start of a transaction. There should also be explicit synchronization points requested by an application within a transaction or batch job. It is preferable to use explicit synchronization for greater control of the number of updates in a unit of recovery.

Changes to data are durable after a synchronization point. That means that the changes survive any subsequent failure.

Figure 6-2 shows the units of recovery (noted as A, B, and C) in a job or transaction. Notice that the points of synchronization are shown, whether explicitly requested by a commit or implicitly at the start and end of the transaction.

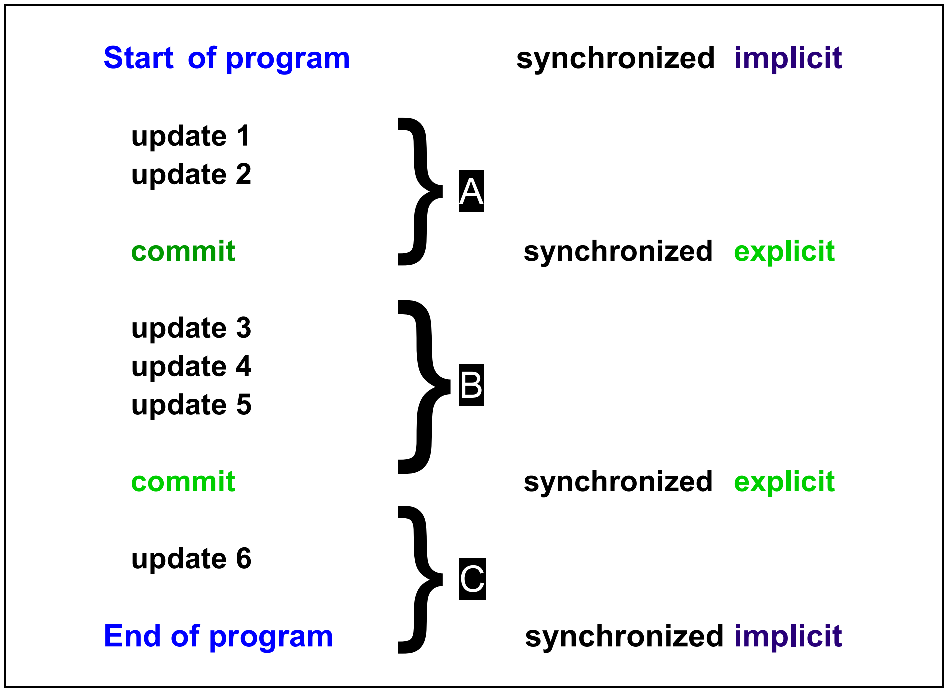


Figure 6-2 Unit of recovery examples

Figure 6-2 demonstrates what could happen, but this is not actually what we would recommend. For reasons described later, we recommend that there should be an explicit commit done before a data set is closed.

Unfortunately, the z/OS RRS and DFSMStvs documentation draw a distinction between unit of work and unit of recovery. In this case, unit of work has its MVS definition (a task running under its own TCB) and can refer to more than one unit of recovery. In Figure 6-2, the sequence of three units of recovery between the start and end of the transaction is a unit of work.

We apologize to the reader for this conflict in terms and, for consistency, will use the z/OS definition of a unit of recovery in this book.

6.3.5 Two-phase commit

Two-phase commit is a technique that is widely used by database management systems to provide an atomic update capability. It is necessary when there are multiple resource managers involved and we have what is known as a distributed unit of work.

Two-phase commit requires that there be a coordinator who is called at synchronization points. In the first phase, the coordinator asks the individual resource managers (such as DFSMStvs) to prepare to commit the changes that they have made. The resource managers must reply to the coordinator to say that they are ready to commit. When they do so, the coordinator will, as the second phase, tell the updaters to make the changes permanent.

In the case of DFSMStvs, z/OS RRS plays the coordinating role. Commit requests are generated implicitly at the successful end of a job step and should also be done explicitly by inserting commit calls in the program logic.

6.3.6 In-flight and in-doubt

A unit of recovery is described as being *in-flight* if it has made changes to a record in a recoverable data set, but has not yet committed or backed out the changes.

A unit of recovery is described as being *in-doubt* in the brief period of time between DFSMStvs agreeing to an RRS request for a commit, and the signal from RRS that all the commit participants have agreed and that the commit should be done.

6.3.7 Repeatable read

With batch programs using RLS, you have a choice of two ways of handling read integrity. The default is *no read integrity* (NRI) where record locks are not obtained for a read. This means that you may read a record and get uncommitted data that may be backed out. This is sometimes called a *dirty read*. The other option is to ask for a *consistent read*. In this case, RLS ensures that the read request waits until updates for that record have been committed.

DFSMStvs adds a third option, *repeatable read* (also known as *consistent read explicit*). This provides a way of ensuring that a record cannot be changed by someone else until the requester of repeatable read either commits or backs out, thereby releasing the lock preventing others from updating the record. It also gives your program isolation from other changes; it cannot see updates made by other programs which have not been committed and so it is a form of consistent read.

The difference between a consistent read and a repeatable read is that a repeatable read keeps a lock on the record read until the unit of recovery requesting the read commits or backs out. Hence, a consistent read gives us a consistent view of data where all associated updates have been committed, but does not maintain that view; a repeatable read allows the requester to maintain that consistent view of the data so that future reads will get the same data until the requester itself decides to commit.

6.3.8 Recoverable data sets

RLS introduced the concept of recoverable and non-recoverable data sets to VSAM itself. Previously, CICS file control supported recoverable VSAM data sets but only for access from CICS transactions.

A *recoverable data set* must be accessed by a system that can perform two-phase commit and backout of failed updates. The VSAM cluster must be defined with either the LOG(UNDO) or LOG(ALL) attributes to be recoverable.

If LOG(UNDO) is specified, backout can be done but not forward recovery. This means that the image of a record before it was changed is logged so that the log entry can be used to undo a change in the event of failure.

If LOG(ALL) is specified, both backout and forward recovery can be done. Forward recovery requires that the image of a record after a change be stored so that the log entry can be used to redo a change.

A *non-recoverable data set* is defined with either the LOG(NONE) attribute or without the log attribute being specified. Neither a backout nor a forward recovery capability exists for a non-recoverable data set. As these capabilities do not exist, they cannot be compromised by batch updates, so batch updating is allowed for non-recoverable data sets.

6.4 CICS support for recoverable VSAM

You can define VSAM data sets to CICS as recoverable resources. This definition is in the CICS file resource definition or the ICF catalog. For RLS and DFSMStvs, the definition must be in the ICF catalog.

CICS can provide logging for both backward and forward recovery. Backward recovery supports transaction backout by writing a copy of a record before it was updated to a log, sometimes called an *undo* log. CICS performs backwards recovery itself. Forward recovery adds the ability to take a backup copy of a VSAM data set and use logged updates to reapply changes to that data set to bring it up to date. This is also referred to as a *redo* log. CICS does not do forward recovery; a program such as CICSVR is needed to use the redo logs to provide forward recovery.

If a transaction ends in error, CICS will back out the changes made by that transaction. CICS provides a commit mechanism used when a logical set of updates has completed. When changes are committed they cannot be backed out.

You have long been able to share VSAM data between CICS systems by using function shipping between Application Owning Regions (AORs) and a File Owning Region (FOR), as shown in Figure 6-3.

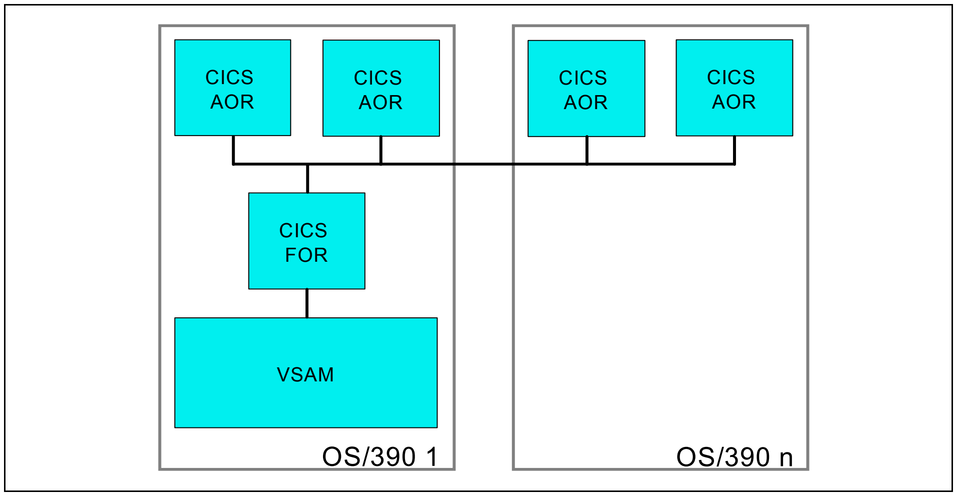


Figure 6-3 Sharing VSAM data through a CICS file owning region

There are several drawbacks with this approach:

- The file owning region is a single point of failure.
- If the workload of the file owning region increases, it may become a performance bottleneck. You do not have the ability to use Parallel Sysplex horizontal growth to provide a growth path.
- Transactions are shipped to the system with the file owning region from other systems using XCF links or VTAM. These can impose performance costs.

6.5 DFSMStvs overview

This section provides an overview of DFSMStvs including its relationship with RLS.

6.5.1 The RLS connection

As we saw in the previous chapter on RLS, which was introduced in DFSMS/MVS 1.3, RLS provides for a new mode of access to VSAM data. Also, we learned that RLS uses the System/390® Coupling Facility to provide the lock structures and data caching abilities to manage and ensure data integrity when VSAM data is shared. This is the reason that a Coupling Facility is required for RLS even if you want to share VSAM data in a monoplex. DFSMStvs extends the sharing of data to CICS and batch level sharing similarly maintaining data integrity for reads and updates. To accomplish this, DFSMStvs must provide the same abilities that CICS provides, such as logging for forward and backward recovery, backout, and a two-phase commit process.

6.5.2 DFSMStvs locking

When a VSAM application issues a GET or a PUT (or the high-level language equivalent of these VSAM macros), VSAM RLS will acquire locks on behalf of the application. A coupling facility is used to hold the locks so that they can be shared by all the SMSVSAM address spaces in a sysplex.

In general, VSAM RLS will obtain share locks for read requests and exclusive locks for update or delete requests. It will wait for a certain period of time if another task holds the required locks. A share lock may be obtained for a resource if there is no lock for that resource or there are share locks held. An exclusive lock may only be obtained if no other locks are held. Note that CICS publications refer to share locks as shared locks.

The DFSMStvs access mode uses VSAM RLS locking and so it is compatible with CICS locking.

6.5.3 DFSMStvs logging

The z/OS system logger is used to provide the logging functions that are needed to ensure data consistency after failure. The redo log name is held as an attribute of a recoverable data set while the undo log name is fixed. Forward recovery logging will be done by both CICS and DFSMStvs to the same redo log. CICS will log changes made by CICS transactions while DFSMStvs will log changes made by its callers as shown in Figure 6-4.

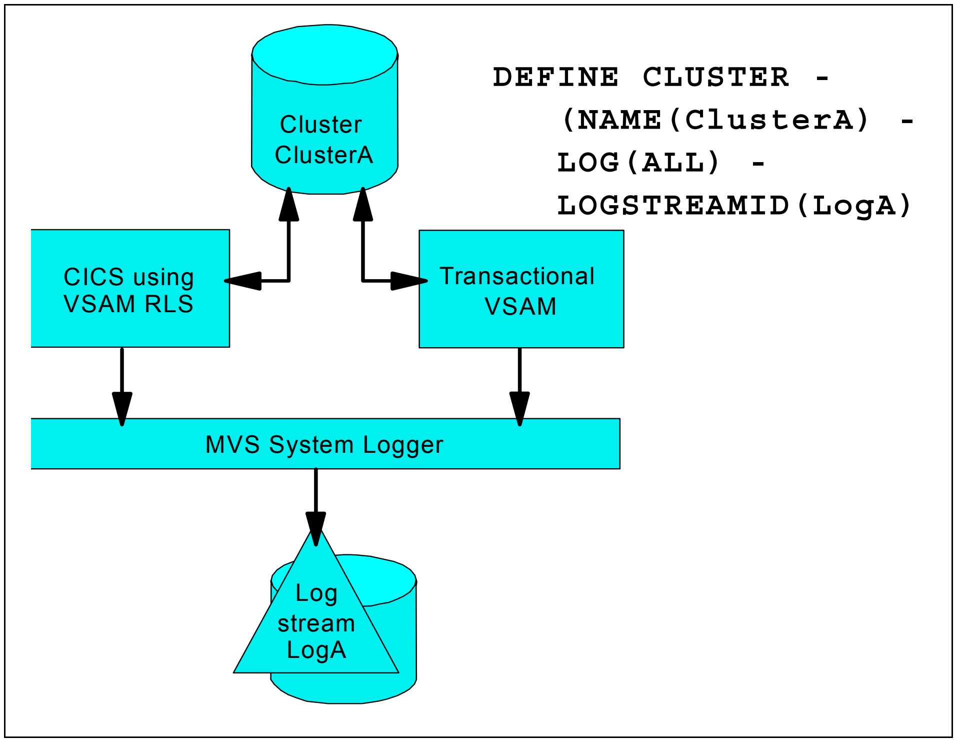


Figure 6-4 Merged forward recovery log

The system logger is used because it can merge log entries from many z/OS images to a single log stream, where a log stream is simply a set of log entries. A single log stream across a sysplex eases management and protects data integrity.

This ability to merge log entries is used for the forward recovery logs to ease recovery. The system logger writes log entries to the coupling facility, disk or both. However, note that each instance of DFSMStvs has a private undo log; the undo logs are not shared. An *instance* of DFSMStvs is the single version of DFSMStvs running in one z/OS image and defined by the IGDSMSxx PARMLIB member for that z/OS image.

DFSMStvs uses these log streams:

1. Primary, also called backout or undo log stream

   There is one backout log stream for each instance of DFSMStvs (SMSVSAM). It is not shared between SMSVSAMs. The name of the backout log stream is constructed from the unique name for an instance of DFSMStvs which you define in SYS1.PARMLIB(IGDSMSxx).

It is named *tvsname*.IGWLOG.SYSLOG, where *tvsname* is of the form IGWTV*nnn*.

2. Secondary, also called shunt log stream

   There is one shunt log stream for each instance of DFSMStvs. It is not shared between SMSVSAMs. The name of the shunt log stream is constructed using the unique name for an instance of DFSMStvs that you define in SYS1.PARMLIB(IGDSMSxx).

   It is named *tvsname*.IGWSHUNT.SHUNTLOG, where *tvsname* is of the form IGWTV*nnn*.

   The shunt log is used when backout requests fail and for long running units of recovery. In these cases log records are moved so that DFSMStvs can better manage space in the primary log.

3. Forward recovery log streams

   Forward recovery logs are used by data sets for which LOG(ALL) is specified. When LOG(ALL) is specified you must also specify LOGSTREAMID to provide the name of the forward recovery logstream. This logstream is used by all instances of DFSMStvs and CICS to provide forward recovery capability.

4. Log of logs

   This is an optional log stream, it is specified in SYS1.PARMLIB(IGDSMSxx). If it is defined, it contains copies of log records that are used to automate forward recovery.

   Although the log of logs is optional, we recommend that it be used if you want to use forward recovery at all. However, when you define a log of logs, it must be present otherwise DFSMStvs fails.

Figure 6-5 shows a simple example with undo and redo logging. The vertical arrow shows the commit point at which time the undo log records are effectively forgotten. Note that the undo log records show how the records existed before the transaction, while the redo log records show how the records exist after they are updated. The log records are prefixed with header information used for recovery or backout.

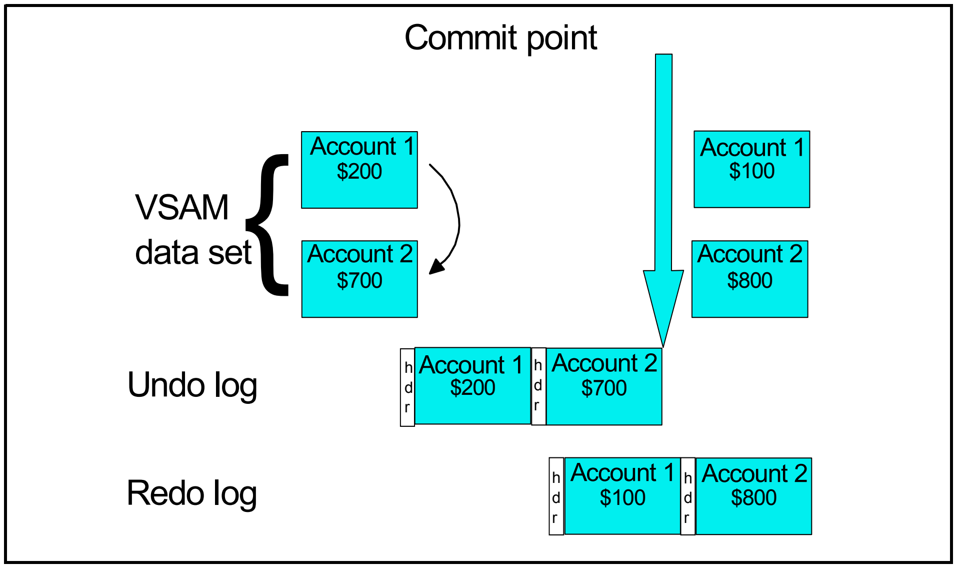


Figure 6-5 Undo and redo logging

6.5.4 Recovery coordination

In the case of DFSMStvs, the resources that are being protected are the records within recoverable VSAM data sets. There are three types of participants in resource recovery:

1. Application: The application program requests the use of resources, in this case records. It is responsible for requesting that changes be committed to make them permanent or that they be backed out to remove them and thereby return the records to their previous state.
2. Resource Manager: DFSMStvs uses the VSAM provided interfaces for two-phase commit and backout. These interfaces allow for:
   - Reading records
   - Updating records
   - Insertion of records
   - Erasure of records
3. Syncpoint manager: RRS is the syncpoint manager. It is responsible for providing a single point of coordination for the commit processing of resources owned by different resource managers. It provides the interfaces that an application uses to:
   - Commit changes
   - Back out changes

z/OS Recoverable Resource Services are used to coordinate commit and backout requests from applications and any other resource managers that may be needed. Apart from DFSMStvs, other resource managers using z/OS RRS are DB2 UDB, IMS and MQSeries. This means that an application can use a combination of VSAM, IMS, DB2 UDB and MQSeries services and have commit and backout performed atomically for all these resources.

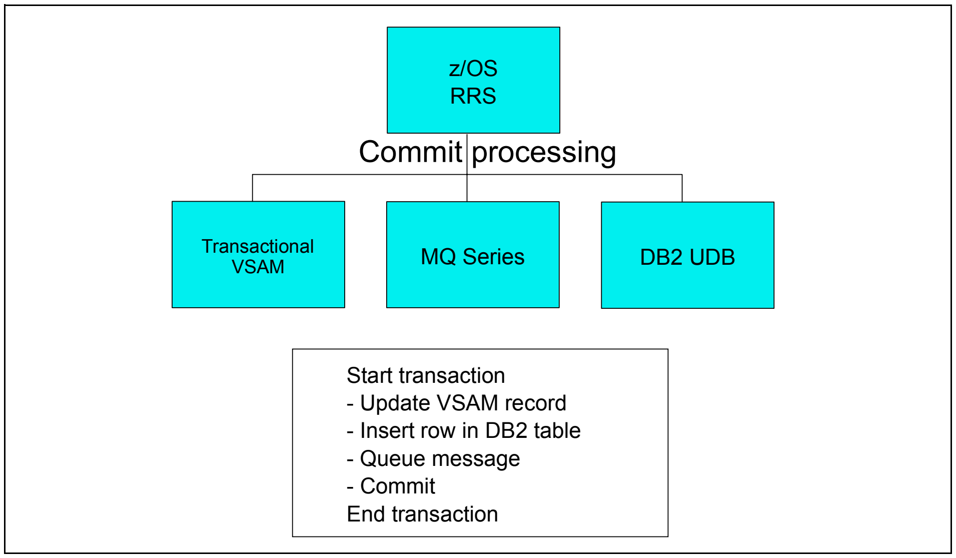


Figure 6-6 RRS as the sync point manager

The DFSMStvs code calls RRS to register its interest in a unit of recovery. A unit of recovery represents the set of changes made by an application to a resource since the last commit (implicit or explicit) or backout. Each unit of recovery is associated with a context.

DFSMStvs writes a copy of an unmodified record (the record as it existed before an update) to the undo log and, when the data set's log attribute is set to ALL, writes a copy of the modified record (as it exists after an update) to the forward recovery log.

When an application requests either a commit or a backout, z/OS RRS will call DFSMStvs to do the commit or backout on behalf of the application. The processing flow is shown here in a simplified form for a successful commit:

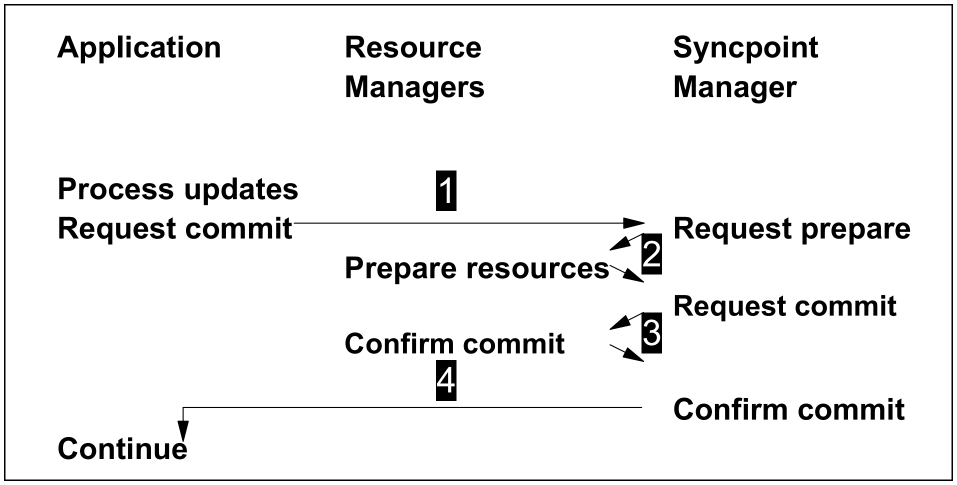


Figure 6-7 Commit processing participants

In Step 1, the application decides that it is ready to commit changes. It requests a commit and that request is passed to the syncpoint manager. The syncpoint manager sees what resources are involved and, in Step 2, asks each resource manager that has expressed interest in this unit of recovery to prepare to commit and notify it of readiness. If all the resource managers reply that they are ready to commit, the syncpoint manager tells them to complete the process in Step 3. When that is done, the syncpoint manager can then confirm that the commit has completed to the application in Step 4.

6.6 Our experiences with implementation

You must have VSAM RLS already set up before you can implement TVS. For details about setting up VSAM RLS, refer to “Implementing VSAM RLS” on page 253.

Once RLS is set up, you would already have defined the necessary lock and cache structures in the coupling facility. Now you need to go through the following steps to implement TVS.

- Define list structures in the coupling facility
- Define the log streams in the coupling facility
- Define DASD staging data sets
- Set up the security definitions
- Define the SMS classes
- Modify SYS1..PARMLIB(IGDSMSxx)

The rest of this section describes each of these steps.

6.6.1 Define list structures in the CFRM policy

You have to define the list structures in the coupling facility resource management (CFRM) policy. The list structures contain the various log streams used by TVS. For details, see “Define the log structures and log streams in LOGR policy” on page 340.

Figure 6-8 provides sample JCL to define the list structures. Four list structures are defined by this job.

```
//LstStruc JOB (999,POK),'CFRM',CLASS=A,REGION=4096K,
//              MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM21) REPLACE(YES)

    CF NAME(CF01) DUMPSPACE(2048) PARTITION(E) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)

    CF NAME(CF02) DUMPSPACE(2048) PARTITION(D) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)
STRUCTURE NAME(LOG_IGWLOG_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF02,CF01)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_IGWSHUNT_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF01,CF02)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_FORWARD_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF01,CF02)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_IGWLGLGS_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF02,CF01)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)
```

Figure 6-8   Define list structures in the CFRM policy for TVS

6.6.2 Define the log structures and log streams in LOGR policy

The definitions for log structures and log streams are done in the system LOGR policy. Their definitions refer to the list structures you defined in the previous step.

A sample JCL is provided in Figure 6-9. In this job we define the log structures first. Then in these log structures, we define the log streams for SYSLOG and SHUNTLOG for three z/OS systems in our sysplex. Finally, we define the log streams for FR.LOG and LOG.OF.LOGS.

```
//TVSLOG JOB (999,POK),'LOGR RRS',CLASS=A,REGION=4M,
//             MSGCLASS=T,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
     DATA TYPE(LOGR) REPORT(YES)

  DEFINE STRUCTURE NAME(LOG_IGWLOG_001) LOGSNUM(5)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_IGWSHUNT_001) LOGSNUM(5)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_FORWARD_001) LOGSNUM(10)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_IGWLGLGS_001) LOGSNUM(1)
          MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE LOGSTREAM
  NAME(IGWTV063.IGWLOG.SYSLOG)
  STRUCTNAME(LOG_IGWLOG_001)
  LS_DATACLAS(SHARE33)
  HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
  LOWOFFLOAD(15) HIGHOFFLOAD(95)
  STG_DUPLEX(YES) DUPLEXMODE(COND)
  STG_DATACLAS(SHARE33)
  DIAG(YES)

  DEFINE LOGSTREAM
  NAME(IGWTV064.IGWLOG.SYSLOG)
  STRUCTNAME(LOG_IGWLOG_001)
  LS_DATACLAS(SHARE33)
  HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
  LOWOFFLOAD(15) HIGHOFFLOAD(95)
  STG_DUPLEX(YES) DUPLEXMODE(COND)
  STG_DATACLAS(SHARE33)
  DIAG(YES)
```

Figure 6-9 Define the log structures and log streams for TVS

```
DEFINE LOGSTREAM
 NAME(IGWTV065.IGWLOG.SYSLOG)
 STRUCTNAME(LOG_IGWLOG_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
 LOWOFFLOAD(15) HIGHOFFLOAD(95)
 STG_DUPLEX(YES) DUPLEXMODE(COND)
 STG_DATACLAS(SHARE33)
 DIAG(YES)

 DEFINE LOGSTREAM
 NAME(IGWTV063.IGWSHUNT.SHUNTLOG)
 STRUCTNAME(LOG_IGWSHUNT_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(100)
 LOWOFFLOAD(15) HIGHOFFLOAD(95)
 STG_DUPLEX(YES) DUPLEXMODE(COND)
 STG_DATACLAS(SHARE33)
 DIAG(YES)

 DEFINE LOGSTREAM
 NAME(IGWTV064.IGWSHUNT.SHUNTLOG)
 STRUCTNAME(LOG_IGWSHUNT_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(100)
 LOWOFFLOAD(15) HIGHOFFLOAD(95)
 STG_DUPLEX(YES) DUPLEXMODE(COND)
 STG_DATACLAS(SHARE33)
 DIAG(YES)

 DEFINE LOGSTREAM
 NAME(IGWTV065.IGWSHUNT.SHUNTLOG)
 STRUCTNAME(LOG_IGWSHUNT_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(100)
 LOWOFFLOAD(15) HIGHOFFLOAD(95)
 STG_DUPLEX(YES) DUPLEXMODE(COND)
 STG_DATACLAS(SHARE33)
 DIAG(YES)
```

Figure 6-10 Define the log structures and log streams for TVS (cont)

```
DEFINE LOGSTREAM
 NAME(IGWTVS.FR.LOG001)
 STRUCTNAME(LOG_FORWARD_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(100)
 LOWOFFLOAD(0) HIGHOFFLOAD(80)
 STG_DUPLEX(NO)

 DEFINE LOGSTREAM
 NAME(IGWTVS.LOG.OF.LOGS)
 STRUCTNAME(LOG_IGWLGLGS_001)
 LS_DATACLAS(SHARE33)
 HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
 LOWOFFLOAD(0) HIGHOFFLOAD(80)
 STG_DUPLEX(NO)
```

Figure 6-11 Define the log structures and log streams for TVS (cont)

Note: IBM recommends that users specify DIAG(YES) for any TVS undo and shunt logs. This will cause the system logger to take a dump in the event of errors such as a lost block.

```
//DEFCFRPK  JOB (999,POK),'CFRM',CLASS=A,REGION=4096K,
//             MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM21) REPLACE(YES)

    CF NAME(CF01) DUMPSPACE(2048) PARTITION(E) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)

    CF NAME(CF02) DUMPSPACE(2048) PARTITION(D) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)
/*-----------------------------------------------------*/
/* RRS STRUCTURES FOR LOGSTREAMS                       */
/*-----------------------------------------------------*/
STRUCTURE NAME(RRS_ARCHIVE_1)
       INITSIZE(8000)
       SIZE(16000)
       PREFLIST(CF01,CF02)
       REBUILDPERCENT(5)
  STRUCTURE NAME(RRS_RMDATA_1)
       INITSIZE(8000)
       SIZE(16000)
       PREFLIST(CF01,CF02)
       REBUILDPERCENT(5)
  STRUCTURE NAME(RRS_MAINUR_1)
       INITSIZE(8000)
       SIZE(16000)
       PREFLIST(CF01,CF02)
       REBUILDPERCENT(5)
  STRUCTURE NAME(RRS_DELAYEDUR_1)
       INITSIZE(8000)
       SIZE(16000)
       PREFLIST(CF01,CF02)
       REBUILDPERCENT(5)
  STRUCTURE NAME(RRS_RESTART_1)
       INITSIZE(8000)
       SIZE(16000)
       PREFLIST(CF01,CF02)
       REBUILDPERCENT(5)
```

Figure 6-12 Sample JCL to define Log Streams

```
RDEFINE LOGSTRM IGWTV001.** UACC(NONE)
Granting access to the log streams:
PERMIT IGWTV001.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
PERMIT IGWTV001.** CLASS(LOGSTRM) ACCESS(READ) ID(authorized_browsers)
PERMIT IGWTV001.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(archive_userid)

RDEFINE LOGSTRM FORWARD.RECOVERY.** UACC(NONE)
RDEFINE LOGSTRM FR.LOG.** UACC(NONE)
RDEFINE LOGSTRM LOG.OF.LOGS UACC(NONE)
PERMIT FORWARD.RECOVERY.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
PERMIT FR.LOG.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
PERMIT LOG.OF.LOGS CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)

Permit all DFSMStvs instances access to each other's system logs
Do the following for each instance:
PERMIT IGWTV001.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
PERMIT IGWTV002.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
PERMIT IGWTV003.** CLASS(LOGSTRM) ACCESS(UPDATE) ID(smsvsam_userid)
Etc.
```

Figure 6-13 DFSMStvs Sample RACF Definitions

6.6.3 Define SMS constructs for DFSMStvs

- Explicit definition of DATACLAS and STORCLAS
- May also want to specify MGMTCLAS
- Log archiving
- Log data back up
- Log migration
- Log stream and log stream staging data sets are single extent VSAM linear data sets (shareoptions '3,3')
- Requires an active SMS address space
- All data sets and logs must be accessible to all members of the sysplex that are required for, or may perform, peer recovery.

6.7 DFSMStvs problem determination tips

Here we provide some tips on how you can determine the cause of an error, condition, or problem.

6.7.1 How to take a dump of the problem?

TVS uses a function called CER_ProcessErrorAndContinue when it encounters an error which recycling the server would not resolve.

- Mechanism for taking a dump without terminating the thread or the server
- It manifests as an 0F4 in module IGWFCPER but it is really generated by the calling module

6.7.2 Classes of errors

- RRS errors (for example, when attempting to register)
- System logger errors (for example, a log stream has become unavailable)
- When all locks should have been released but some are left orphaned
- When locks should have been marked retained but are not
- When an attempt to ENDREQ an RPL fails
- When an attempt to allocate or deallocate a data set fails
- When end of volume fails during backout

6.7.3 Determining the Failing Module

- Go into IPCS and open the dump.
- Go to IPCS option 6 and enter SUMMARY FORMAT.
- Find the failing TCB (start by going to the bottom and finding the one with a completion code of 0F4).
- Find the RTM2 work area and extract the value in register 13.
- On the command line enter IP IGWVRNSA <value in register 13>.
- This displays a summary of the calling chain by running the save areas.
- The failing module is the one in front of IGWFCPER.
- Any module name beginning with IGW8 or IGW9 is a DFSMStvs module and the part of TVS responsible for expressing interest in units of recovery, commit and backout.
- Modules beginning with IGW8 are part of recoverable file services.
- Modules beginning with IGW9 are part of the DFSMStvs logger.
- Reason codes in the form of 6Fxxxxxx are from recoverable file services.
- Reason codes in the form of 70xxxxxx are from the DFSMStvs logger.

6.7.4 Apparent batch job hangs

This may be caused by many different reasons other than TVS. If you feel that a batch has become hung, it could be from several reasons including:

- RLS or DFSMStvs
- Catalog
- The system logger
- RRS
- Anyone else with whom the batch job may have been communicating

Start by dumping both SMSVSAM and the batch job.

Using IPCS option 6 and SUMMARY FORMAT, find the hung TCB (usually the batch job's last TCB).

Determine with who it was cross memory using the linkage stack and dump that as well.

6.7.5 Other hangs

Here are some actions that you can take to try to analyze the cause of hang scenarios:

- First check for GRS contention: D GRS,C
- Next take a dump of SMSVSAM, including its data spaces
- Go into IPCS and enter on the command line: IP IGWVIPCS
  - Select option 1: TCB analysis

The bottom of the report should list the holders of latches and who wants them. Identify any waiters or holders of latches who are currently suspended.

6.7.6 Quiescing a data set

To quiesce a data set you need to first display the jobs that are using the data set using the following commands:

```
DISPLAY SMS,DSNAME
IDCAMS SHCDS LISTDS(dsname),JOBS
```

Once you have identified the jobs then allow them to either complete or cancel them.

To quiesce the data set use the following command:

```
VARY SMS,SMSVSAM,SPHERE
```

Perform the operations which required the data set to be quiesced.

Enable the data set for RLS and DFSMStvs use:

```
VARY SMS,SMSVSAM,SPHERE
```

6.7.7 Close/delete/rename of data set with inflight UR

Closing the data set could allow any of the following to occur:

- The data set can be deleted
- The data set can be renamed
- A PermitNonRLSUpdate can be done
- The data set can be quiesced
- The locks associated with the data set can become retained

Do not delete/rename data sets with an outstanding inflight UR, if backout is required, UR will be shunted.

Do not delete/rename data sets with shunted log records and retained locks; it can cause loss of association between the data set and its log records and locks.

A new data set could be allocated with the old name.

6.7.8 New and changed system level commands for DFSMStvs

Several commands have been either added or changed to support DFSMStvs.

DISPLAY SMS,TRANVSAM,ALL

This command displays information about the instance of DFSMStvs on this system, or on all systems in the sysplex when the ALL keyword is specified. The output includes the following information:

- The activity keypoint (AKP) trigger, which is the number of logging operations between the taking of keypoints
- The status of this instance of DFSMStvs (initializing, active, quiescing, quiesced, disabling, disabled)
- How DFSMStvs started:
  - Cold start

    The log data was not read, and any old data was discarded.
  - Warm start

    The log data was read and processed.

- DFSMStvs status with respect to resource recovery services (RRS)
- The quiesce timeout value
- All logs known to this instance of DFSMStvs, including the log of logs if one is in use
- The number of active units of recovery

Figure 6-14 is the output of the command from our test system.

```
 IEE932I 319
  IGW800I 16.32.01 DISPLAY SMS,TRANSACTIONAL VSAM,ALL
DISPLAY SMS,TRANSACTIONAL VSAM,ALL - SERVER STATUS
   System   TVSNAME  State   Rrs    #Urs     Start     AKP    QtimeOut
   -------- -------- ------ ----- -------- --------- -------- --------
   SC63     IGWTV063 ACTIVE REG          0 WARM/WARM     1000      300
   SC65     IGWTV065 ACTIVE REG          0 WARM/WARM     1000      300
   SC64     IGWTV064 ACTIVE REG          0 WARM/WARM     1000      300
  DISPLAY SMS,TRANSACTIONAL VSAM,ALL LOGSTREAM STATUS
  LogStreamName:  IGWTV063.IGWLOG.SYSLOG
   System     TVSNAME         State      Type       Connect Status
   --------   --------      ------------ ---------- --------------
   SC63      IGWTV063         Enabled    UnDoLog    Connected
 LogStreamName:  IGWTV063.IGWSHUNT.SHUNTLOG
  System     TVSNAME         State      Type       Connect Status
  --------   --------      ------------ ---------- --------------
  SC63      IGWTV063         Enabled    ShuntLog   Connected
 LogStreamName:  IGWTVS.LOG.OF.LOGS
  System     TVSNAME         State      Type       Connect Status
  --------   --------      ------------ ---------- --------------
  SC63      IGWTV063         Enabled    LogOfLogs  Connected
  SC64      IGWTV064         Enabled    LogOfLogs  Connected
  SC65      IGWTV065         Enabled    LogOfLogs  Connected
LogStreamName:  IGWTV065.IGWLOG.SYSLOG
  System     TVSNAME         State      Type       Connect Status
  --------   --------      ------------ ---------- --------------
 SC65      IGWTV065         Enabled    UnDoLog    Connected
LogStreamName:  IGWTV065.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC65      IGWTV065         Enabled    ShuntLog   Connected
LogStreamName:  IGWTV064.IGWLOG.SYSLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC64      IGWTV064         Enabled    UnDoLog    Connected
LogStreamName:  IGWTV064.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC64      IGWTV064         Enabled    ShuntLog   Connected
```

Figure 6-14 Output of D SMS, TRANSVSAM,ALL

DISPLAY SMS,SHUNTED, {SPHERE(*sphere*)|UR({*urid*|ALL})}

This command displays the entries currently contained in the shunt logs of the systems in the sysplex. Entries are moved to the shunt log when DFSMStvs is unable to finish processing a syncpoint, for example, due to an I/O error. As long as a shunted entry exists, the locks associated with that entry are retained.

These types of information can be displayed in response to this command. When neither the SPHERE nor URID keyword is specified, this command results in a list of systems in the sysplex and the number of units of recovery which that system has shunted:

- When the SPHERE keyword is specified, this command results in a list of shunted work for the sphere specified for all of the systems in the sysplex.
- When the URID keyword is specified, this command results in a list of shunted work for the unit of recovery specified for all of the systems in the sysplex. When ALL is specified, this command results in a list of shunted work for all shunted units of recovery for all the systems in the sysplex. To avoid flooding the console, DFSMStvs writes out 255 lines and then issues a WTOR to determine whether or not to continue.

If the error is correctable, the installation might choose to fix the problem and then request that DFSMStvs again attempt processing of the entry by issuing the SHCDS RETRY command. If the data set cannot be restored to a point where it is consistent with the log entries, so that it does not make sense to attempt processing of the log entry again, the installation might choose to discard the log entry by issuing the SHCDS PURGE command.

Figure 6-15 is the output of the command showing that there are no shunted units of work currently in the DFSMStvs shunt log.

```
 IEE932I 452
  IGW803I 17.19.03 DISPLAY SMS,SHUNTED (Summary Data)
SysName   # Urid(s) SysName   # Urid(s) SysName   # Urid(s)
  --------  --------  --------  --------  --------  --------
  SC63             0  SC64             0  SC65             0
```

Figure 6-15 Output of the D SMS,SHUNTED(ALL) command

DISPLAY SMS,JOB(jobname)

Displays information about a particular job that is using DFSMStvs services on one of the systems in the sysplex. The output includes:

- The name of the current step within the job
- The current URID for the job
- The status of the unit of recovery (in-reset, in-flight, in-prepare, in-commit, in-backout, indoubt)

DISPLAY SMS,URID(*urid* |ALL)

This command displays information about a particular unit of recovery currently active within the sysplex or about all units of recovery currently active on the system, on which the command was issued, on whose behalf DFSMStvs has performed any work. This parameter does not include information about work that has been shunted, because you can use the DISPLAY SMS,SHUNTED command to display that information. This parameter also does not include information about units of recovery that might be in restart processing as a result of an earlier failure. This work is not considered to be currently active, because it is not associated with any batch job, and the units of recovery associated with the work will end as soon as commit or backout processing for them can be completed. The output includes this information:

- The age of the unit of recovery
- The name of the job with which the unit of recovery is associated
- The name of the current step within the job
- The status of the unit of recovery (in-reset, in-flight, in-prepare, in-commit, in-backout, indoubt)
- The user ID associated with the job

Figure 6-16 shows the output from the command.

```
RESPONSE=SC63
 IEE932I 489
 IGW802I DFSMS REQUEST TO DISPLAY ACTIVE TRANSACTIONAL VSAM  UR(s)
 WAS REJECTED,  SPECIFIED URID(s) ARE NOT ACTIVE
 ON ANY TRANSACTIONAL VSAM INSTANCE IN THE SYSPLEX.
```

Figure 6-16 Output of D SMS,URID(ALL) cDFSMStvsommandDFSMStvs

DISPLAY SMS,LOG(*logstreamid*|ALL)

This command displays information about a log stream that DFSMStvs is currently using on one of the systems in the sysplex. If ALL is specified, information is displayed about all of the logs in use on the entire sysplex. The output includes the status of the log stream (failed or available), type of log (undo, shunt, forward recovery, or log of logs), the job name and URID of the oldest unit of recovery using the log, and a list of all DFSMStvs instances that are using the log.

If information about a specific log stream is requested and the log stream is either a system log or a forward recovery log, the output includes the names of the jobs using the log stream.

This command might be issued to determine why a log stream is increasing in size. If a unit of recovery is long running, DFSMStvs would be unable to delete any log blocks that contain data associated with the unit of recovery, which in turn would make truncation of the log stream impossible.

Figure 6-17 shows the output of this command using a specific logstream name, `IGWTV063.IGWLOG.SYSLOG`.

```
RESPONSE=SC63
 IEE932I 512
 IGW804I 17.54.29 DISPLAY SMS,LOG
 DISPLAY SMS,LOG      - LOG STREAM STATUS
   Name: IGWTV063.IGWLOG.SYSLOG     State: Enabled      Type: UnDo
   System   TVSNAME  JobName  Urid of Oldest Log Block
   -------- -------- -------- --------------------------------
   SC63     IGWTV063 **NONE** --------- NO ACTIVE UR ---------*
 DISPLAY SMS,LOG      - LOG STREAM USAGE
   LogStreamName: IGWTV063.IGWLOG.SYSLOG
   System   TVSNAME  JobName  JobName  JobName  JobName  JobName
   -------- -------- -------- -------- -------- -------- --------
   SC63     IGWTV063 **NONE**
 *OLDEST URID ACROSS ALL SYSTEMS IN THE SYSPLEX
```

Figure 6-17 Output of D SMS,LOG(IGWTV063.IGWLOG.SYSLOG command

DISPLAY SMS,DSNAME(dsn)

For a given fully qualified data set name, this command displays the jobs currently accessing the data set using DFSMStvs on the systems within the sysplex. Figure 6-18 shows that the specified data set is not open for DFSMStvs access.

```
IGW805I DFSMS REQUEST TO DISPLAY TRANSACTIONAL VSAM USAGE OF
 DATASET: IGWTV063.IGWLOG.SYSLOG WAS REJECTED.
 DATASET NOT KNOWN TO TRANSACTIONAL VSAM.
```

Figure 6-18 Output of the D SMS, DSNAME(dsn) commandDFSMStvs

DISPLAY SMS,OPTIONS

This command displays all of the SMS parameters and their status at the time this command is issued. The display indicates whether each option is on or off, what data sets are being used, the size of regions, the time interval for recording data, and all other parameter specifics. When DFSMStvs is running on the system, the output of this command includes DFSMStvs-related information. In the output shown in Figure 6-19, we only show the DFSMStvs related information

|

```
 IGD002I 18:31:24 DISPLAY SMS 578
  /* This output has been edited to remove all non DFSMStvs related data*/
LOCAL_DEADLOCK = 15
  GLOBAL_DEADLOCK = 4
  REVERIFY = NO
  ACSDEFAULTS = NO
  DSNTYPE = PDS
  PDSESHARING = EXTENDED
RLS_MAX_POOL_SIZE = 100MB
  RLSINIT = YES
  RLSTMOUT = 0
  COMPRESS = GENERIC
  LOG_OF_LOGS = IGWTVS.LOG.OF.LOGS
  QTIMEOUT = 300
  TVSNAME = 063
  AKP = 1000
  TV_START_TYPE = WARM
  MAXLOCKS = (0,0)
  CICSVR_INIT = YES
  CICSVR_DSNAME_PREFIX = DWWUSER.V3R1M0
  Rls_MaxCfFeatureLevel = Z
  PDSE_MONITOR = (YES,0,0)
```

Figure 6-19 Modified Output from the D SMS,OPTIONS command

6.7.9 SET SMS and SETSMS commands

The SET SMS command and the SETSMS command use the same subparameters as the new DISPLAY command subparameter described above.

- **SET SMS**

  This command, with the subparameters supplied, *prepares* the storage management subsystem environment.

- **SETSMS**

  This command, with the subparameters supplied, *activates* the storage management subsystem environment.

6.7.10 VARY SMS command

The VARY SMS command, with the subparameters supplied, changes the state of the storage management subsystem environment.

Note: When using the VARY SMS for TVS, the following subparameters are valid:

- LOG
- SMSVSAM,SPHERE
- TRANVSAM
- TRANVSAM(*nn*n),PEERRECOVERY

- **Log**

  This enables, quiesces, or disables DFSMStvs access to a log stream.
- **SMSVSAM,SPHERE**

  This quiesces or unquiesces a data set for DFSMStvs or RLS access.
- **TRANVSAM**

  This enables, quiesces, or disables a DFSMStvs instance or all DFSMStvs instances in the sysplex.
- **TRANVSAM(nnn),PEERRECOVERY**

  This starts or stops peer recovery processing for a failed instance of DFSMStvs.

For more detailed information on new and changed system level commands, see *z/OS, V1R4 DFSMStvs Planning and Operating Guide*, GC26-7485.

6.7.11 SYS1.PARMLIB changes

IFAPRDxx member of PARMLIB needs to be updated to reflect the addition of the priced feature DFSMStvs.

Important: DFSMStvs is a priced feature of DFSMS. To be able to initialize DFSMStvs, users *must* update the IFAPRDxx member of PARMLIB

Here is an example of the entry to IFAPRDxx that was used on our test system;

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        ID(5694-A01)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(DFSMSTVS)
        STATE(ENABLED)
```

Figure 6-20 Sample IFAPRDxx parmlib member to enable DFSMStvs

Figure 6-21 shows sample IGDSMSxx member in SYS1.PARMLIB. The parameters relevant to DFSMStvs are in bold letters. Each variable is described below.

```
SMSACDS(acds)
   COMMDS(commds)
   INTERVAL(nnn|15)
   DINTERVAL(nnn|150)
   REVERIFY(YES|NO)
   ACSDEFAULTS(YES|NO)
   SYSTEMS(8|32)
   TRACE(OFF|ON)
   SIZE(nnnnnK|M)
   TYPE(ALL|ERROR)
   JOBNAME(jobname|*)
   ASID(asid|*)
   SELECT(event,event....)
   DESELECT(event,event....)
   DSNTYPE(LIBRARY|PDS)
   RLSINIT(NO|YES)
   RLS_MAX_POOL_SIZE(nnn|100)
   SMF_TIME(NO|YES)
   CF_TIME(nnn|3600)
   BMFTIME(nnn|3600)
   CACHETIME(nnn|3600)
   DEADLOCK_DETECTION(iii|15,kkk|4)
   RLSTMOUT(nnn|0)
   SYSNAME(sys1,sys2....)
   TVSNAME(nnn1,nnn2....)
   TV_START_TYPE(WARM|COLD,WARM|COLD...)
   AKP(nnn|1000,nnn|1000)
   LOG_OF_LOGS(logstream)
   QTIMEOUT(nnn|300)
   MAXLOCKS(max|0,incr|0)
```

Figure 6-21 Sample SYS1.PARMLIB(IGDSMSxx) member

- **RLSINIT(NO|YES)**

  This is used to activate the SMSVSAM address space.

- **RLS_MAX_POOL_SIZE(nnn|100)**

  This specifies the maximum size, in megabytes, of the SMSVSAM local buffer pool.

- **SMF_TIME(NO|YES)**

  This is used to align the interval-type SMF 42 records for DFSMS with the SMF_TIME interval.

- **CF_TIME(nnn|3600)**

  This is used to align creation of all CF related SMF 42 subtypes.

- **BMFTIME(nnn|3600)**

  This specifies the number of seconds that SMS is to wait between recording SMF records for buffer management facility (BMF) cache use. You can specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds), and the default is 3600 (one hour). The SMF_TIME keyword, if set to `YES`, overrides the BMFTIME keyword.

- **CACHETIME(nnn|3600)**

  This specifies the number of seconds between recording SMF records for device cache use. The CACHETIME parameter applies only to the volumes behind an IBM 3990 Storage Control with cache unit. You can specify a value from 1 to 86399 (23 hours, 59 minutes, 59 seconds), and the default is 3600 (one hour). The SMF_TIME keyword, if set to `YES`, overrides the CACHETIME keyword.

- **DEADLOCK_DETECTION(iii|15,kkk|4)**

  This specifies the interval for detecting deadlocks between systems.

- **RLSTMOUT(nnn|0)**

  This specifies a timeout value for DFSMStvs requests for required locks.

- **SYSNAME(sys1,sys2....)**

  This specifies the name or names of the systems on which DFSMStvs instances are to run.

- **TVSNAME(nnn1,nnn2....)**

  This specifies the identifier or identifiers of DFSMStvs instances that are to run in the sysplex.

- **TV_START_TYPE(WARM|COLD,WARM|COLD...)**

  This specifies the type of start that each instance of DFSMStvs is to perform.

- **AKP(nnn|1000,nnn|1000)**

  This specifies the activity keypoint trigger value, which is the number of logging operations between taking keypoints, for one or more.

- **LOG_OF_LOGS(logstream)**

  This specifies the log stream that is to be used as the log of logs.

- **QTIMEOUT(nnn|300)**

  This specifies the quiesce exit timeout value.

- **MAXLOCKS(max|0,incr|0)**

  Specifies the maximum number of unique lock requests that a single unit of recovery can make.

For more detailed information about new and changed subparameters in IGDSMSxx, see *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348.

6.7.12 Changes to Job Control Language (JCL)

Here is a list of the JCL changes to support DFSMStvs.

- **RLSTMOUT**

  This subparameter of the EXEC statement specifies a timeout value for DFSMStvs requests for required locks.

  - **CRE**

    This subparameter of the RLS statement obtains a shared lock for VSAM record-level sharing (RLS).

For more detailed information on these changes, see *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348.

6.7.13 Changes to IDCAMS

There are numerous changes and additions to IDCAMS commands. Here are some of the modifications.

- **ALLOCATE**

  Here is a list of changes to the ALLOCATE command.

  - **BWO(TYPECICS)**

    The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this parameter activates BWO processing for DFSMStvs.

  - **DATACLASS CISIZE**

    For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

  - **SHAREOPTIONS**

    When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).

- **ALTER**

  Here is a list of changes to the ALTER command.

  - **BWO(TYPECICS)**

    The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this activates BWO processing for DFSMStvs.

  - **LOG(UNDO)**

    This specifies that changes to the sphere accessed in DFSMStvs mode can be backed out using an external log. DFSMStvs considers the sphere recoverable.

  - **LOG(ALL)**

    This specifies that changes to the sphere accessed in DFSMStvs mode can be backed out and is also forward recovered using external logs. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined.

  - **LOGSTREAMID**

    This changes or adds the name of the DFSMStvs forward-recovery log stream, for all components in the VSAM sphere.

- **DEFINE ALTERNATEINDEX**

  The changes to this command are listed below.

  - **BUFFERSPACE**

    When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.

  - **CONTROLINTERVALSIZE**

    For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

  - **KEYRANGES**

    You cannot open key range data sets for RLS or DFSMStvs processing because DFSMS no longer supports this parameter (as of V1R3).

  - **SHAREOPTIONS**

    When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).

  - **WRITECHECK**

    When you use DFSMStvs access, DFSMS ignores the WRITECHECK parameter.

- **DEFINE CLUSTER**

  Changes to the DEFINE CLUSTER are described below.

- **BUFFERSPACE**

  When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.

- **BWO(TYPECICS)**

  The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this activates BWO processing for DFSMStvs.

- **CONTROLINTERVALSIZE**

  For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

- **KEYRANGES**

  You cannot open key range data sets for RLS or DFSMStvs processing, because DFSMS no longer supports this parameter (as of V1R3).

- **LOG(UNDO)**

  This specifies that changes to the sphere accessed in DFSMStvs mode can be backed out using an external log. DFSMStvs considers the sphere recoverable.

- **LOG(ALL)**

  This specifies that changes to the sphere accessed in DFSMStvs mode can be backed out and it is also forward recovered using external logs. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined. If you use LOG(NONE), DFSMStvs considers the sphere to be nonrecoverable.

- **LOGSTREAMID**

  Changes or adds the name of the DFSMStvs forward-recovery log stream, for all components in the VSAM sphere.

- **SHAREOPTIONS**

  When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).

- **WRITECHECK**

  When you use RLS or DFSMStvs access, DFSMS ignores the WRITECHECK parameter.

- **DEFINE PATH**

  Here are the changes to the DEFINE PATH command.

  - **NOUPDATE**

    This has the same meaning for DFSMStvs as it does for RLS.

- **SHCDS**

  Here are the changes to the SHCDS command.

  - **LISTDS**

    A new, optional JOBS keyword returns a list of the jobs that currently access the data set in DFSMStvs mode.

  - **LISTSHUNTED**

    This lists information about work that was shunted due to an inability to complete a syncpoint (commit or backout) for a data set, a unit of recovery, or all shunted units of recovery.

  - **PURGE**

    This discards log entries and releases the associated locks, for use when a data set is damaged and you cannot restore it to a state that is consistent with the log entries.

  - **RETRY**

    This retries the syncpoint; it is for use when you can restore a data set to a state that is consistent with the log entries.

6.7.14 Messages and codes

For DFSMStvs descriptions of VSAM return and reason codes, see *DFSMStvs Administration Guide*, GC26-7483.

For descriptions of new and changed DFSMSdfp messages and return and reason codes for DFSMStvs, refer to *z/OS V1R4 DFSMStvs Messages and Code*s.

For a listing of other new and changed DFSMSdfp messages and return and reason codes, see *z/OS Summary of Message Change*s.

6.7.15 Macros that have been changed to support DFSMStvs

Several macros have been changed to add new subparameters in support of DFSMStvs. Here is a list of the macros that have either changed or that you need to understand if your application is going to be DFSMStvs enabled:

- The ACB Macro
- The GENCB Macro

Refer to this publication for detailed information: *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348.

A

Sample code

This appendix contains useful JCL and code samples that can be modified and used in your installation.

JRIO API examples

Here some examples of JRIO APIs for accessing VSAM data sets:

Locate a record by key in keyed access record file

```
IKeyedAccessRecordFile karf =
    new KeyedAccessRecordFile("//JOE.KSDS",
                     ;         JRIO_READ_MODE);
IRecordFile index = karf.getPrimaryIndex();
karf.positionForward(index, key);
karf.read(index, buffer);
karf.close();
```

Position to a record in a random access record file

```
IRandomAccessRecordFile rarf =
    new RandomAccessRecordFile("//JOE.KSDS",
                     ;         JRIO_READ_MODE);

rarf.read(buffer);   // reads the first record (record 0)

rarf.positionLast();
rarf.positionPrev();
rarf.read(buffer);   // reads the last record (record n-1)

rarf.positionFirst();
rarf.positionNext();
rarf.read(buffer);   // reads the second record (record 1)

rarf.seek(5L);
rarf.read(buffer);   // reads the sixth record (record 5)

rarf.close();
```

Read a record from a keyed access record file

```
IKeyedAccessRecordFile karf =
      new KeyedAccessRecordFile("//JOE.KSDS");
IRecordFile index = karf.getPrimaryIndex();
// or: IRecordFile index =
//       karf.getAlternateIndex("//JOE.KSDS.AIX");


byte[ ] buffer = new byte[JRIO_MAX_RECORD_LENGTH];

for(;;)
```

```
{
    // optional: karf.positionForward(key);
    // or:       karf.positionForwardGE(key);
    int bytesRead = karf.read(index, buffer);

    if (bytesRead != JRIO_READ_EOF)
    {
        // process(buffer);
    }
    else
    {
        break;
    }
}
karf.close();
```

Read a record from a random access record file

```
IRandomAccessRecordFile rarf =
       new RandomAccessRecordFile("//JOE.SEQ");

 byte[ ] buffer = new byte[JRIO_MAX_RECORD_LENGTH];

 for(;;)
 {
     // rarf.seek(recNo);
     int bytesRead = rarf.read(buffer);

     if (bytesRead != JRIO_READ_EOF)
     {
         // process(buffer);
     }
     else
     {
         break;
     }
 }
 rarf.close();
```

Update a record in a keyed access record file

```
IKeyedAccessRecordFile karf =
    new KeyedAccessRecordFile("//JOE.KSDS",
                      ;         JRIO_READ_WRITE_MODE);
IRecordFile index = karf.getPrimaryIndex();
karf.read(index, buffer);
// modify non-key field(s) in buffer
```

```
karf.update(index, buffer);
karf.close();
```

Accessing the VSAM Shared Information (VSI)

You can code the following instructions to get the length and address of VSI the data to be sent to anotherOS/390 image:

- Load ACB address into register RY.
- To locate the VSI for a data component:

`L   RX,04(,RY)`	Put AMBL address into register RX
`L   1,52(,RX)`	Get data AMB address
`L   1,68(,1)`	Get VSI address
`LH  0,62(,1)`	Load data length
`LA  1,62(,1)`	Point to data to be communicated

- To locate the VSI information for an index component of a key-sequenced data set:

`L   RX,04(,RY)`	Put AMBL address into register RX
`L   1,56(,RX)`	Get index AMB address
`L   1,68(,1)`	Get VSI address
`LH  0,62(,1)`	Load data length
`LA  1,62(,1)`	Point to data to be communicated

Similarly, the location of the VSI on the receiving processor can be located. The VSI level number must be incremented in the receiving VSI to inform the receiving processor that the VSI has changed. To update the level number, assuming the address of the VSI is in register 1:

`LA  0,1`	Place increment into register 0
`AL  0,64(,1)`	Add level number to increment
`ST  0,64(,1)`	Save new level number

All processing of the VSI must be protected by using ENQ/DEQ to prevent simultaneous updates to the transmitted data.

Sample programs extract from SMF record type 64

In this section we provide two sample programs:

1. SMF64 that helps you identify VSAM data sets with heavy access and good candidates for SMB.
2. SMFLSR helps you identify VSAM data sets opened in LSR mode. Once they are identified, you can use the data set names as input to IBM tool to see the probable change in CIsize with the new CI size calculation algorithm.

SMF64 sample code

The sample in Example A-1 can be used to find more information about jobs using heavily VSAM data sets. It is based on SMF record type 64 issued when data set component is closed.

See the documentation inside the Example A-1 for:

- ► The parms you can use to extract information
- ► The JCL you use to extract SMF records from SYS1.MAN*
- ► The JCL to execute the sample program
- ► The report description fields

Example: A-1 SMF64 sample code

```
//*-----------------------------------------------------------------*
//*  JCL:                                                           *
//*  //RELAT   EXEC PGM=SMF64,PARM=SEE_BELOW                        *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                        *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                        *
//*  //REPORT   DD SYSOUT=*,LRECL=133                               *
//*                                                                 *
//*  THE PARM IS OPTIONAL: DEFAULT(EXCP=10000)                      *
//* ==> YOU CAN USE **ONLY** 1 OF 3 LISTED BELOW                    *
//*    DSN=DATA_SET_NAME                                            *
//*        LISTS INFORMATION ABOUT VSAM DATA SETS WHOSE NAME BEGINS  *
//*        WITH "DATA_SET_NAME".                                    *
//*        EX.: 'DSN=MY.DATA'  LISTS INFORMATION ABOUT ALL VSAM DATA *
//*             BEGINING WITH MY.DATA*                              *
//*        EX.: 'DSN=MY.DATA '  LIST ONLY DATA SET MY.DATA          *
//*                                                                 *
//*    JOB=MYJOB                                                    *
//*        LISTS ACCESS TO VSAM DATA SETS DONE BY ALL JOBS MYJOB*    *
//*        EX: PARM='JOB=MY.JOB'  LIST ALL JOBS MYJOB*              *
//*        EX: PARM='JOB=MYJOB ' LIST ONLY ABOUT MYJOB              *
//*                                                                 *
//*    EXCP=MAX_6_BYTES  LIST ONLY FOR DATA SET WITH EXCP>=PARM_EXCP *
//*                                                                 *
//* ==> YOU CAN A DATE AND/OR AN INTERVAL TO LIST                      *
//*    JD=AADDD  JULIAN DATE                                            *
//*    BH=HH INTERVAL BEGINING HOUR                                     *
//*    EH=HH INTERVAL FINAL HOUR                                        *
//*                                                                     *
//*    EXAMPLES:                                                        *
//*    PARM='DSN=XXX,BH=07,EH=10'                                       *
//*    PARM='DSN=XXX,JD=02360'                                          *
//*    PARM='EXCP=NNNN'                                                 *
//*    PARM='JOB=JJJ,JD=02360,BH=07,EH=07'  (FROM 07:00 TO 07:59)       *
//*  -------------------------------------------------------------      *
//*  -------------------------------------------------------------      *
//*  JCL:  DUMP  SMF 64 RECORD FROM SYS1.MAN                            *
//*  //STEP1     EXEC  PGM=IFASMFDP
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA
//*  //SYSPRINT  DD    SYSOUT=A
//*  //SYSIN     DD  *
//*    INDD(INDD,OPTIONS(DUMP))
//*    OUTDD(DUMPOUT,TYPE(64:64))                                       *
//*                                                                     *
//* REPORT DESCRIPTION                                                  *
//* XF=EXTENDED FORMAT DATA SET    N-XF(NON EXTENDED FORMAT)            *
//* XP=COMPRESSED FORMAT DATA SET  N-XP(NON COMPRESSED)                 *
//* XA=EXTENDED ADDRESSABLE DATA SET  N-XA(NON EXTENDED ADDRESSAB.)     *
//* INSERT,DELETE,READ,UPDATE,GET, EXCP, CA AND CI SPLITS  ARE FROM     *
//*    OPEN TO CLOSE (ARE NOT ACCUMULATED)                              *
//* ACCESS BY; KEY, RBA, SEQ, DIR, SKIP                                 *
//* PROCESSING OPTIONS:                                                 *
//*   CNV: CONTROL INTERVAL ACCESS                                      *
//*   IN: INPUT                                                         *
//*   OUT: OUTPUT                                                       *
//*   ICI: IMPROVED CONTROL INTERVAL PROCESSING                         *
//*   DW: DEFERRED WRITE                                                *
//*   SIS: SEQUENTIAL INSERT STRATEGY                                   *
//*   UB:  USER MANAGED BUFFERING                                       *
//*   CBSHR: VSAM STRUCTURE CONTROL BLOCK SHARED                        *
//*   CBFIX: VSAM CONTROL BLOCKS AND BUFFERS FIXED IN REAL STORAGE      *
//*   BUFFERING MANAGEMENT: LSR, GSR, NSR, RLS                          *
//*   BUFF31: 31-BIT ADDRESSING MODE FOR BUFFERS                        *
//*   BUFF24: 24-BIT ADDRESSING MODE FOR BUFFERS                        *
//*   SMB: OPTIMIZATION  USED:DO,DW,SO,SW,(OR CO,CR WHEN LOAD)          *
//*        VSP: USER SPECIFIED AMOUNT OF VIRTUAL STORAGE THRU SMBVSP    *
//*        HWT: USER SPECIFIED HIPERSPACE BUFFER SMBHWT                 *
//*        BUFF31: SMB USED 31-BIT ADDRESSING MODE FOR BUFFERS          *
//*        CB31: SMB USED 31-BIT ADDRESSING MODE FOR CONTROL BLOCKS     *
//*        IR: INSUFFICIENT VIRTUAL STORAGE FOR DO PROCESSING           *
//*-------------------------------------------------------------------*
```

```
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - VSAM STATS'
SMF64 CSECT
SMF64 AMODE 31
SMF64 RMODE 24
         STM   R14,R12,12(R13)      * LINKAGE CONVENTION
         LR    R12,R15              * R12 BASE
         USING SMF64,R12
         LA    R15,SAVE
         ST    R15,8(R13)
         ST    R13,SAVE+4
         LR    R13,R15
         L     R7,LIM_EXCP          * DEFAULT EXCPS
         LA    R6,RECORD            * REPORT MAPPING
         L     R5,0(R1)             * PARM
         LH    R4,0(R5)             * LENGTH
         LTR   R4,R4                * NO PARM?
         BZ    DO_OPEN              * YES, USE DEFAULT
         BCTR  R4,0                 * -1   EXECUTE
         LA    R5,2(,R5)            *
         LA    R14,DO_OPEN
SEARCHP  EX    R4,TRT_PARM
         BNZ   CALC_LEN
         LR    R1,R4
         LA    R4,0
         B     ID_PARM
*
CALC_LEN LA    R8,1(,R1)
         SR    R1,R5
         SR    R4,R1
         BCTR  R1,0
         BAS   R14,ID_PARM
         LTR   R4,R4
         BZ    DO_OPEN
         LR    R5,R8
         BCTR  R4,0
         B     SEARCHP
*
ID_PARM  CLC   0(4,R5),=C'DSN='
         BNE   PARMJOB
         LA    R3,4
         SR    R1,R3
         BZR   R14
         LR    R11,R1
         EX    R1,MVDSN
         OI    FLAG,X'01'             * SELECT BY DSN
         MVI   TIT_OPT,X'40'
         MVC   TIT_OPT+1(L'TIT_OPT-1),TIT_OPT
         MVC   TIT_OPT(4),=C'DSN:'
         MVC   TIT_OPT+4(L'RDSN),RDSN
         BR    R14
*
PARMJOB  CLC   0(4,R5),=C'JOB='
         BNE   PARMEXCP
         LA    R3,4
         SR    R1,R3
         BNPR  R14
         LR    R11,R1
         EX    R1,MVJOBN
         OI    FLAG,X'02'          * SELECT BY JOBN
         MVI   TIT_OPT,X'40'
         MVC   TIT_OPT+1(L'TIT_OPT-1),TIT_OPT
         MVC   TIT_OPT(8),=C'JOBNAME:'
         MVC   TIT_OPT+9(L'RJOB),RJOB
         BR    R14
*
PARMEXCP CLC   0(5,R5),=C'EXCP='
         BNE   PARMJD
         LA    R3,5
         SR    R1,R3
         BNPR  R14
         MVI   T_EXCP,X'40'
         MVC   T_EXCP+1(L'T_EXCP-1),T_EXCP
         EX    R1,MVEXCP
         EX    R1,PACK
         CVB   R7,DOUBLE
         ST    R7,LIM_EXCP
         BR    R14
*
PARMJD   CLC   0(3,R5),=C'JD='
         BNE   PARMBH
         LA    R3,7
         CR    R1,R3
         BNER  R14                 * INVALID. IGNORES
         PACK  DATAJ,3(5,R5)
         OI    DATAJ+3,X'0F'
         OI    FLAG,X'04'
         BR    R14
*
PARMBH   CLC   0(3,R5),=C'BH='
         BNE   PARMEH              * INVALID, IGNORES
         LA    R3,4
         CR    R1,R3
         BNER  R14                 * INVALID. IGNORES
         TM    3(R5),X'F0'
         BNOR  R14
         TM    4(R5),X'F0'
         BNOR  R14
         PACK  DOUBLE,3(2,R5)
         CVB   R3,DOUBLE
         LTR   R3,R3
         BZR   R14
         M     R2,F3600           * SECONDS
         M     R2,F100            * SECONDS * 100
         ST    R3,T_INIT
         OI    FLAG,X'08'         * FLAG INITIAL INTERVAL
         BR    R14
*
PARMEH   CLC   0(3,R5),=C'EH='
         BNER  R14
         LA    R3,4
         CR    R1,R3
         BNER  R14                * INVALID. IGNORES
         TM    3(R5),X'F0'
         BNOR  R14
         TM    4(R5),X'F0'
         BNOR  R14
         PACK  DOUBLE,3(2,R5)
         CVB   R3,DOUBLE
         LTR   R3,R3
         BZR   R14
         LA    R3,1(,R3)          * HOUR LIMIT FOR COMPARATION
         M     R2,F100            *
         M     R2,F3600           * SECONDS * 100
         BCTR  R3,0               *
         ST    R3,T_FIM
         OI    FLAG,X'80'         * FINAL INTERVAL FLAG
         BR    R14
*
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
         LA    R8,0               * TITLE CONTROL
NUMLIN   LA    R9,55              * 55 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1             * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64        * 64?
         BNE   READ
         CLI   2(R10),X'00'
         BNE   READ
         TM    SMF64RIN,X'80'     * COMPONENT CLOSE?
         BZ    READ
         TM    SMF64DTY,X'80'      * DATA SET?
         BNO   READ
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         TM    FLAG,X'01'          * DSN?
         BO    PR_DSN
         TM    FLAG,X'02'          * JOB?
         BNO   PR_EXCP
*
PR_JOB   EQU   *
         EX    R11,CPJOB           * JOBNAME MATCHES?
         BNE   READ
         B     COMMON
*
PR_DSN   EQU   *
         EX    R11,CPDSN           * DSN MATCHES?
         BNE   READ
         B     COMMON
*
PR_EXCP  EQU   *
         DROP  R10
         USING SMFRCD64,R15
         L     R3,SMF64DEP          EXCP FOR DATA SET
         CR    R3,R7                COMPARE WITH LIMIT
         BL    READ
         B     COMMON
         DROP  R15
*
COMMON DS      0H
         USING SMFRCD64,R10        SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID    SYSID
         MVC   R_JOB,SMF64JBN      JOBNAME
         TM    FLAG,X'04'          DATE?
         BNO   MV_DATA
         CP    SMF64DTE+1(3),DATAJ+1(3)  COMPARE DATE
         BNE   READ
MV_DATA  UNPK  DATA,SMF64DTE
         TM    FLAG,X'88'          INTERVAL SELECTION?
         BNO   MV_TME              IGNORES IF NOT COMPLETE INTERVAL
         CLC   SMF64TME,T_INIT
         BL    READ
         CLC   SMF64TME,T_FIM
         BH    READ
MV_TME   ICM   R3,B'1111',SMF64TME
         L     R1,F100
         LA    R2,0
         DR    R2,R1              * SECONDS
         L     R1,F3600            * SECONDS IN  HOURS
         LA    R2,0
         DR    R2,R1               * HOUR EM R3
         CVD   R3,DOUBLE
         UNPK  HH,DOUBLE+6(2)
         OI    HH+1,X'F0'
         L     R1,F60
         LR    R3,R2                * REMAINING
         LA    R2,0
         DR    R2,R1                * MINUTS
         CVD   R3,DOUBLE
         UNPK  MM,DOUBLE+6(2)
         OI    MM+1,X'F0'
         CVD   R2,DOUBLE            * REMAINING SECOUNDS
         UNPK  SS,DOUBLE+6(2)
         OI    SS+1,X'F0'
         LA    R4,R_ACCESS
         TM    SMF64DTY,X'20'       EXTENDED FORMAT?
         BNO   N_XF                 DOES NOT USES EXT FACILITIES
         MVC   0(L'XF,R4),XF
         LA    R4,L'XF+1(,R4)
         TM    SMF64DTY,X'10'       * COMPRESSED?
         BNO   IF_XA                  NO. JUMP
         MVC   0(2,R4),=C'XP'       SET COMPRESSED
         LA    R4,3(,R4)
IF_XA    TM    SMF64DTY,X'02'       EXTENDED ADDRESS DATA SET?
         BNO   N_XA
         MVC   0(L'XA,R4),XA        SET EXTENDED ADDRESSABILITY
         LA    R4,L'XA+1(,R4)
         B     IF_RLS
*
N_XF     MVC   0(L'NXF,R4),NXF      SET NON EXTENDED FORMAT
         LA    R4,L'NXF+1(,R4)
*
N_XA     MVC   0(L'NXA,R4),NXA      SET NON EXTENDED ADDRESS DATA
         LA    R4,L'NXA+1(,R4)
IF_RLS   TM    SMF64DTY,X'08'       RECORD LEVEL SHARED?
         BNO   M_EXCP
         MVC   0(L'RLS,R4),RLS      SET RLS
         LA    R4,L'RLS+1(,R4)
*
         DROP  R10
         USING SMFRCD64,R15
*
M_EXCP   L     R3,SMF64DEP           EXCPS
         CVD   R3,DOUBLE             ZONED DECIMAL
         MVC   EXCPS,MODEL
         ED    EXCPS,DOUBLE+2
         MVC   R_DDN,SMF64DDN       * DDNAME
         MVC   R_DSN,SMF64CLN       * CLUSTER NAME
         L     R3,SMF64DIN          * INSERTIONS
         CVD   R3,DOUBLE
         MVC   INSERT,MODEL
         ED    INSERT,DOUBLE+2
         L     R3,SMF64DDE          * RECORDS DELETED
         CVD   R3,DOUBLE
         MVC   DELETE,MODEL
         ED    DELETE,DOUBLE+2
         L     R3,SMF64DUP          * RECORDS UPDATED
         CVD   R3,DOUBLE
         MVC   UPDATE,MODEL
         ED    UPDATE,DOUBLE+2
         L     R3,SMF64DRE          * RECORDS RETRIEVED
         CVD   R3,DOUBLE
         MVC   READS,MODEL
         ED    READS,DOUBLE+2
         L     R3,SMF64DCS          * CONTROL INTERVAL SPLITS
         CVD   R3,DOUBLE
         MVC   CI_SPLIT,MODEL
         ED    CI_SPLIT,DOUBLE+2
         L     R3,SMF64DAS          * CONTROL AREA SPLITS
         CVD   R3,DOUBLE
         MVC   CA_SPLIT,MODEL
         ED    CA_SPLIT,DOUBLE+2
         TM    SMF64MC1,X'80'       * KEY ACCESS?
         BNO   IF_RBA
         MVC   0(3,R4),=C'KEY'
         LA    R4,4(R4)
         B     TYPEACC
*
IF_RBA   TM    SMF64MC1,X'40'       * RBA?
         BNO   IF_CI
         MVC   0(3,R4),=C'RBA'
         LA    R4,4(R4)
         B     TYPEACC
*
IF_CI    TM    SMF64MC1,X'20'       * ACCESS BY CONTROL INTERVAL?
         BNO   TYPEACC
         MVC   0(3,R4),=C'CNV'
         LA    R4,4(R4)
         B     TYPEACC
*
TYPEACC  TM    SMF64MC1,X'10'       * SEQUENTIAL ACCESS?
         BNO   IF_DIR
         MVC   0(3,R4),=C'SEQ'
         LA    R4,4(R4)
*
IF_DIR   TM    SMF64MC1,X'08'       * DIRECT?
```

```
         BNO   IF_INP
         MVC   0(3,R4),=C'DIR'
         LA    R4,4(R4)
IF_INP   TM    SMF64MC1,X'04'      * INPUT?
         BNO   IF_OUT
         MVC   0(2,R4),=C'IN'
         LA    R4,3(R4)
*
IF_OUT   TM    SMF64MC1,X'02'      * OUTPUT?
         BNO   IF_UBUF
         MVC   0(3,R4),=C'OUT'
         LA    R4,4(R4)
IF_UBUF  TM    SMF64MC1,X'01'      * USER MANAGED BUFFERING
         BNO   MC2
         MVC   0(2,R4),=C'UB'
         LA    R4,3(R4)
MC2      TM    SMF64MC2,X'10'     SKIP SEQUENTIAL?
         BNO   IF_CBS
         MVC   0(4,R4),=C'SKIP'
         LA    R4,5(R4)
IF_CBS   TM    SMF64MC2,X'02'     SHARED CONTROL BLOCKS?
         BNO   IF_LSR
         MVC   0(5,R4),=C'CBSHR'
         LA    R4,6(R4)
IF_LSR   TM    SMF64MC3,X'40'     LSR?
         BNO   IF_GSR
         MVC   0(3,R4),=C'LSR'
         LA    R4,4(R4)
         B     IF_DW
*
IF_GSR   TM    SMF64MC3,X'20'     GSR
         BNO   IF_ICI
         MVC   0(3,R4),=C'GSR'
         LA    R4,4(R4)
         B     IF_DW
*
IF_ICI   TM    SMF64MC3,X'10'     IMPROVED CONTROL-INTERVAL
         BNO   NSR
         MVC   0(3,R4),=C'ICI'
         LA    R4,4(R4)
         B     IF_DW
*
NSR      MVC   0(3,R4),=C'NSR'
         LA    R4,4(R4)
*
IF_DW    TM    SMF64MC3,X'08'     DEFERRED-WRITE?
         BNO   IF_SIS
         MVC   0(2,R4),=C'DW'
         LA    R4,3(R4)
```

```
IF_SIS   TM    SMF64MC3,X'04'     SEQUENTIAL INSERT STRATEGY?
         BNO   IF_FIX
         MVC   0(3,R4),=C'SIS'
         LA    R4,4(R4)
*
IF_FIX   TM    SMF64MC3,X'02'     CB FIXED IN REAL?
         BNO   IF_BUFF
         MVC   0(5,R4),=C'CBFIX'
         LA    R4,6(R4)
*
IF_BUFF  MVC   0(6,R4),BUFF31
         TM    SMF64MC3,X'01'     BUFFERS ABOVE?
         BO    IF_SMB
         MVC   0(6,R4),BUFF24
IF_SMB   LA    R4,7(,R4)
         TM    SMF64SMB,X'80'     SMB USED?
         BNO   PUTREC
         MVC   0(3,R4),=C'SMB'
         LA    R4,4(,R4)
         TM    SMF64SMB,X'02'     LOAD OPTIMIZED FOR SPEED?
         BNO   IF_CR
         MVC   4(2,R4),=C'CO'
         B     IF_DO
IF_CR    TM    SMF64SMB,X'01'     LOAD OPTIMIZED FOR RECOVERY?
         BNO   IF_CR
         MVC   4(2,R4),=C'CR'
IF_DO    LA    R4,3(,R4)
         TM    SMF64SMB,X'20'     DO BIAS USED?
         BNO   IF_SO
         MVC   4(2,R4),=C'DO'
         B     IF_VSP
IF_SO    TM    SMF64SMB,X'10'     SO BIAS USED?
         BNO   IF_SW
         MVC   4(2,R4),=C'SO'
         B     IF_VSP
IF_SW    TM    SMF64SMB,X'08'     SW BIAS USED?
         BNO   IF_DFW
         MVC   4(2,R4),=C'SW'
         B     IF_VSP
IF_DFW   TM    SMF64SMB,X'04'     DW BIAS USED?
         BNO   IF_VSP
         MVC   4(2,R4),=C'DW'
         B     TM_VSP
IF_VSP   LA    R4,3(,R4)
TM_VSP   TM    SMF64RSC,X'80'     AMOUNT OF VIRTUAL BUFFER INFORMED
         BNO   IF_HWT
         MVC   0(3,R4),VSP
         LA    R4,4(,R4)
IF_HWT   TM    SMF64RSC,X'40'     USED HIPERSPACE BUFFERS?
```

```
         BNO   IF_RE31
         MVC   0(3,R4),HWT
         LA    R4,4(,R4)
IF_RE31  TM    SMF64RSC,X'20'     RMODE31=BUFF USED
         BNO   IF_CB31
         MVC   0(7,R4),BUFF31
IF_CB31  TM    SMF64RSC,X'10'     RMODE31=CB   USED
         BNO   IF_878
         MVC   0(5,R4),CB31
IF_878   TM    SMF64RSC,X'10'     RMODE31=CB   USED
         BNO   PUTREC
         MVC   0(5,R4),IREG
*
PUTREC   LTR   R8,R8              TITLE?
         BNZ   GR_REC
         PUT   REC,TIT
         PUT   REC,CAB
         LA    R8,15
GR_REC   PUT   REC,RECORD
         PUT   REC,REGL2
         PUT   REC,REGL3
         MVI   R_ACCESS,C' '
         MVC   R_ACCESS+1(L'R_ACCESS-1),R_ACCESS
         BCT   R9,READ
         PUT   REC,TIT
         PUT   REC,CAB
         B     NUMLIN
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         SVC   3
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*---------------------------------------------------------------------*
*  EXECUTE INSTRUCTIONS
*---------------------------------------------------------------------*
MVDSN    MVC   RDSN(0),4(R5)
MVJOBN   MVC   RJOB(0),4(R5)
MVEXCP   MVC   T_EXCP(0),5(R5)
PACK     PACK  DOUBLE,5(0,R5)
         USING SMFRCD64,R10
CPJOB    CLC   RJOB(0),SMF64JBN
         DROP  R10
         USING SMFRCD64,R15
CPDSN    CLC   RDSN(0),SMF64CLN
         DROP  R15
TRT_PARM TRT   0(0,R5),TBVIRG
```

```
*-----------------------------------------------------------------------*
*  DCBS
*-----------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*-----------------------------------------------------------------------*
*  WORK AREA
*-----------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
LIM_EXCP DC    F'10000'
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
F3600    DC    F'3600'
F100     DC    F'100'
F60      DC    F'60'
FLAG     DC    X'00'
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM - ACCESS STATISTICS '
TITEXCP  DC    C'EXCP COUNT >= '
T_EXCP   DC    C'10000   '
         ORG   TIT+36
TIT_OPT  DS    CL54
         ORG   ,
CAB      DS    0CL134
         DC    134C' '
         ORG   CAB+1
         DC    C'JOBNAME'
         DS    CL2
         DC    C'DATA SET NAME'
         DS    CL32
         DC    C'DDNAME'
         DS    CL3
         DC    C'SYS'
         DS    CL2
         DC    C'TIMESTAMP'
         DS    CL6
         DC    C'MISCELLANEOUS: ACCESS, STATS SINCE LAST OPEN'
         ORG   ,
         DS    0F
RECORD DS      0CL134
         DC    134C' '
         ORG   RECORD+1
R_JOB    DS    CL8
         DS    CL1
R_DSN    DS    CL44
```

```
         DS    CL1
R_DDN    DS    CL8
         DS    CL1
R_SYSID  DS    CL4
         DS    CL1
R_STAMP  DS    0CL14
DATA     DS    CL5
         DS    CL1
HH       DS    CL2
         DC    C’:’
MM       DS    CL2
         DC    C’:’
SS       DS    CL2
         ORG   ,
REGL2    DS    0CL134
         DC    134C’ ’
         ORG   REGL2+1
         DC    C’EXCPS:’
EXCPS    DS    CL12
         DS    CL1
R_ACCESS DS    CL114
         ORG   ,
REGL3    DS    0CL134
         DC    134C’ ’
         ORG   REGL3+1
         DC    C’INSERTS:’
INSERT   DS    CL12
         DS    CL1
         DC    C’DELETES:’
DELETE   DS    CL12
         DS    CL1
         DC    C’UPDATES:’
UPDATE   DS    CL12
         DS    CL1
         DC    C’READ:’
READS    DS    CL12
         DS    CL1
         DC    C’CI-SPLIT:’
CI_SPLIT DS    CL12
         DS    CL1
         DC    C’CA-SPLIT:’
CA_SPLIT DS    CL12
         ORG   ,
MODEL    DC  X’40202020202020202020202120’
RDSN     DS    0CL44
         DC    44C’ ’
RJOB     DS    0CL8
         DC    8C’ ’
NXF      DC    CL4’N-XF’
```

```
         ORG   NXF+2
XF       DS    CL2
         ORG   ,
NXA      DC    CL4'N-XA'
         ORG   NXA+2
XA       DS    CL2
         ORG   ,
RLS      DC    C'RLS'
DATAJ    DS    XL4
         DS    0F
T_INIT   DS    X'00000000'
T_FIM    DS    X'00000000'
BUFF31   DC    C'BUFF31'
BUFF24   DC    C'BUFF24'
CB31     DC    C'CB31'
VSP      DC    C'VSP'
HWT      DC    C'HWT'
IREG     DC    C'IR'
TBVIRG   DC    256X'00'
         ORG   TBVIRG+C','
         DC    X'01'
         ORG   ,
*---------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*---------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMF64
//*-------------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSIN   DD  *
  ENTRY   SMF64
  NAME    SMF64(R)
/*
```

SMFLSR sample program

Since z/OS V1 R3, the algorithm to calculate the minimum index CI size changed. The assembler sample code in Example A-2 helps you identify the VSAM data sets opened in LSR mode. With the change in calculation, you can face problems when:

- Under CICS, if you are explicitly coding the number of buffers of each size in the LSRPOOL, two things can happen:

– The amount of buffers for the new size is not enough leading to performance problems
– Buffers for the new index CI size do not exist causing the open to fail. The buffer size must be the same size of the index CI size or larger.

- Under other applications, when the CI index size buffer is defined in BLDVRP and is lower than the new index CI size calculation.

Example: A-2

```
//*-------------------------------------------------------------------*
//*  JCL:                                                             *
//*  //RELAT   EXEC PGM=SMFLSR                                        *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                          *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                          *
//*  //REPORT   DD SYSOUT=*,LRECL=133                                 *
//*                                                                   *
//* REPORT DESCRIPTION:                                               *
//* SYSID JOBNAME DDNAME DATASET NAME CI-SIZE KEY-LENGTH               *
//*                                                                   *
//* RETURN CODES:                                                     *
//*    0 - FOUND DATA SETS IN LSR MODE                                *
//*    4 - NO FOUND                                                   *
//*===================================================================*
//*                                                                   *
//*  JCL: DUMP  SMF 64 RECORD FROM SYS1.MAN                           *
//*                                                                   *
//*  //STEP1     EXEC  PGM=IFASMFDP                                   *
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????                       *
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),           *
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA                *
//*  //SYSPRINT  DD    SYSOUT=A                                       *
//*  //SYSIN     DD  *                                                *
//*    INDD(INDD,OPTIONS(DUMP))                                       *
//*    OUTDD(DUMPOUT,TYPE(64:64))                                     *
//*                                                                   *
//*-------------------------------------------------------------------*
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - VSAM OPEN FOR LSR MODE'
SMFLSR CSECT
SMFLSR AMODE 31
SMFLSR RMODE 24
         STM   R14,R12,12(R13)     * LINKAGE CONVENTION
         LR    R12,R15             * R12 BASE
         USING SMFLSR,R12
         LA    R15,SAVE            * END SAVE DESTE PGM
         ST    R15,8(R13)
         ST    R13,SAVE+4          * SALVA END SAVE ANTERIOR
         LR    R13,R15
         LA    R6,4                * RC NOT FOUND
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
SET_TIT  LA    R8,0                * TITLE CONTROL
         LA    R9,66               * 66 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1              * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64         * 64?
         BNE   READ
         CLI   2(R10),X'00'        * DISCARD SPANNED RECORDS
         BNE   READ
         TM    SMF64RIN,X'E0'      * CLOSE,VOL SWITCH,OUT OF SPACE
         BZ    READ                * NO. DISCARD
         TM    SMF64DTY,X'40'      * INDEX?
         BNO   READ                * NO.DISCARD
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         DROP  R10
         USING SMFRCD64,R15
         TM    SMF64MC3,X'40'     LSR?
         BNO   READ               NO, DISCARD
         TM    SMF64SMB,X'80'     SMB USED?
         BO    READ               YES. DISCARD. SMB BUILDS BUFFER POOL
         DROP  R15                ACCORDING TO CISIZE
*
         USING SMFRCD64,R10        SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID    SYSID
         MVC   R_JOB,SMF64JBN      JOBNAME
*
         DROP  R10
         USING SMFRCD64,R15
*
         MVC   R_DDN,SMF64DDN      * DDNAME
         MVC   R_DSN,SMF64CLN      * CLUSTER NAME
         L     R3,SMF64DCI         * CI-SIZE
         CVD   R3,DOUBLE
         MVC   R_CI,MODEL
         ED    R_CI,DOUBLE+4
         LA    R3,0
         ICM   R3,B'0011',SMF64DKL         * KEY-SIZE
         CVD   R3,DOUBLE
         MVC   R_KEY,MODEL+4
         LTR   R8,R8             TITLE?
         BNZ   GR_REC
         PUT   REC,TIT
         PUT   REC,CAB
         LA    R8,15
GR_REC   PUT   REC,RECORD
         LA    R6,0
         BCT   R9,READ
         B     SET_TIT
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         LR    R15,R6            RETURN CODE
         L     R13,SAVE+4        LINKAGE CONVENTION
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
*
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*----------------------------------------------------------------------*
*  DCBS
*----------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*----------------------------------------------------------------------*
*  WORK AREA
*----------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM - DATA SET WITH LSR ACCESS'
         ORG   ,
CAB      DS    0CL134
         DC    134C' '
         ORG   CAB+1
         DC    C'SYSTEM '
         DS    CL2
         DC    C'JOBNAME'
         DS    CL3
         DC    C'DDNAME'
         DS    CL4
         DC    C'DATA SET NAME'
         DS    CL32
         DC    C'KEY SIZE'
         DS    CL2
         DC    C'INDEX CI SIZE'
         ORG   ,
         DS    0F
RECORD DS      0CL134
         DC    134C' '
         ORG   RECORD+1
R_SYSID  DS    CL4
         DS    CL5
R_JOB    DS    CL8
         DS    CL2
R_DDN    DS    CL8
         DS    CL2
R_DSN    DS    CL44
         DS    CL1
R_KEY    DS    CL6
         DS    CL4
R_CI     DS    CL6
         ORG   ,
MODEL    DC  X'402020202120'
*---------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*---------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMFLSR
//*-------------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_DATA_SET
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_DATA_SET
//L.SYSIN   DD  *
  ENTRY   SMFLSR
  NAME    SMFLSR(R)
/*
```

REXX code to list compression ratio

Compression is useful when the compression rate is high, to compensate for the CPU used to perform the compress. The compression rate depends on the data.

You can use the REXX in Example A-4 to list the compress ratio for:

- All data sets (VSAM KSDS or non-VSAM) from a catalog.
- An specific data set

You can execute the REXX via:

- TSO. or
- Batch, using the JCL shown in Example A-3

Example: A-3 Sample JCL for running REXX in batch

```
//REXX      EXEC  PGM=IRXJCL,
//  PARM='REXX_NAME +UCAT.VSBOX01'
//SYSTSPRT  DD  SYSOUT=*
//SYSEXEC   DD DISP=SHR,DSN=PDS_REXX
//SYSTSIN   DD DUMMY
```

Example: A-4

```
/* REXX */
/*TRACE R  */
 /*****************************************************************/
 /* FUNCTION:                                                     */
 /* DISPLAY THE COMPRESSION RATIO FOR:                            */
 /*  1 -  DATA SET: EXEC REXX_NAME 'DATA_SET_NAME'                */
 /*  OR, WHEN THE PARM IS PREFIXED BY "+":                        */
 /*  2 -  FOR ALL COMPRESSED DATA SETS FROM A CATALOG:            */
 /*      EXEC REXX_NAME '+CATALOG_NAME'                           */
 /*****************************************************************/
 PARSE UPPER ARG DSN
 IF DSN = '' THEN
    DO
    SAY "NO DSN OR CATALOG INFORMED. "
    EXIT
    END
 PARSE VAR DSN '+' CAT
 IF CAT /= '' THEN  DSN = '**'
    ELSE CAT = ' '
MODRSNRC = SUBSTR(' ',1,4)
CSIFILTK = SUBSTR(DSN,1,44)
CSICATNM = SUBSTR(CAT,1,44)
CSIRESNM = SUBSTR(' ',1,44)
CSIDTYPS = SUBSTR(' ',1,16)
/*  OPTIONS */
CSICLDI  = SUBSTR(' ',1,1)
CSIRESUM = SUBSTR(' ',1,1)          /*   RESUME FLAG                 */
CSIS1CAT = SUBSTR('Y',1,1)          /*   SEARCH MORE THAN 1 CATALOG  */
CSIRESRV = SUBSTR(' ',1,1)
```

```
CSINUMEN = '0003'X

CSIFLD1  = SUBSTR('COMPIND COMUDSIZUDATASIZ',1,24)

CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIRESRV
CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1
WORKLEN = 1024
DWORK = '00000400'X || COPIES('00'X,WORKLEN-4)
FOUND = 0
NUMERIC DIGITS 16

RESUME = 'Y'

DO WHILE RESUME = 'Y'
   ADDRESS LINKPGM 'IGGCSI00  MODRSNRC  CSIFIELD  DWORK'
   LCC = RC
   IF LCC ¬= 0 THEN
      DO
      REASON = C2X(MODRSNRC)
      SAY 'IGGCSI00 NOT OK RC:' LCC 'REASON:' REASON
      EXIT
      END
   RESUME = SUBSTR(CSIFIELD,150,1)  /* RESUME FLAG LOOP CTL */
   USEDLEN = C2D(SUBSTR(DWORK,9,4)) /* USED WORK AREA LENGTH  */
   POS1=15                          /* AREA INFO ENTRY  */
   /*****************************************************************/
   /*  PROCESS WORK AREA DATA                                       */
   /*****************************************************************/
   DO WHILE POS1 < USEDLEN            /* PROCESS  WORK AREA */
      IF SUBSTR(DWORK,POS1+1,1) = '0' THEN POS1 = POS1 + 50
      ELSE
         DO
         CSIEFLAG = SUBSTR(DWORK,POS1,1)
         X = BITAND(CSIEFLAG,'20'X)            /*  VALID? */
         IF X /= '20'X THEN  POS1 = POS1 + 50
         ELSE
             DO
             X = BITAND(SUBSTR(DWORK,POS1+56,1),'60'X)
             TOTDATA = C2D(SUBSTR(DWORK,POS1+46,2))
             IF X = '60'X THEN
                DO
                X =  BITAND(SUBSTR(DWORK,POS1+57,1),'FF'X)
                IF X /= 'FF'X  THEN
                  DO
                  FOUND = 1
                  DSN = SPACE(SUBSTR(DWORK,POS1+2,44),0)
                  STLEN = C2D(SUBSTR(DWORK,POS1+50,2))
                  NDLEN = C2D(SUBSTR(DWORK,POS1+52,2))
                  RDLEN = C2D(SUBSTR(DWORK,POS1+54,2))
                  USIZE = C2D(SUBSTR(DWORK,POS1+65,RDLEN))
                  XPRESS = C2D(SUBSTR(DWORK,POS1+57,NDLEN))
                  RATIO = SPACE(FORMAT((1-(XPRESS/USIZE))*100,16,2),0)
                  SAY "COMPRESSION RATIO OF  " DSN ": " RATIO "%"
                  END
                END
             POS1 = POS1 + TOTDATA + 46
             END
         END
   END
ENDIF FOUND = 0 THEN SAY "NO COMPRESSED DATA SETS FOUND "
```

SMFRLS Sample program

(*)

Example: A-5

```
//*-------------------------------------------------------------------*
//*  FUNCTION:                                                        *
//*  RLS CF STRUCTURE AND I/O STATISTICS                              *
//*                                                                   *
//*-------------------------------------------------------------------*
//* REPORT DESCRIPTION                                                *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM THE LOCAL         *
//*        SHARED BUFFER POOL                                         *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM DE CF CACHE       *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM DASD              *
//*      # OF UPDATES, INSERTS, DELETES, READS, SPLITS CI AND CA      *
//* SYSTEM ID, JOBNAME, DATE AND TIME WHEN JOB STARTED, DDNAME AND    *
//* DATA SET NAME                                                     *
//*-------------------------------------------------------------------*
//*  JCL:                                                             *
//*  //RELAT   EXEC PGM=SMFRLS                                        *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                          *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                          *
//*  //REPORT   DD SYSOUT=*,LRECL=133                                 *
//*                                                                   *
//*  PARM: NO PARM USED                                               *
//*  ----------------------------------------------------------       *
//*  JCL DUMP  SMF 64 RECORD FROM SYS1.MAN                            *
//*  //STEP1     EXEC  PGM=IFASMFDP
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA
//*  //SYSPRINT  DD    SYSOUT=A
//*  //SYSIN     DD  *
//*    INDD(INDD,OPTIONS(DUMP))
//*    OUTDD(DUMPOUT,TYPE(64:64))                                      *
//*--------------------------------------------------------------------*
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - RLS HITS STATS'
SMFRLS CSECT
SMFRLS AMODE 31
SMFRLS RMODE 24
         STM   R14,R12,12(R13)      * LINKAGE CONVENTION
         LR    R12,R15              * 12 BASE REGISTER
         USING SMFRLS,R12
         LA    R15,SAVE
         ST    R15,8(R13)
         ST    R13,SAVE+4
         LR    R13,R15
*
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
SET_TIT  LA    R8,0                 * TITLE CONTROL
         LA    R9,14                * 60 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1               * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64          * 64?
         BNE   READ
         CLI   2(R10),X'00'
         BNE   READ
         TM    SMF64RIN,X'80'       * COMPONENT CLOSE?
         BZ    READ
         TM    SMF64DTY,X'80'       * DATA SET?
         BNO   READ
         TM    SMF64DTY,X'08'       RECORD LEVEL SHARED?
         BNO   READ
         USING SMFRCD64,R10         SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID     SYSID
         MVC   R_JOB,SMF64JBN       JOBNAME
         MVC   R_DSN,SMF64DNM       * COMPONENT NAME
MV_DATA  UNPK  DATE,SMF64RSD          JOB START DATE (JULIAN)
MV_TME   ICM   R3,B'1111',SMF64RST    TIME HUNDREDTHS OF SECONDS
         L     R1,F100                SINCE MIDNIGHT
         LA    R2,0
         DR    R2,R1               * SECONDS
         L     R1,F3600            * SECONDS IN  HOURS
         LA    R2,0
         DR    R2,R1             * HOUR EM R3
         CVD   R3,DOUBLE
         UNPK  HH,DOUBLE+6(2)
         OI    HH+1,X'F0'
         L     R1,F60
         LR    R3,R2              * REMAINING
         LA    R2,0
         DR    R2,R1              * MINUTS
         CVD   R3,DOUBLE
         UNPK  MM,DOUBLE+6(2)
         OI    MM+1,X'F0'
         CVD   R2,DOUBLE          * REMAINING SECOUNDS
         UNPK  SS,DOUBLE+6(2)
         OI    SS+1,X'F0'
*
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         DROP  R10
         USING SMFRCD64,R15
         MVC   R_DDN,SMF64DDN     * DDNAME
         L     R3,SMF64DIN        * INSERTIONS
         CVD   R3,DOUBLE
         MVC   INSERT,MODEL
         ED    INSERT,DOUBLE+2
         L     R3,SMF64DDE        * RECORDS DELETED
         CVD   R3,DOUBLE
         MVC   DELETE,MODEL
         ED    DELETE,DOUBLE+2
         L     R3,SMF64DUP        * RECORDS UPDATED
         CVD   R3,DOUBLE
         MVC   UPDATE,MODEL
         ED    UPDATE,DOUBLE+2
         L     R3,SMF64DRE        * RECORDS RETRIEVED
         CVD   R3,DOUBLE
         MVC   READS,MODEL
         ED    READS,DOUBLE+2
         L     R3,SMF64DCS        * CONTROL INTERVAL SPLITS
         CVD   R3,DOUBLE
         MVC   CI_SPLIT,MODEL2
         ED    CI_SPLIT,DOUBLE+4
         L     R3,SMF64DAS        * CONTROL AREA SPLITS
         CVD   R3,DOUBLE
         MVC   CA_SPLIT,MODEL2
         ED    CA_SPLIT,DOUBLE+4
         L     R3,SMF64BMH        * HIT LOCAL SHARED BUFFER POOL
         CVD   R3,DOUBLE
         MVC   HITLBP,MODEL
         ED    HITLBP,DOUBLE+2
         L     R3,SMF64CFH         * HIT CF CACHE STRUCTURE
         CVD   R3,DOUBLE
         MVC   HITCF,MODEL
         ED    HITCF,DOUBLE+2
         L     R3,SMF64RIO         * DATA RECORDS FROM DASD
         CVD   R3,DOUBLE
         MVC   DASD,MODEL
         ED    DASD,DOUBLE+2
*
PUTREC   LTR   R8,R8              TITLE?
         BNZ   GR_REC
         LA    R8,1
         PUT   REC,TIT
         PUT   REC,REC_SPC
         PUT   REC,HEADER
         PUT   REC,REC_SPC
         LA    R15,0
         ST    R15,RC
GR_REC   PUT   REC,REC_JOB
         PUT   REC,REC_RLS
         PUT   REC,REC_IO
         PUT   REC,REC_SPC
         BCT   R9,READ
         B     SET_TIT
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         L     R13,SAVE+4
         L     R14,12(R13)
         L     R15,RC
         LM    R0,R12,20(R13)
         BR    R14
*
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*-------------------------------------------------------------------*
*  EXECUTE INSTRUCTIONS
*-------------------------------------------------------------------*
PACK     PACK  DOUBLE,5(0,R5)
*-------------------------------------------------------------------*
*  DCBS
*-------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*-------------------------------------------------------------------*
*  WORK AREA
```

```
*----------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
F3600    DC    F'3600'
F100     DC    F'100'
F60      DC    F'60'
RC       DC    F'4'
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM RLS - CF STRUCTURE STATISTICS '
         DC    C'SINCE LAST OPEN'
         ORG   ,
HEADER   DC    134C' '
         ORG   HEADER+1
         DC    C'SYSID'
         DS    CL1
         DC    C'JOBNAME'
         DS    CL2
         DC    C'START DT/TIME'
         DS    CL2
         DC    C'DDNAME'
         DS    CL3
         DC    C'DATA SET NAME'
         ORG   ,
         DS    0F
REC_JOB  DS    0CL134
         DC    134C' '
         ORG   REC_JOB+1
R_SYSID  DS    CL4                 1
         DS    CL2                 5
R_JOB    DS    CL8                 7
         DS    CL1                 15
DATE     DS    CL5                 16
         DS    CL1                 21
HH       DS    CL2                 22
         DC    C':'                24
MM       DS    CL2                 25
         DC    C':'                27
SS       DS    CL2                 28
         DS    CL1                 30
R_DDN    DS    CL8                 31
         DS    CL1                 39
R_DSN    DS    CL44                36
         ORG   ,
REC_RLS  DC    134C' '
```

```
         ORG   REC_RLS+1
         DC    C'LOCAL BP HIT:'   1
HITLBP   DS    CL12              14
         DS    CL1               26
         DC    C'CF CACHE HIT:'  27
HITCF    DS    CL12              40
         DS    CL1               52
         DC    C'I/O DASD:'      53
DASD     DS    CL12              62
         ORG   ,
REC_IO   DS    0CL134
         DC    134C' '
         ORG   REC_IO+1
         DC    C'INS:'          23
INSERT   DS    CL12             27
         DS    CL1              39
         DC    C'DEL:'          40
DELETE   DS    CL12             44
         DS    CL1              56
         DC    C'UPDT:'         57
UPDATE   DS    CL12             62
         DS    CL1              74
         DC    C'READ:'         75
READS    DS    CL12             80
         DS    CL1              92
         DC    C'CI-SPLT:'      93
CI_SPLIT DS    CL6             101
         DS    CL1             107
         DC    C'CA-SPLT:'     108
CA_SPLIT DS    CL6             116
         ORG   ,
REC_SPC  DC 134C' '
MODEL    DC  X'4020202020202020202020202120'
MODEL2   DC  X'402020202120'
*-----------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*-----------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMFRLS
//*--------------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSIN   DD  *
  ENTRY   SMFRLS
  NAME    SMFRLS(R)
/*
```

GTF procedure example

Example: A-6   GTF Procedure example

```
//*-------------------------------------------------------------------*
//*  TRACE START: UNDER SDSF  ==>    /S GTFIO
//*  REPLY:
//*    NN AHL125A  RESPECIFY TRACE OPTIONS OR REPLY U
//*    TO USE OPTIONS         ==>     NN,U
//*    TO CHANGE OPTIONS ==> NN,TRACE=SSCHP,IOP,PCI,CCWP,JOBNAMEP
//*    ANSWER THE REPLY:
//*    NN AHL101A  SPECIFY TRACE EVENT KEYWORDS SSCHP IOP CCWP,JOBNAMEP
//*    WITH ==>
//*    NN,SSCH=(ADRS),CCW=(SI,CCWN=512,DATA=16,PCITAB=5),JOBNAME=JJJJ
//*          ADRS; DEVICES ADDRESS TILL 128: ADDI-ADDF
//*    NEW REPLY:  RR AHL102A  CONTINUE TRACE DEFINITION OR REPLY END
//*    ANSWER  ==> RR,END
//*    AGAIN REPLY  MSG AHL125A,       ---------> N,U
//*    TO STOP THE TRACE, COMMAND      --------->  /S NNNNN
//*    NNNNN IS THE STEPNAME SHOWN IN "DA" SDSF OPTION
//*==================================================================
//*  MEMBER MHL CONTAINS THE GTF TRACE OPTIONS:
//*  TRACE=SSCHP,IOP,PCI,CCWP,JOBNAMEP
//*==================================================================
//GTFIO   PROC MEMBER=GTFVSAM,M=MHL
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
//    TIME=1440,REGION=6M
//IEFRDER DD   DSNAME=SYS1.&M..TRACE,UNIT=SYSDA,SPACE=(CYL,(150)),
//        DISP=(NEW,CATLG)
//SYSLIB  DD   DSNAME=MHLRES2.TESTS.JCL(&MEMBER),DISP=SHR
```

B

Miscellaneous performance items

In this appendix, we describe the lab environment used for our measurements throughout the book, and have a general discussion on caching, and common error messages indicating broken data sets and output from the EXAMINE command.

Our test environment

We ran a few experiments to clarify some performance and usability aspects of VSAM. However, it happened in a non-controlled environment, where we cannot guarantee the same level of multiprogramming, and the same load in our DASD controller. Nevertheless, the results are sound, if you take into consideration such variables as number of EXCPs, I/O Connect time, and CPU time to compare the runs.

The jobs are totally I/O bound due to *read-and-forget* and *write-from-thin-air*.

Hardware configuration

The CEC is a 2064 z900-1C7. Our three LPs (SC63, SC64, SC65) have 1.5 GB each of central storage and are located in the same CEC.

The two CF images have 1.0 GB each of central storage, also located in the above referenced in CECCF. The links to the CFs and between the CFs (for SM duplexing) are all IC peer-to-peer links. There are 2 shared links between each z/OS and each CF, and f links between the CFs (for duplexing).

You have 8 FICON paths to an ESS model 800 controller (not turbo) with a small I/O load.

Software configuration

All tests were done under z/OS V1 R4.

General lab description

The data for our tests were generated using IEBDG utility. The JCL and commands used are shown in Example B-1. We generated 2M records to create a master KSDS cluster, with random binary key. The data was the same for all tests, except for those tests compared with the data of first edition of this book.

The type of data and the number of records are not the same as that used in the first edition of this redbook. We increased the number of records and types of data due to changes in CPU velocity, data rate (like FICON) and enhancements in ESS (SHARK).

Example: B-1 JCL for data test

```
//CREATE    EXEC PGM=IEBDG
//SYSPRINT  DD SYSOUT=*
//TESTOUT   DD DSN=MHLRES2.DATA.VSAM,DISP=(,CATLG),UNIT=SYSDA,
// LRECL=300,RECFM=FB,SPACE=(CYL,(500,500),RLSE),
// VOL=SER=(DJL001,DJL002)
//SYSIN     DD *
  DSD OUTPUT=(TESTOUT)
  FD  NAME=KEY1,LENGTH=4,FORMAT=RA
  FD  NAME=KEY2,LENGTH=4,FORMAT=RA
  FD  NAME=DATA1,LENGTH=30,FORMAT=AL,ACTION=RP
  FD  NAME=DATA2,LENGTH=30,PICTURE=15,P'941246876976215',ACTION=RP
  FD  NAME=DATA3,LENGTH=100,FORMAT=AN
  FD  NAME=DATA4,LENGTH=100,FORMAT=CO
  FD  NAME=DATA5,LENGTH=32,FORMAT=RA
  CREATE QUANTITY=2000000,NAME=(KEY1,KEY2,DATA1,DATA2,DATA3,DATA4,DATA5)
```

The sequencial access tests were done using an assembler program that opens the VSAM data set and reads it all. No others processing are done with the records.

The direct access tests were done using an assembler program. The kind of tests with direct access was two:

- Read direct: Read a sequencial file with 200,000 keys to read in direct access in the VSAM data set. The keys in the sequencial file are *not* in key sequence.
- Inserts: Read a sequencial file with 200,000 keys to read in direct access in the VSAM data set. If the keys does not exits, inserts the new key in the VSAM data set. The keys in the sequencial file are *not* in key sequence and 100,000 are insertions to the VSAM data set.

The load of the data set was done using IDCAMS.

The jobs access a KSDS master cluster with the following characteristics:

- LRECL = 300 bytes
- Key length = 8 bytes
- Number of records = 2,000,000 (not uniform key values distribution)
- Free Space = (10,20)
- Data component CI size 4096
- Index component CI size chosen by VSAM

There is also a physical sequential file where the 300-bytes records represents inclusions or reads in the master, with *no* repeated keys.

- DIRECT GET: 199,998 records, 10%

- GET SEQUENTIAL: Reads sequentially all records of the VSAM data set (2,000,000 records), with minimum record processing.
- INSERTIONS: 99,999 records (5%). Splits after insertions:
  - Data component:
    - Control interval: 20,810
    - Control Area: 0
  - Index component:
    - Control interval: 0
    - Control area: 0

What do we measure?

To compare the performance results among the runs, we select the following variables:

- Total I/O connect time

  Connect time is when the channel is transferring data from or to the cache. As we keep in all the experiences, always the same volume located in the same controller and accessed by the same channel type (so, always the same data rate), the connect time is an indirect measurement about how many bytes were transferred along the I/O operations. Then, the only way of decreasing the total I/O connect time is by moving less data to or from storage.

  Variations in the load of the controller should not affect this value.

  The total connect time was captured with GTF.

- Total I/O Disconnect time

  Disconnect time occurs when the channel is not doing activities related to the execution of the channel program along the I/O operation. It means that the target record for a read is not in the cache and the disk access is a must. Then, the only ways of decreasing the total I/O disconnect time is by moving less data to or from storage or improving the use of the cache

  Variations in the load of the controller should affect this value.

  The total disconnect time was captured with GTF.

- Number of EXCPs

  In SMF account information for SAM data sets, one EXCP equates one physical block transfer.

  For VSAM data sets one EXCP equates to one real I/O operation for data or index. It is not true, that the number of EXCPs for VSAM measures the number of transferred CIs. One EXCP in may mean more than one CI being transferred. The number of CIs per I/O depends very much in the number of buffers and in the buffering technique. Consequently, the number of EXCPs is not repetitive, that is, the same job reading the same data set may present a different number of EXCPs. The reasons justifying the number of I/O operations instead of CIs are:

  - We can measure the VSAM capacity in saving I/O operations and consequently saving TCB and SRB time.
  - The number of I/O transferred CIs can be cached from LISTC in the catalog or from SMF record 64

  In our measurements, the number of EXCPs corresponds to the number of I/O operations.Variations in the load of the controller should not affect this value.

- Elapsed time

  Elapsed time is the wall clock time to run the job used in the test. If the I/O is faster, in I/O bound job the elapsed time should be less.

  Variations in the load of the controller and the system should affect this value.

- Total CPU time (TCB and SRB)

  All the runs are totally I/O bound, meaning no processing at all. Then the TCB time can be charged to the preparation of the I/O operation, including VSAM buffer pool management. SRB time here is spent along I/O interrupts processing only.

  Variations in the load of the controller and system should affect very mildly this value.

RLS experiments

Refer to “Batch RLS performance experiments and comparison” on page 300.

DASD cache concepts

A cache is a fast storage (no mechanical movement) located in the DASD controller with two functions: to minimize access to disks (by having hits) and to serve as a *speed matching buffer* to synchronize elements with different speeds as channels and disks in a cache miss. In this appendix the word disk does not have the same meaning of DASD. Disk implies RAID media (FBA, SSA or SCSI) used by modern controllers. DASD still means the logical 3390/3380.

To have random hits (saving disk access) for reads and writes, the I/O workload must revisit its data, however in certain cases it does not happen. In this case such workloads are called *cache unfriendly*. Usually, we may have two types of hits for VSAM data, when the application revisits:

- Exactly the same logical record in a CI already in cache
- The same data CI (already in cache) because other logical record (a sort of lucky)

For sequential access, it is important to say that, cache does not save data CI disks I/O operations. The cache only tries to match the speed of the disks and channels.

With the new controllers a cache miss implies accessing the RAID disks and not the logical 3390/3380 device (which do not exist physically). Because the mapping between the 3390/3380 tracks to the FBA RAID disks is not externalized, it is not important anymore the relative location of files in order to avoid long Seeks or even RPS misses.

It is interesting to note that heavy cache access reduces DASD skew (unequal 3390/3380 device utilization) because data formerly in heavily used devices now became electronically accessible due to the LRU algorithm. Refer th the DASD Activity report in Figure B-1.

```
 I=92%  DEV            ACTV RESP IOSQ ---DELAY--- PEND DISC CONN
VOLSER NUM   MX  LCU   RATE TIME TIME DPB CUB DB  TIME TIME TIME
TOTTSJ 256C      005A 0.024   11    0 0.0 0.0 0.0  0.2  8.2  2.2
TOTJS1 250C      0059 5.764    6    0 0.0 0.0 1.2  1.5  0.1  4.7
TSMS50 650D   4  0144 139.6    1    0 0.0 0.0 0.0  0.3  0.0  0.7
```

Figure B-1 DASD Activity Report

Increasing the I/O rate, decreases the disconnect time (less the disconnect more hits in cache) due to the LRU algorithm is keeping in cache the most used elements.

There are two types of cache: volatile and non-volatile (NVS). NVS is used to keep DFW, dual copy and XRC remote copy records. In this chapter we use the word cache meaning the volatile cache and NVS for the non-volatile.

The cache is made of CMOS DRAM storage for Data and Directory. The directory entries describes the data elements (4 KB in ESS) and are ordered in the following queues:

- LRU (pointing to active data elements)
- Free (available for staging from disk or writes)
- Pinned (changed data in cache and the respective disk is not available)
- Defective

Destage is the movement from cache to disk caused by previous DFW and CFW writes (to be covered later) and it is asynchronous with the I/O operation which executed the writes. By the way, demotion is not a synonym of destaging. Demotion means to take a data element out from cache. If there is a valid copy in disk there is not a destage. If not, a destage is done. There are three types of destaging:

- To relieve contention, when NVS or Cache become full. Controlled by modified LRU algorithms, when the percentage of NVS occupancy is greater than X%. If greater than Y% (Y>X), then the DFW I/O operation bypass the NVS cache going synchronously from volatile cache to disk (called DFW bypass)
- Done in background by the controller when the percentage of NVS occupancy is above another threshold Z% (Z < X)
- Forced by Z EOD command or by an hardware error

The amount of data destaged is usually more than one 3390/3380 track. A smart controller can Identify other records changed in adjacent tracks of the record to be destaged and destage all of them. It saves rotational delays in the disks and bypass the RAID write penalty.

Stage is the movement from DASD to cache, it can be synchronous (a read miss) for a random request or asynchronous for a sequential look ahead read. Pay attention that random and direct have the same meaning.

A cache hit may overlap with staging/destaging operations, for the same 3390/3380 logical device. It is possible to have concurrent access to same 3390/3380 device data but not in the same track, for example, one hit in the cache and an asynchronous destage in the disks back storage (not in the same 3390/3380 track). This is not the ESS PAV feature, because here there is just one active I/O operation, that is the one with the hits in the cache, the destage is just controller housekeeping.

Controller accounts data about number of I/O operations, hits, misses, destages, in a volume or in a data set basis. These values are shown by IDCAMS LIISTDATA command, by RMF Cache report and used in SMS for Dynamic Caching for data sets.

Set Subsystem CCW activates the use of caching in the controller (subsystem) and in individual volumes. This CCW allows cache modes as *normal, DFW* and CFW. All these modes may be asked explicitly by software through the Define Extent CCW (per each I/O operation), in some cases the controller can adaptively change the mode due to the observed pattern of access. Following is the description of such CCW.

Define Extent CCW provides a 16 bytes parameters, which define limits on subsequent operations (extent information, avoidance of writes, for example), provide a blocksize value, and specify caching control mode (or hints) for the channel program. The caching mode have the granularity of I/O operation. These modes are not totally mutual exclusive, and the same I/O operation may have more than one mode. The modes are:

- Track Level Cache (TLC)
- Record Level Cache (RLC)
- Sequential:
  - Reads:
    - Sequential Access
    - Sequential Pre-stage
  - Writes
- Least Recently Used (LRU)
- Normal Caching
- CFW
- DFW
  - With the possibility of Quick Writes
- Bypass Cache
- Inhibit Cache Loading

Cache Modes

An explanation of these modes is:

- Track Level Cache (TLC)

  In TLC the unit of transport between cache and disks are the 3390/3380 tracks. TLC is more adequate to sequential. When in TLC mode the 3390/3380 track is staged in cache in one pass (if first miss is in record zero (R0)) or in two pass (if first miss is not in R0). RVA only has track level cache because the compact/compress data cause little traffic when moving logical 3390/3380 tracks.

- Record Level Cache (RLC)

  In RLC the unit is the referred logical 3390/3380 physical record. RLC is very adequate to random (also called direct) processing.

  Let us explain what do we mean by “logical 3390/3380 physical record”. The word *logical* indicates that the 3390/3380 does not real exist. The word *physical* indicates that we are talking about the physical record (the block) in the logical 3390/3380 track...

RLC improves performance for applications that do not exhibit good locality of reference and where the cost of track caching out weights the benefits. RLC reduces the costs associated with cache miss I/Os by staging less data into cache (less cache pollution), which frees the volume and other activity more quickly. Reading less data into cache also enables data to stay in cache longer, which increases the chances of future cache hits.

ESS controller is able to switch from one to the other depending on the access pattern, independently of the Define Extent CCW.

RLC is mutually exclusive from TLC

RLC can be activated for VSAM SMS managed maybe-cache data sets. SMS via DCME decides in Define Extent when to use RLC for reads. Refer to , “Using cache in an SMS data set” on page 409. In addition to deciding to enable or to disable the cache for maybe-cache SMS data sets, SMS picks up between RLC or TLC for reads.

- Sequential

  When in sequential mode, the controller uses the cache just as a speed matching buffer, to synchronize different speeds. There are two types.

  In *sequential read mode*, the controller does the pre-staging of a certain number of future referenced logical 3390/3380 tracks. After used, the tracks are or not demoted from cache depending on the sub mode:

  – *Sequential Access*: The used data is a strong candidate to be demoted

  – *Sequential Pre-stage*: The used data is protected by the LRU algorithm.

  Sequential read can be activated:

  – Explicitly by software through Define Extent CCW (also called sequential hint), as declared by VSAM ESDS for Sequential Access. KSDS/VRRDS in certain conditions declare Sequential Pre-stage.

  – By sequential detect, where the controller detects sequential access (six sequential referred logical 3390/3380 physical tracks in the ESS).

    Because KSDS/VRRDS VSAM organizations (in certain conditions) do not use the Define Extent. Then, in this case, it is interesting to avoid CI /CA splits in data sets usually processed sequentially. Splits make the logical sequence different from the physical sequence, and the controllers only detect the physical sequence pattern.

- Least Recently Used (LRU)

  LRU is not properly a mode, but a technique to maintain in the cache the most referenced elements. It is the major algorithm to control cache demoting, admitting that, if an element was referenced in the past, it is going to be again in the future. LRU is automatically set off when sequential caching, inhibit cache load and bypass cache modes are active

- Normal Caching

  In this mode the NVS cache is not used. There are four cases to consider in this mode:

  – For a read hit, there is a data transfer from cache to channel followed by a channel end (CE)/ device end (DE) I/O interrupt. The directory entry is LRU update (meaning that the data is to be kept in cache for a while).

  – For a read miss, the channel is disconnected, the disk is accessed to stage data to cache (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy). After that, the channel is reconnected and the data is transferred to channel from cache, CE/DE I/O interrupt, stage rest of track (if in track mode), LRU update.

    If all the serving channels are busy, there is not an RPS miss, because the data is already in the cache. With the new controllers, RPS misses only occur when the internal path to disks are busy and the disk needs a new revolution.

  – For a write hit, because the NVS cache is not used, it is like a miss. The channel moved data to volatile cache and disconnects. The disk is accessed synchronously to write data (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy).

    The word *synchronously* means that the write to disk is done without the end of the I/O operation be posted to the application.

    After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. LRU update (meaning that the data is to be kept in cache for a while)

  – For a write miss the channel move data to cache and disconnects. The disk is accessed synchronously to write data (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy).

    The word *synchronously* means that the write to disk is done without the end of the I/O operation be posted to the application.

    After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. No LRU update, meaning that the cache copy if the data is ready to be demoted

- Cache Fast Write (CFW)

  Is used for temporary data sets and consequently does not use the NVS for writes. It must be allowed by the Set Subsystem CCW issued by IDCAMS.

  – For reads and write misses is identical to normal caching.

  – For a write hit, the data is transfer from the channel to the cache followed by a CE/DE I/O interrupt and the directory entry is LRU update (meaning that the data is to be kept in cache for a while). Later on asynchronously the data will be destaged to disks.

Used by Sort for temporary files and for creating PDSE members (before the Stow) when the Hiperspace is full. The exploiter must declare CFW in the Define Extent CCW.

- DASD Fast Write (DFW)

  DFW allows the use of the NVS cache for writes. It avoids accessing disks for a write hit. It must be allowed by the Set Subsystem CCW issued by IDCAMS.

  – For reads is identical to Normal Caching.

  – For a write hit, the data is transfer from the channel to the cache and NVS, followed by a CE/DE I/O interrupt and the directory entry is LRU update (meaning that the data is to be kept in cache for a while). Some modern controllers as ESS sends the CE/DE earlier with one copy of the data in the NVS (other copy in the channel adapter buffer) doing the copy to the volatile cache immediately after the CE/DE.

    Later on and asynchronously the data will be destaged to disks. However, if the NVS cache is under stress the controller is smart enough in sending the data from volatile cache directly to disks (synchronously). This situation is called DFW bypass and the channel stays disconnected along this data transfer. If you recall the dam story is like creating a bypass in the dam.

- Almost 100% DFW Hits (Quick writes)

  This mode allows the use of the NVS cache for writes. It avoids accessing disks for a write hit and almost 100% of the write misses. In certain controllers, its logic is included in the RLC licensed internal code (LIC).

  Almost 100% DFW hits is also called "quick writes". It allows a DFW miss turn to a hit (provided that adequate NVS space is available).

  The reason causing the access of disks for a write data miss in a DFW mode is the verification of the record length. To clarify this point refer to Figure B-2 and follow the description of the existent two types of write CCWs.

  – *Write format*, also called write count-key-data, formats the 3390/3380 track by overlaying the old records in the track and creating a count with the respective data (usually with a zero content), the rest of the track is erased. The length of the data record is included in the write count-key-data CCW and copied in the count. In this case, the previous record data length is not important because it is overlaid.

    All the write format I/O operations are considered a write hit.

  – *Write modified*, also called write data, change the contents of the data portion of a pre-formatted record which length is already described in the count. The length of the data record is also included in the write data CCW. Any mismatch between this length and the one already specified in the count of the formatted record is posted by the channel (to the application) and in some cases, it may stop the execution of the channel program. Then is clear the need of accessing the 3390/3380 track in a DFW write modified miss because the controller is not able to verify (and compare) the length in the write modified CCW and the count. On the other hand is now clear why the write format is always a hit, because there is no need of such verification.

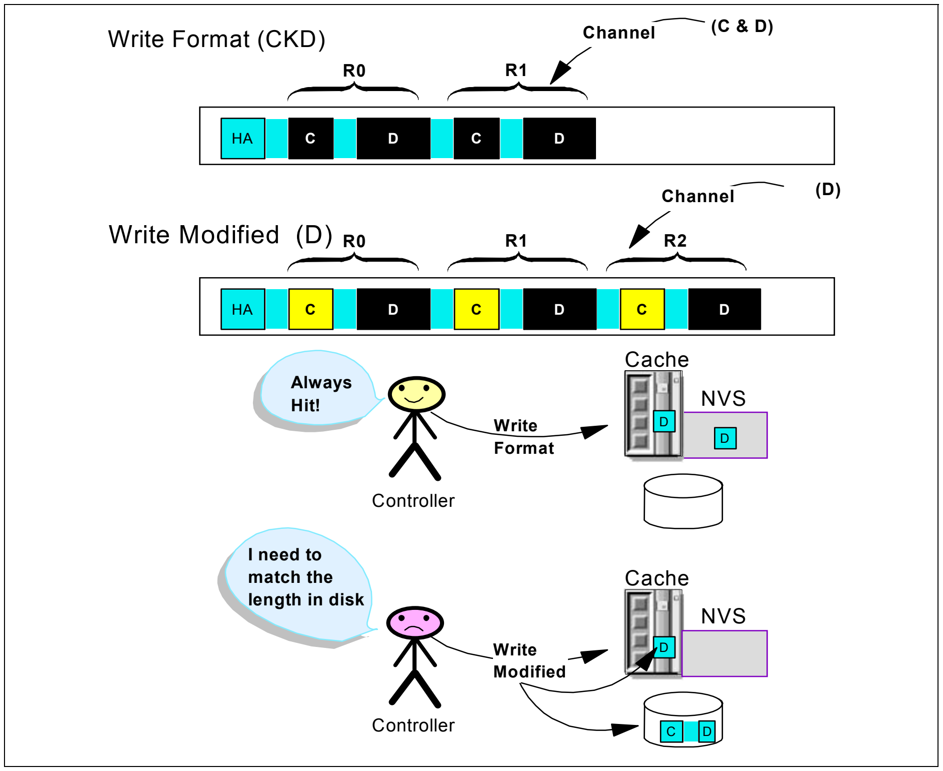


Figure B-2 Types of writes

For quick writes the controller must know or be able to predict the length of the record avoiding the access to the disks.

Quick writes is implemented in two sub-modes associated with record level cache. Remember that in some IBM controllers, quick writes are associated with the record level cache LIC:

– Record Level Cache I:

  It is a joint VSAM/controller implementation.

    Because VSAM is a trusted partner (regular data format) all writes become quick writes, that is, the controller decides not to go to disk to verify the data record length. This function benefits IMS, DB2 and CICS users of VSAM. The data set could be SMS or not, it requires DFSMS/MVS 1.2 or PTF equivalent.

  - Record Level Cache II:

    It is non-adaptive (the controller does by itself), with no Define Extent CCW software intervention. When RLC II is activated it cannot be disabled. In the first access to the record there is a disk access (miss) and the record length is moved and kept in cache for the future requests. It aims to reach "almost 100% writes hits". The controller automatically determines whether to use the track cache algorithm as it processes I/Os to the volumes with cache active. RLC II requires:

    - Volume behind a controller with RLC II installed
    - TLC enabled for that volume
    - DFW enabled for that volume

- Inhibit Cache Load (ICL)

  If a read hit occurs read from cache and LRU update. If a read miss or a write access the disks through volatile cache, no LRU update. Then, cache space is not allocated for any new tracks from DASD. Write operations that do not require access to DASD are not inhibited by this setting.

  Used by DFDSS, DFSORT™ (Sortin), SMS (never-cache and some of the maybe-cache data sets). As a general rule, it maybe used by an application which knows that the record to be requested is not going to be used in the near future (as for cache unfriendly accesses).

  Ignore ICL is an option set in VPD, when your workload is not aware of a huge cache size

- Bypass Cache

  Where the I/O request must be executed in the disks (even if the data is in the cache). However, the data always pass through cache without LRU update, consequently the copy is ready to be demoted. If a hit in the cache, the LRU is updated. Cache images of tracks modified on the device will be invalidated. Tracks modified in cache but not on DASD are destaged to DASD before access is allowed. For Duplex volumes this includes destaging to both devices. DASD Fast Write operation is disabled with this attribute active.

  It is used by ICKDSF, paging, and Write Check access method option.

  Ignore BYP is an option set in VPD, being used when your workload is not aware of a huge cache size.

Using cache modes in a non-SMS data set

An I/O operation towards a non-SMS data set is able to use some of the cache modes. In order to do that, you must use IDCAMS Setcache command to activate:

- Globally in the controller:
  - Cache in general
  - DFW (or the use of NVS)
  - CFW

  RMF in cache subsystem status reporting Figure B-3 shows the result of such operation:

```
----------------------------------------------------------------------------
                     CACHE SUBSYSTEM STATUS
----------------------------------------------------------------------------

SUBSYSTEM STORAGE           NON-VOLATILE STORAGE        STATUS

CONFIGURED      256.0M      CONFIGURED       8.0M       CACHING
AVAILABLE       254.9M      PINNED           0.0        NON-VOLATILE
STORAGE
PINNED            0.0                                   CACHE FAST WRITE
OFFLINE           0.0                                   IML DEVICE AVAILABLE
```

Figure B-3 Cache Subsystem Status report

- Locally in each volume:
  - Normal cache
  - DFW (or the use of NVS)
  - Dual Copy

  RMF in Cache Activity report in Figure B-4 shows the result of such operation:

```
VOLSER     D83STE      NUM    0D84
----------------------------------------------------------------------------
                                                        CACHE DEVICE STATUS
----------------------------------------------------------------------------

CACHE STATUS                                           DUPLEX PAIR STATUS

CACHING             - ACTIVE            DUPLEX PAIR         - NOT ESTABLISHED
DASD FAST WRITE     - ACTIVE            STATUS              - N/A
PINNED DATA              - NONE                             DUAL COPY VOLUME
```

Figure B-4 Cache activity report

These options are passed to the controller by IDCAMS. Without SMS all the data sets in volume follow these rules (normal or DFW). The other cache modes must be declared explicitly by the requester through an IOS, which builds the define extent CCW.

Using cache in an SMS data set

SMS uses the define extent CCW to set some cache modes along the I/O operation for an SMS data set. If there is a conflict between Define Extent and IDCAMS volume setting the more restrictive option prevails.

For example: IDCAMS says non-DFW for the volume and define extent says DFW for the I/O operation, then it will be non-DFW.

However, there are certain cache modes not set by SMS as bypass cache and CFW. In this case the requester should use interface directly with IOS in order to have theses options in the define extent CCW.

Cache usage attributes

An opened SMS data set may have one of three cache usage attributes, in regard to DASD cache:

- **Must-cache data set**: Uses cache/NVS for the I/O operations.
- **Never-cache data set**: Does not use cache (only for buffering) and does not use NVS. The ICL mode is requested in Define Extent CCW.
- **May-cache data set**: Uses cache/NVS depending on the cache/NVS constraints.

The same data set may have one of the above attributes for sequential accessing mode and a different one for direct accessing mode. This is also true for read access and write access. These attributes are based in the SMS storage class (SC) installation parameters (MSR for Direct, MSR for sequential or BIAS respectively). MSR is the desired I/O service time in milliseconds and BIAS the expected dominance of reads or writes operations.

The cache usage attributes are assumed at open time, for a data set. These cache attributes are kept constant until the data set is closed.

Dynamic Cache Management Enhanced (DCME)

DCME is a function in the controller able to produce cache information in a system and in a data set basis. SMS uses data from DCME in order to adjust the use of the cache for may-cache data sets. With DCME, SMS distinguishes between good (cache-friendly) and poor cache (cache unfriendly) candidate data sets when deciding which I/Os to which data sets should be cached.

DCME produces two key measurements:

- *Data set cache behavior*:

  DCME maintains information about the hit ratios achieved by I/Os to each may-cache data set. DCME continuously updates this information so that it can make decisions on which I/Os to which data sets should be cached, based on the most recent I/O activity.

  When considering whether or not to cache I/Os to a data set, two controller resources must be accounted for: the cache and the NVS. A data set might very well be a good user of the cache and a poor user of the NVS. SMS, therefore, maintains two criteria: one general (for all I/Os) and one specific for writes.

  Two data set related indicators are calculated:

  – Overall Hit Ratio: Reads Hits + Writes Hits / Cachable IOs

    Used to decide to cache a Read request.

  – Write Hit Ratio: Write Hits / Write IOs

    Used to decide to cache a Write request.

  A hit is perceived by a disconnect time less that 0.5 ms.

  These indicators are weighed averages of previous values to avoid sudden changes.

- Subsystem load (here the word subsystem means DASD controller)

The DCME in controller produces global data about the cache usage.

A Subsystem Threshold (ST) indicator is timely calculated by SMS from the global DCME data. Higher the ST more cache contention. These statistics are periodically collected by SMS. The time is controlled by DINTERVAL at IGDSMSxx Parmlib (default 150 seconds).

ST reflects the DASD cache performance and is composed by the figures of Read Hit Ratios and DFW bypass for all the cached volumes in the DASD subsystem.

The DFW bypass occurs when a DFW hit request requires NVS, but this storage is not available due to contention. In this case, the I/O request will bypass the NVS and is executed directly from the volatile cache to the disks.

Also, based on the statistics, SMS maintains two global average indicators:

- Cache Control Indicator (CCI), the percent of may-cache I/O requests allowed to use cache.

- NVS Control Indicator (NVSCI), the percent of may-cache DFW I/O requests allowed to use NVS.

These values used for this caculation can be displayed using the D SMS,CACHE command (Figure B-5).

```
IGD002I 18:09:11 DISPLAY SMS 276
 SSID    DEVS     READ     WRITE    HIT RATIO    FW BYPASSES
 00FF       5     N/A      N/A          97%            0
 8900       8     N/A      N/A          98%            0
 8902       5     N/A      N/A          98%            0
 8904       4     N/A      N/A          99%            0
 8903       7     N/A      N/A          99%            0
 8901       4     N/A      N/A          98%            0
 000A       8     N/A      N/A          99%            0
 3000      21     N/A      N/A          99%            0
 8905       6     N/A      N/A          97%            0
 6004       8     N/A      N/A          90%            0
 0028       4     N/A      N/A          98%           24
 00FD       7     N/A      N/A          98%            0
 00FC       4     N/A      N/A          99%            0
 00FE       1     N/A      N/A          98%            0
```

Figure B-5 D SMS,CACHE output

The following legend explains the columns in Figure B-5:

- **Ssid** = Subsystem Identifier
- **Devs** = Number of managed devices attached to subsystem
- **Read** = Percent of data on managed devices eligible for caching
- **Write** = Percent of data on managed devices eligible for DFW
- **Hit Ratio** = Percent of reads with cache hits
- **Fw Bypasses** = Number of fast write bypasses due to NVS overload

An I/O operation for a may-cache data set has three states:

- *Normal*, the data set I/O is allowed to use the cache through the define extent CCW.
- *Inhibit*, the data set I/O does not use the cache that is, the Inhibit Cache Load bit is set on the define extent first CCW of a Read channel program, in order to inhibit the staging of the cache. In the case of a Write channel program, the Inhibit DFW bit is set on the define extent first CCW of a Write channel program, to inhibit DFW, and consequently the staging of NVS.
- *Force*, the data set I/O is cached so that the indicators can be evaluated.

The logic is:

- After open for the first 100 I/Os and the first 100 Writes the I/Os are *forced*.
- The data set indicator is compared with subsystem threshold. If does not exceed, the data set I/Os are going to be inhibited for the next 5000 I/Os. If exceeds, the data set I/Os are going to be normal for a certain amount of time, where the comparison is going to be done again.

So, if the ST indicator is going up, the exclusion from cache for may-cache data sets is gradual. No sudden and dramatic changes in the DASD subsystem performance are caused.

As a DCME by-product DFSM I/O statistics (I/O rates, I/O response time, I/O service time components, caching statistics for reads and writes) are collected in new SMF records in data set and storage class basis.

Never-cache candidates data sets

There are some data sets that are not good candidates for DASD caching, such as:

- Data sets that are processed track-by-track, as the input to the dump function of DFDSS itself. However, in this case the installation does not need to care about this, because DFDSS uses the adequate option (inhibit cache load) in the Define Extent command.
- Data sets which have a very poor direct revisit pattern.
- ASM paging data sets.

Share options analogy

Refer to Figure B-6, to follow this explanation:

Once upon a time, there were two lands (MVS A and MVS B) separated by a river. All the culture accumulated by the people living there was stored in two shared VSAM data sets strategically located in the middle of the separating river. In each land, there were two sets of people, the round-head and the square-head (pay attention that each set lives on both sides of the river). Each head shaped people used their own data set (black data set for the square and white data set for the round head). The data sets were accessed by students, the readers (for read) and by the professors, the updaters (for writes).

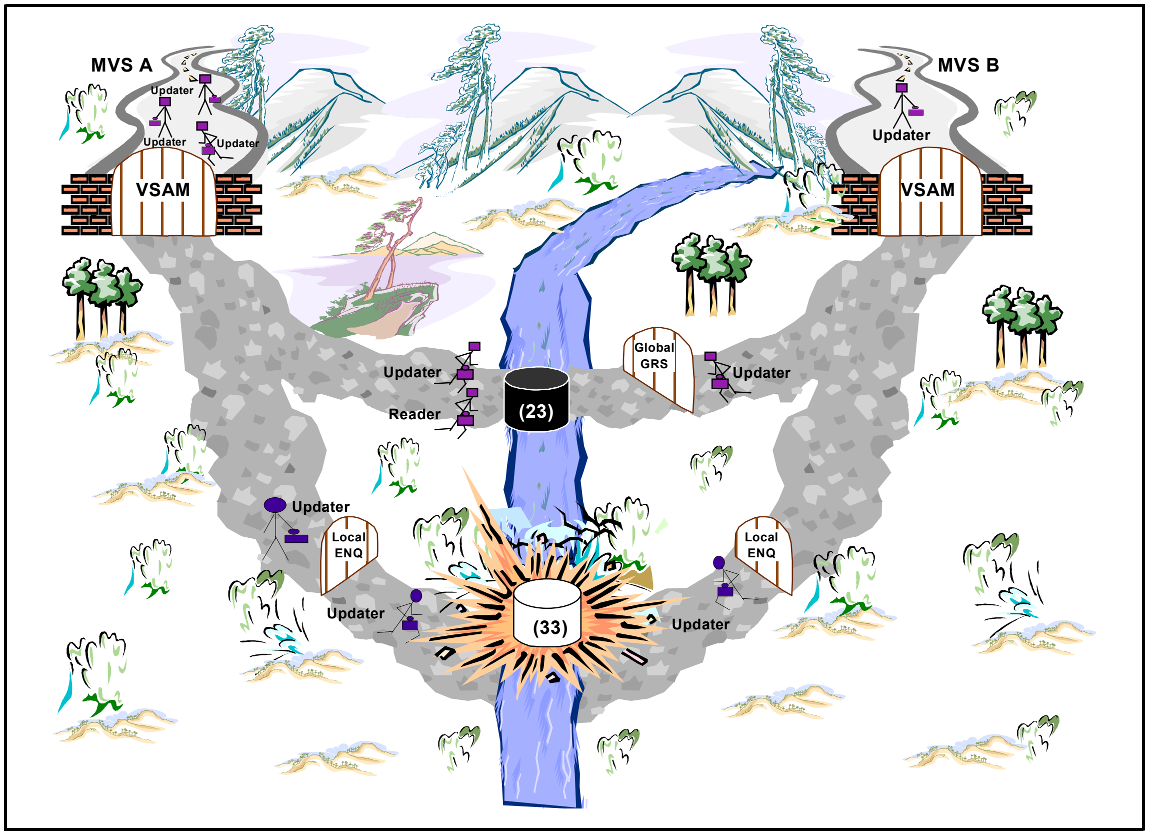


Figure B-6 Sharing VSAM data sets

The square-heads (from the both lands) care about write integrity and not about read integrity, so they choose shareoptions (2 3) for their data set and they use GRS adequately for their purpose.

The round-heads (from the both lands) cared about write and read integrity. They decided to implement that through GRS/ENQ mechanism only. The shareoptions of the round-head data set is (3 3).

The picture is showing what finally happened:

- In the square-head story, we may see, updaters being held by the VSAM gate in both sides of the river. This was caused by cross region 2 in shareoptions. Only one updater succeeds in passing the gate (in each side). All others open for output fail with a return code in ACB. However, the reader in MVS A was not blocked by VSAM gate (no read integrity).

To guarantee write integrity between updaters from the two lands, a global GRS/ENQ is implemented guaranteeing that a second updater (from MVS B in the picture) which arrived last, be held at the GRS gate.

- For the round-head story, there are no VSAM gates for the round-heads because of cross region option 3. In both sides they implement a local ENQ gate to guarantee read and write integrity. That is, only one updater or several readers from each MVS are allowed to the round-head data set.

  However, they did not read the *GRS Primer* book. The ENQ name is not made global to GRS, and then two updaters (each from each land) are allowed to update concurrently the round-head data set, then blowing up the integrity.

Symptoms (messages) from a broken data set

The most common messages associated with broken data sets events are:

- `IDC3302I ACTION ERROR ON dsname`

  Explanation: An error was detected while attempting to access the data set. See the associated message in the program listing for explanation.

- `IDC3308I ** DUPLICATE RECORD xxx`

  Explanation: The output data set of a Repro command already contains a record with the same key or record number. In the message text:

  - xxx For an indexed data set, the first five bytes of the duplicate key, in hexadecimal format. For a relative record data set, the relative record number (in decimal) of the duplicate record.
  - System Action: The system does not write the record. The system continues processing with the next record, unless this is a copy catalog and a duplicate record is encountered or there has been a total of four errors. The system ends in either case. For example, if a duplicate record is encountered while Repro is copying a catalog, the system ends processing.

  If the record in the input file with the duplicated key is to replace the one in the data set, you should specify the replace option. If not, check your Repro input.

- `IDC3314I RECORD xxx OUT OF SEQUENCE`

  Explanation: The key of the record to be written is less than or equal to the key of the last record written. In the message text:

  - xxx The first five bytes in hexadecimal format of the key of the record that is out of sequence.

- System Action: If the output data set is a virtual storage access method (VSAM) data set, the system ends processing of the command after four errors.
- Application Programmer Response: Rearrange the records to be written so that they are in ascending key sequence. The record can be written to the data set using skip sequential processing. Run the job again and the output data set will be opened for skip sequential processing (because data already exists in the data set) and records that were out of sequence will be written.

-
  ```
  IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
           {RPLFDBWD=nnnnnnnn}
  ```

  An error was encountered during VSAM open, close, or action request processing, as indicated in the text of the message:

  nnnnnnnn The meaning can be found in DFSMS/MVS Macro Instructions for Data Sets.

  rc The return code, as follows:

  - For CLOSE errors, we present only the return codes associated with broken data set situation:

128	Index search horizontal chain pointer loop encountered.
136	Not enough virtual storage was available in the program's address space for a work area for CLOSE.
184	An uncorrectable I/O error occurred while VSAM was completing outstanding I/O requests.
246	The compression management services (CMS) close function failed.

  - For OPEN errors, we present only the return codes associated with broken data set situation:

76	Attention message: The interrupt recognition flag (IRF) was detected for a data set opened for input processing This indicates that DELETE processing was interrupted. The structure of the data set is unpredictable; the access method services DIAGNOSE command may be used to check the data set for structural errors.

88	A previous extend error has occurred during EOV processing of the data set.
96	Attention message: an unusable data set was opened for input.
104	Attention message: the time stamp of the volume on which a data set is stored doesn't match the system time stamp in the volume record in the catalog; this indicates that extent information in the catalog record may not agree with the extents indicated in the volume's VTOC.
108	Attention message: the time stamps of a data component and an index component do not match; this indicates that either the data or the index has been updated separately from the other. Check for possible duplicate VVRs.
116	Attention message: the data set was not properly closed or was not opened. If the data set was not properly closed, then data may be lost if processing continues. Use the access method services VERIFY command to attempt to close the data set properly. In a cross-system shared DASD environment, a return code of 116 can have two meanings: - The data set was not properly closed. - The data set is opened for output on another processor. **Note**: If you use the VERIFY command, this message can appear again when VERIFY processing opens the data set. If VERIFY processing then successfully closes the data set, VERIFY processing issues condition code 0 at the end of its processing. In addition, an empty cluster cannot be verified.
132 One of the following errors occurred:	
	- Not enough storage was available for work areas. - The format-1 DSCB or the catalog cluster record is incorrect.
136	Not enough virtual-storage space is available in the program's address space for work areas, control blocks, or buffers.

140	The catalog indicates this data set has an incorrect physical record size.
160	The operands specified in the ACB or GENCB macro are inconsistent with each other or with the information in the catalog record. This error can also occur when the VSAM cluster being opened is empty.
164	An uncorrectable I/O error occurred while VSAM was reading the volume label.
168	The data set is not available for the type of processing specified, or an attempt was made to open a reusable data set with the reset option while another user had the data set open.
184	An uncorrectable I/O error occurred while VSAM was completing an I/O request.
190	An incorrect high-allocated RBA was found in the catalog entry for this data set. The catalog entry is bad and will have to be restored.
192	An unusable data set was opened for output.
193	The interrupt recognition flag (IRF) was detected for a data set opened for output processing.
194	Direct access of a compressed data component is not allowed.
200	Volume is unusable.
212	The ACB MACRF specification is GSR or LSR and the data set requires create processing.
232	Reset (ACB MACRF=RST) was specified for a nonreusable data set and the data set is not empty.
240	Format-4 DSCB and catalog time stamp verification failed during volume mount processing for output processing.

– For a Logical I/O Error

4	End of data set encountered (during sequential retrieval), or the search argument is greater than the high key of the data set. Either no EODAD routine is provided, or one is provided and it returned to VSAM and the processing program issued another GET.

8	You attempted to store a record with a duplicate key, or there is a duplicate record for an alternate index with the unique key option.
12	You attempted to store a record out of ascending key sequence in skip-sequential mode; record had a duplicate key; for skip-sequential processing, your GET, PUT, and POINT requests are not referencing records in ascending sequence; or, for skip-sequential retrieval, the key requested is lower than the previous key requested. For shared resources, buffer pool is full.
16	Record not found.
20	Record already held in exclusive control by another requester.
28	Data set cannot be extended because VSAM cannot allocate additional direct-access storage space. Either there is not enough space left to make the secondary allocation request, you attempted to increase the size of a data set while processing with SHROPT=4 and DISP=SHR, or the index CI is not large enough to hold the entire CA. This error could also be due to a data set trying to extend beyond 4GB on a system that does not support extended addressability.
32	An RBA specified that does not give the address of any data record in the data set.
40	Insufficient virtual storage in the user's address space to complete the request.
116	During initial data set loading (that is, when records are being stored in the data set the first time it's opened), GET, POINT, ERASE, direct PUT, and skip-sequential PUT with OPTCD=UPD are not allowed. During initial data set loading, VERIFY is not allowed except for an entry-sequenced data set (ESDS) defined with the RECOVERY option. For initial loading of a relative record data set, the request was other than a PUT insert.
128	A loop exists in the index horizontal pointer chain during index search processing.
144	Incorrect pointer (no associated base record) in an alternate index.

156	An addressed GET UPD request failed because the control interval flag was on, or an incorrect control interval was detected during keyed processing. In the latter case, the control interval is incorrect for one of the following reasons: - A key is not greater than the previous key. - A key is not in the current control interval. - A spanned record RDF is present. - A free space pointer is incorrect. - The number of records does not match a group RDF record count. - A record definition field is incorrect. - An index CI format is incorrect.
212	Unable to split index; increase index CI size.
236	Validity check error for SHAREOPTIONS 3 or 4.
245	A severe error was detected by the compression management services (CMS) during compression processing.
246	A severe error was detected by the compression management services (CMS) during decompression processing.
250	A valid dictionary token does not exist for the compressed data set. The data record cannot be decompressed.
254	I/O activity on the data set was not quiesced before the data set was closed.

- `IDC3350I synad[SYNAD]message[from VSAM]`

  Explanation: An I/O error occurred for a VSAM data set. The message text, format, and explanation of VSAM I/O errors are provided in DFSMS/MVS Macro Instructions for Data Sets.

- `IEC070I rc[(sfi)]- ccc,jjj,sss,ddname, dev,ser,xxx,dsname,cat`

  Explanation: An error occurred during EOV (end-of-volume) processing for a VSAM data set. In the message text:

rc	Reason code. This field indicates the reason for the error. The reason codes, their meanings, and the corresponding system action and required responses are listed under message IEC161I.
sfi	Subfunction information (error information returned by another subsystem or component). This field appears

	only for certain return codes, and its format is shown with those codes to which it applies.
ccc	Problem Determination (PDF) Function code. The PDF code is for use by IBM if further problem determination is required. If the PDF code has meaning for the user, it will be documented with the corresponding Reason Code (rc).

```
IEC070I   RC32 , RC202 , RC8 , RC18 , RC24 , RC104 , RC203 MSGIEA000I
IOS000I CMD REJ, COMMAND REJECT

ADR970E   HSM  MISSING CI within SEQUENCE SET TRACK TRACKS TRK TRKS TRKS=0
TRACKS=0 EXTENT
```

IDCAMS EXAMINE messages

The most frequent error messages issued by EXAMINE command are:

```
IDC01714I  ERROR LOCATED at OFFSET xxx

IDC01720I  INDEX CONTROL INTERVAL DISPLAY at RBA xxx FOLLOWS

IDC11703I  DUPLICATE KEYS in INDEX

IDC11704I  INDEX KEYS are NOT in SEQUENCE

IDC11705I  INDEX RECORD CONTAINS DUPLICATE INDEX POINTERS

IDC11707I  DUPLICATE INDEX POINTERS FOUND in SEQUENCE SET

IDC11711I  INDEX CONTROL INTERVAL COUNT ERROR

IDC11715I INDEX HIGH-USED RBA is NOT a MULTIPLE of CI SIZE IDC11724I  DATA
COMPONENT CA  NOT  KNOWN to SEQUENCE SET IDC11725I  SEQUENCE SET RBA
INCONSISTENT with VSAM- MAINTAINED RBA

IDC11727I  INDEX HIGH-USED RBA GREATER THAN HIGH-ALLOCATED

IDC11728I  DATA FOUND in EMPTY CI

IDC11733I  DATA COMPONENT KEY SEQUENCE ERROR MSGIDC11758I  SOFTWARE EOF FOUND
in INDEX CI

IDC11763I  RBA of INDEX CI GREATER THAN HIGH-USED RBA IDC11771I  INVALID RBA
GENERATED

IDC11772I  HORIZONTAL POINTER CHAIN LOOP
```

C

Catalog performance

In this appendix we discuss various aspects of catalog performance:

- Catalog performance
- Catalog CPU consumption
- Hints and tools to analyse performance problems related with catalogs.

Performance

As the number of systems sharing resources grow in today's parallel sysplex environment, contention for resources can increase resulting in slower response times or reduced throughput. One of the resources that could have increased contention is a catalog. Elongation of catalog requests has the potential to not only effect the performance of work within the system performing the request, but can effect other systems in the sysplex as well.

There have been performance enhancements made to the catalog sharing protocol and to the component that manages these requests, Global Resource Serialization (GRS). These should be investigated if they have not already been implemented as they were designed for environments where there are many systems sharing resources.

Enhanced Catalog Sharing

DFSMS/MVS V1R5 introduced Enhanced Catalog Sharing (ECS) which offered an alternative to the VVDS mode of sharing that was introduced in DFP V3. If catalog sharing is done in VVDS mode, information is read from the VVDS on the volume where the catalog is allocated to see if the catalog has been changed by another system since the last time it was accessed by the system wanting to access the catalog. This requires multiple ENQs and I/O. With ECS, the VVDS is moved into a coupling facility reducing the serialization and I/O requirements that VVDS mode has.

GRS configuration

GRS Star configuration was introduced in OS/390 V1R2 and is an excellent alternative to GRS Ring configuration.

In a GRS Ring configuration, the sharing systems communicate with each other via channel-to-channel connections or use XCF. The primary means for passing information is the ring system authority message (RSA-message). The RSA-message contains the information each system needs to protect the integrity of resources. The RSA-message passes from one system in the ring to another. No system can grant a request for a global resource until the other systems in the ring know about the request. With this architecture, ENQ delay time will increase every time a new system is added.

The introduction of GRS Star Topology provides an environment where additional systems can be introduced into a parallel sysplex with minimal impact to resource serialization performance. GRS Star utilizes the coupling facility in a parallel sysplex and uses it as the hub of the star. Thus, rather than passing the RSA-message around the ring to all of the systems before a serialization request is granted, serialization information is stored in a central location (the coupling facility). This eliminates the overhead of communicating with all of the systems that used to make up the Ring.

Diagnosing prolonged catalog ENQ times

When analyzing prolonged catalog ENQ times, verify:

- LPAR definition
- GRS configuration
- Catalog workload
- I/O response time and workload

Since elongation of catalog requests could be the result of increased workload, historical data is required to determine if workload has actually increased. SMF records 70-79 and `F CATALOG,REPORT,PERFORMANCE` command output can provide information concerning workload trends and system utilization.

Prior to any diagnosis, the following tasks should be performed to gather the data that will be required for diagnosing a perceived catalog performance problem:

- Document LPAR configuration: This can be obtained from the RMF Partition Data Report.
- RMF records 70-79: Insure that SMFPRMxxXnn in PARMLIB includes SMF records 70-79. Include type 79 subtype (7) for SENQ reports. Refer to *z/OS MVS System Management Facitilies (SMF)*, SA22-7630.
- Set up and start RMF reporting. Refer to the *RMF User's Guide,* SC33-7990 for setting up RMF Post Processor, Monitor II, and Monitor III sessions.

  Specify ENQ(DETAIL) to report job names owning the resource, have the longest period of contention, and who are waiting for a resource.
- Insure that the dispatching priority for CAS and GRS has been allowed to default.
- If GRS is in ring mode:
  - Insure that SYSIGGV2 is converted and SYSZVVDS is not converted.
  - Set RESMIL to one.
  - Run GENQRESP found in SYS1.SAMPLIB(ISGNQRSP) to determine base response time for a GRS ENQ/DEQ request.

- Implement ENQ/RESERVE/DEQ Monitor described in chapter 3 of the *z/OS MVS Planning: Global Resource,* SA22-7600 manual.
- For catalog, periodically issue the following series of commands (possibly at the beginning of each shift):
  - `F CATALOG,REPORT` to display CAS specifications.
  - `F CATALOG,REPORT,CACHE` to display catalog cache hits.
  - `F CATALOG,LIST` to display waits, etc.
  - `F CATALOG,REPORT,PERFORMANCE` to record number of entries into catalog, number of ENQs, etc.
  - `F CATALOG,REPORT,PERFORMANCE(RESET)` to reset counters.

Catalog LOCATE Flow

If you suspect that catalog contention is causing bad system performance, an understanding of how a catalog request is processed and the components involved is needed.

The scenarios described below assume VVDS mode sharing and a GRS Ring environment, although GRS Star could be in use. Example C-1 is the output from a GTF trace during a catalog LOCATE request:

Example: C-1 Flow of a Catalog LOCATE request

```
SVC 26
SVC 56 SYSZTIOT
SVC 48 SYSZTIOT
SVC 56 SYSIGGV2 CATALOG.MVSICFM.VMASTER
SVC 56 SYSZVVDS CATALOG.MVSICFM.VMASTER
SVC 56 SYSZVVDS VOLA
SSCH   72C6
IO     72C6
SVC 48 SYSZVVDS VOLA
SVC 48 SYSZVVDS CATALOG.MVSICFM.VMASTERSVC 48 SYSIGGV2 CATALOG.MVSICFM.VMASTER
SVC 56 SYSZPCCB PCCB
SVC 48 SYSZPCCB PCCB
SVC 56 SYSIGGV2 CAT1.UCAT
SVC 56 SYSZVVDS CAT1.UCAT
SVC 56 SYSZVVDS VOLB
SSCH   7526
IO     7526
SVC 48 SYSZVVDS VOLB
SVC 48 SYSZVVDS CAT1.UCAT
SVC 48 SYSIGGV2 CAT1.UCAT
```

The trace entries listed above are:

- SVC 26: LOCATE
- SVC56: ENQ
- SVC48: DEQ
- SSCH: Start subchannel
- IO: I/O to the device

The trace shows the resources being serialized and when I/O is performed to obtain the requested catalog entry. The number of Locates issued and the length of time it takes to process ENQ requests are the primary factors that determine how long it will take to return a catalog request.

LPAR considerations

When there are systems in the parallel sysplex that are running in LPAR mode and there are multiple partitions in one CEC competing for resources, the LPAR definitions should be examined. The reason for this is that before CAS or GRS can do any work, they have to be dispatched in their z/OS or OS/390 image and the partition they are running in has to be dispatched to a physical processor. To insure that the partitions are given enough access to the physical processors:

- Check logical CP configuration
  - Number of CPs available to partition's operating system
  - Operating system dispatches work to a logical processor
  - LPAR associates a physical processor to a logical processor to execute work
- Weight and capping:
  - Weight specifies the amount of capacity guaranteed to partition at time of high CPU utilization
  - Capping specifies the amount of capacity that the partition is not allowed to exceed even if their are processor resources available
  - Number of logical CPUs and weight are static parameters entered on LPAR definition panels
- Review RMF CPU and Partition reports for partition performance. The RMF Partition Data Report can be used to gather information concerning LPAR

A situation which could cause perceived bad performance is an LPAR environment referred to as having 'short' engines. For example, consider you have a CEC that has 4 physical CPs, each having a capacity of 25 MIPs:

- There are 2 partitions defined each with 3 logical CPs sharing the 4 physical CPs.
  - Partition 1 has a weight of 75 and
  - Partition 2 has been defined with a weight of 25 (the weight of the partition is less than the total MIPs of the logical processors assigned to it).

  Assume that this a period of time where there is high CPU activity for both partitions.

  - Partition 2 is dispatched and GRS is dispatched to one of the physical CPs.
  - Partition 2 reaches its weight limit and partition 1 requires a CP.
  - The physical CP that had been assigned to partition 2 and had GRS dispatched is taken away from partition 2 and assigned to partition 1.

  The z/OS or OS/390 image that is running in partition 2 is not aware that the physical processor has been taken away from it and the work GRS was to perform has to wait.

  The work GRS was to perform could be a serialization request from another system that was to be passed onto another system. This serialization request isl not granted until the GRS in partition 2 is once again assigned a physical processor. This impacts the length of time it takes for a request to be granted and if it is a serialization request for a catalog or VVDS, could elongate the elapsed time of a LOCATE request.

GRS environment

Catalog requests could be prolonged due to a poorly tuned GRS configuration or application contention for the resource. General tuning information for GRS ring and star configurations can be found in the chapter 10 of *z/OS MVS Planning: Global Resource,* SA22-7600 manual. The following information are excerpts taken from the manual. For more detail and complete information, you should refer to *z/OS MVS Planning: Global Resource,* SA22-7600.

The first step in determining what actions to take is to discriminate between a problem with the global resource serialization complex and the applications it serves. There are several kinds of problems that can occur which will affect global resource serialization processing:

1. Tuning:

   A poorly tuned system can elongate global resource serialization requests (ENQ and DEQ). An example of a poorly tuned system could be

   a. In a GRS Ring complex: Where some of the systems have too high a RESMIL value.
   b. In a GRS Star complex: Where the lock structure is too small, causing excessive false contention in the lock structure.

Tuning the complex will alleviate these problems.

2. Intersystem communication breakdown:

   Global resource serialization relies on inter-system communication, through XCF communication facilities, which can be either CTCs or coupling facility signaling structures. Communication failures or delays may cause global resource serialization to take recovery actions which can delay and/or elongate ENQ and DEQ request processing.

3. Coupling facility availability:

   The loss of a coupling facility may cause problems with a global resource serialization ring. In star mode, if the ISGLOCK structure fails or the containing coupling facility is lost, the systems in the sysplex cooperate to rebuild the structure.

4. Resource allocation:

   Even if your global resource serialization complex is well tuned, a combination of applications, system utilities, and online users can impede workload progress due to the use of resources. For example, a long running job or utility can hold data sets exclusively, effectively blocking other jobs and users from proceeding. In more extreme cases, it is possible that a set of requests can cause a deadlock for resource requests by causing a situation where a set of users requires resources held by other users. This situation can only be remedied by breaking the deadlock, usually by canceling one or more of the jobs in the deadlock.

To determine what the GRS configuration or environment is, issue **`D GRS,ALL`**. It displays:

- System information
- CTC link information
- Resource contention information
- RNL change information
- The contents of all the RNLs

A subset of the `D GRS,ALL` command can be used to determine if the GRS complex is operating normally. **`D GRS,SYSTEM`** displays:

- System name of each system in the GRS complex
- The state of each system in the GRS star complex
  - Connecting
  - Connected
  - Rebuilding
- The state of each system in the GRS ring complex
  - Active

  - Inactive
  - Quiesced
  - Joining
  - Restarting
  - Migrating
- The communication status of each system in the GRS complex
  - For a ring complex:
    - Whether or not there is a functioning CTC link between this system and the system name whose information is being displayed
    - When using XCF signaling, the XCF paths being used
  - For a star complex:
    - The coupling facility structure name for the GRS lock structure

  In aGRS Ring configuration, if the GRS complex is running without disruption, slowdowns are usually caused by an improperly tuned ring. Delays of the RSA message can become pronounced on larger complexes, delaying the initiation of new workload. To speed up the ring, the installation can take one, two, or all of the following actions:

  - Speed up the RSA: Speed of the RSA message is dependent upon the RESMIL value. Try using a RESMIL of 1 or 2 on all of the systems
  - Use ring acceleration:

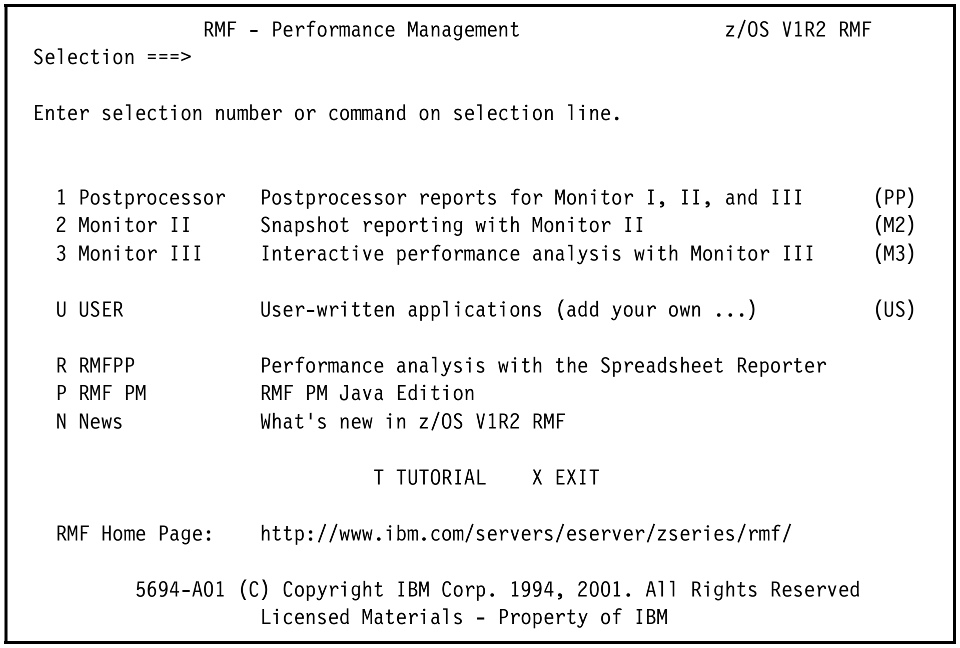
    Reduces the number of systems that must see an ENQ before the issuing system may grant ownership of a resource. There are possible integrity concerns but the opportunity for such failures is small. Specifying ACCELSYS(2) can reduce the overall response time for an ENQ/DEQ request
  - Use GRS Star configuration: Star configuration outperforms any ring complex by orders of magnitude.

There is more information in the *z/OS MVS Planning: Global Resource,* SA22-7600 manual concerning:

- Ring disruption recovery
- GRS ring rebuild
- XCF/XES connectivity and performance
- ISGLOCK structure request processing

Catalog contention

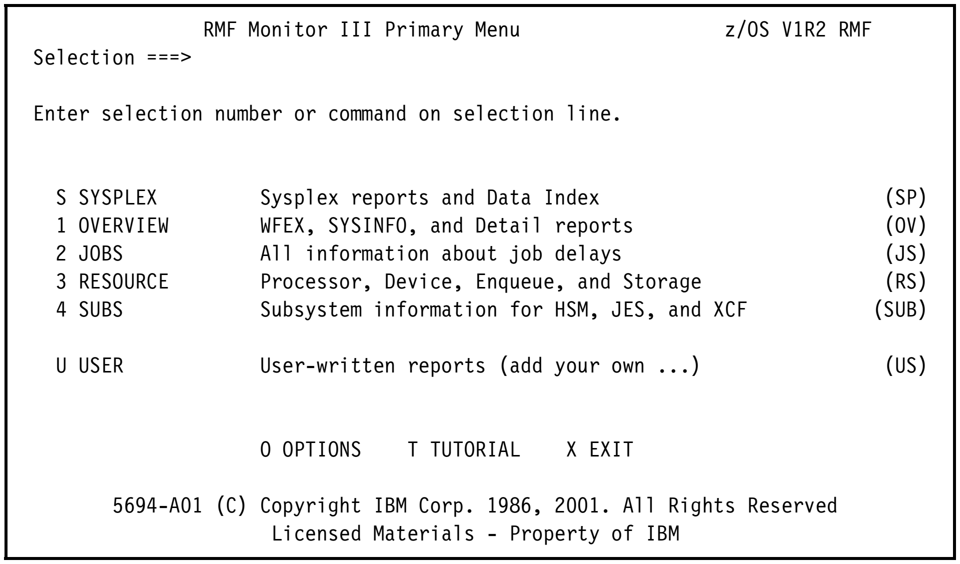
If system performance appears to be bad, RMF is good place to start with your investigation. From the ISPF TSO command shell (option 6), enter the **RMF** command. This command presents you a screen with the various RMF reporting options, as shown in In Figure C-1.



```
               RMF - Performance Management                  z/OS V1R2 RMF
Selection ===>

Enter selection number or command on selection line.


  1 Postprocessor   Postprocessor reports for Monitor I, II, and III      (PP)
  2 Monitor II      Snapshot reporting with Monitor II                    (M2)
  3 Monitor III     Interactive performance analysis with Monitor III     (M3)

  U USER            User-written applications (add your own ...)          (US)

  R RMFPP           Performance analysis with the Spreadsheet Reporter
  P RMF PM          RMF PM Java Edition
  N News            What's new in z/OS V1R2 RMF

                              T TUTORIAL    X EXIT

  RMF Home Page:    http://www.ibm.com/servers/eserver/zseries/rmf/

         5694-A01 (C) Copyright IBM Corp. 1994, 2001. All Rights Reserved
                    Licensed Materials - Property of IBM
```

Figure C-1 RMF Monitor panel

If the application work is running slower than it was before, it could be caused by delays. These delays could be caused by a unit of work waiting for requests to be processed by DFSMShsm, JES, Operator Reply, GRS, and several other system functions. From the RMF Monitor Panel, select option 3, for Monitor III. Then Monitor III options are shown in Figure C-2.



```
               RMF Monitor III Primary Menu                 z/OS V1R2 RMF
Selection ===>

Enter selection number or command on selection line.


  S SYSPLEX         Sysplex reports and Data Index                      (SP)
  1 OVERVIEW        WFEX, SYSINFO, and Detail reports                   (OV)
  2 JOBS            All information about job delays                    (JS)
  3 RESOURCE        Processor, Device, Enqueue, and Storage             (RS)
  4 SUBS            Subsystem information for HSM, JES, and XCF        (SUB)

  U USER            User-written reports (add your own ...)             (US)


                    O OPTIONS    T TUTORIAL    X EXIT

       5694-A01 (C) Copyright IBM Corp. 1986, 2001. All Rights Reserved
                     Licensed Materials - Property of IBM
```

Figure C-2 RMF Monitor III Primary Menu

If a particular job is perceived to be running slowly, you can use the option 2 to display information concerning what could be causing a job to be delayed. If it is because of catalog ENQ contention, it will be shown RMF Job Report Selection report. Selecting option 2 from the RMF Monitor III Primary Menu displays the panel shown in Figure C-3.

```
                RMF Job Report Selection Menu
Selection ===>

Enter selection number or command and jobname for desired job report.

  Jobname ===> MYJOB___

  1 DEVJ           Delay caused by devices                    (DVJ)
 1A DSNJ           .. Data set level                          (DSJ)
  2 ENQJ           Delay caused by ENQ                         (EJ)
  3 HSMJ           Delay caused by HSM                         (HJ)
  4 JESJ           Delay caused by JES                         (JJ)
  5 JOB            Delay caused by primary reason          (DELAYJ)
  6 MNTJ           Delay caused by volume mount               (MTJ)
  7 MSGJ           Delay caused by operator reply             (MSJ)
  8 PROCJ          Delay caused by processor                   (PJ)
  9 QSCJ           Delay caused by QUIESCE via RESET command    (QJ)
 10 STORJ          Delay caused by storage                     (SJ)
 11 XCFJ           Delay caused by XCF                         (XJ)
```

Figure C-3 RMF III Job Selection Menu

Then select option 2. This will tell you if ENQs are causing the delays for this particular job.

If heavy contention for a catalog resource is suspected after using RMF, you could use `D GRS` commands to further investigate. You can issue commands to see if there is contention and if contention exists, which job is blocking other requesters for a particular resource.

By itself, resource contention is not a sign of a problem. However, contention held for a long period of time among the same resources and requesters may be an indication of a problem.

GRS provides diagnostic commands to help you determine the source of contention:

- **`DISPLAY GRS,CONTENTION`**

  Provides an alphabetized list of all visible ENQ resources that are in contention. Each resource is reported with the owners and waiters of the resource.

  The `DISPLAY GRS,CONTENTION` command only reports contention for SCOPE=SYSTEM resources that are allocated on the system where the command executed. It does *not* report contention for SCOPE=SYSTEM resources on other systems in the Complex.

- **`DISPLAY GRS,ANALYZE,WAITER`**

  Provides a list of requesters that have been waiting the longest for ENQ resources. Each waiter is reported with:

  – The resource name and scope.
  – The count of waiters and blockers of the resource.
  – The top blocker of the resource.

  Each waiter is reported with its wait time, system resource that it was ENQed on, and the type of access (shared or exclusive) requested. The counts of waiters and blockers are explicitly returned only when the count is greater than one.

- **`DISPLAY GRS,ANALYZE,BLOCKER`**

  Provides a list of the requesters that have been blocking ENQ resources for the longest time. Each blocker is reported with:

  – Resource name and scope.
  – The count of waiters and blockers of the resource.

  The block time, system the blocker ENQed from, jobname, and the type of access requested are also reported.

- **`DISPLAY GRS,ANALYZE,DEPENDENCY`**

  Provides resource allocation dependency analysis:

  a. Starting with each of the longest ENQ waiters, an analysis is performed, iteratively chaining from waiter to top blocker until either a request that is not waiting is found, or a resource allocation deadlock is detected.

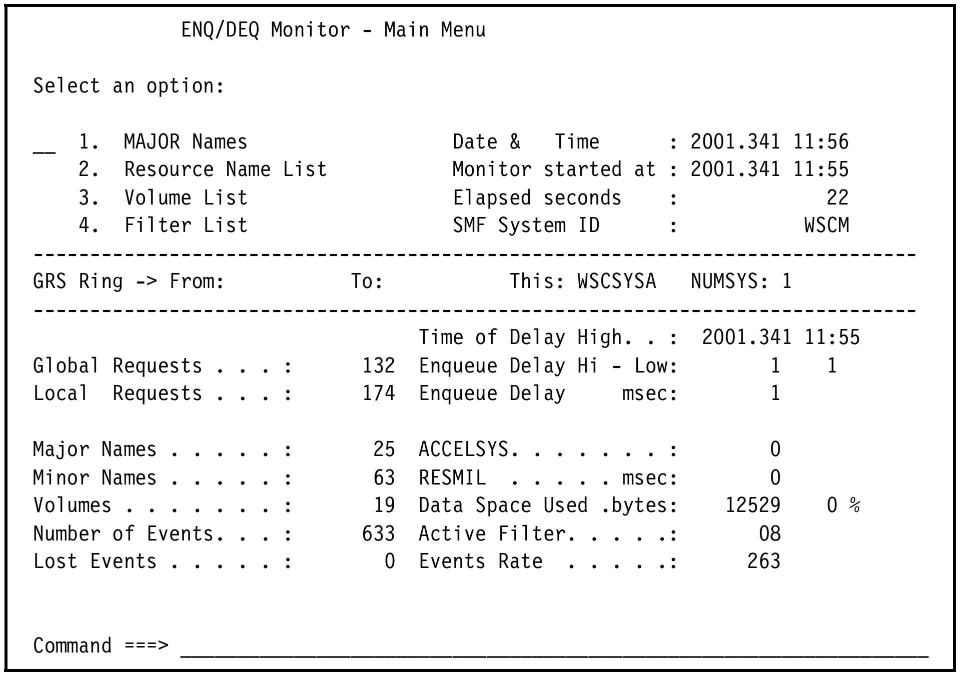
  b. Starting with the top blockers of a specified resource, an analysis is performed, iteratively chaining from waiter to top blocker until either a request that is not waiting is found, or a resource allocation deadlock is detected.

The various forms of the `DISPLAY GRS,ANALYZE` command are available on OS/390 Release 3 and higher with APAR OW38979 installed on the system.

The *z/OS MVS Planning: Global Resource,* SA22-7600manual, in chapter 10, demonstrates how these command can be used to diagnosis suspected contention problems.

Once it has been determined that catalog contention exists, the next step is to determine whether it's due to the length of time it takes to satisfy a catalog request, an increase in CAS workload, or a combination of both. The use of GRS ENQ/RESERVE/DEQ Monitor and `F CATALOG,REPORT PERFORMANCE` commands are useful tools for determining the length of ENQs and how many are being issued.

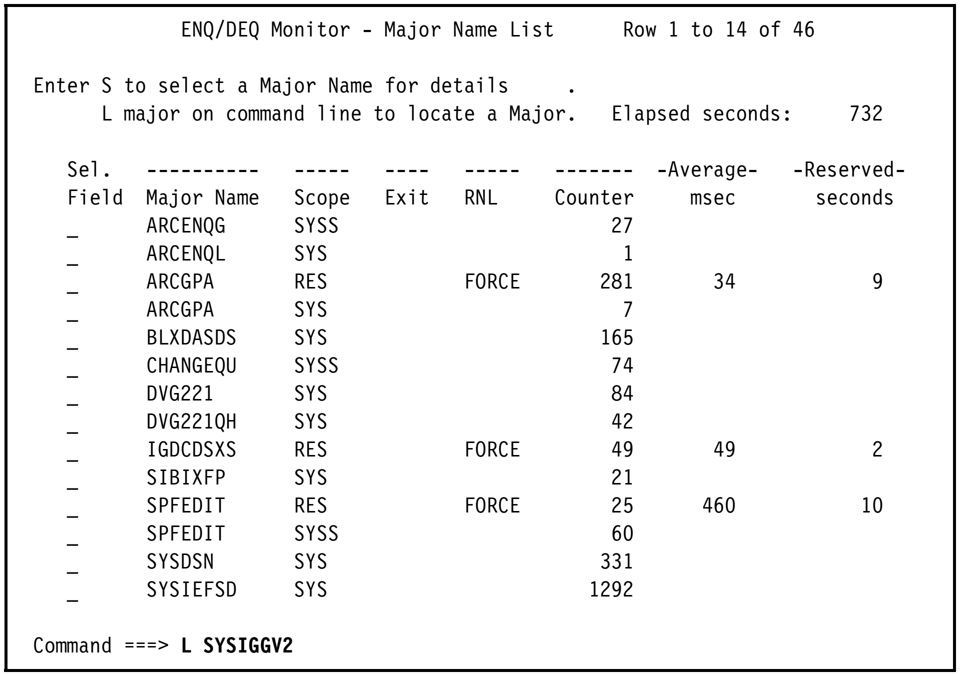
To implement the ENQ/RESERVE/DEQ Monitor, refer to chapter 2 of the *z/OS MVS Planning: Global Resource,* SA22-7600manual. Once invoked, the panel shown in Figure C-4 is displayed:



```
             ENQ/DEQ Monitor - Main Menu

 Select an option:

 __  1.  MAJOR Names                 Date &   Time      : 2001.341 11:56
     2.  Resource Name List          Monitor started at : 2001.341 11:55
     3.  Volume List                 Elapsed seconds    :            22
     4.  Filter List                 SMF System ID      :          WSCM
 ------------------------------------------------------------------------------
 GRS Ring -> From:           To:           This: WSCSYSA   NUMSYS: 1
 ------------------------------------------------------------------------------
                                  Time of Delay High. . :  2001.341 11:55
 Global Requests . . . :      132  Enqueue Delay Hi - Low:        1    1
 Local  Requests . . . :      174  Enqueue Delay     msec:        1

 Major Names . . . . . :       25  ACCELSYS. . . . . . . :        0
 Minor Names . . . . . :       63  RESMIL  . . . . . msec:        0
 Volumes . . . . . . . :       19  Data Space Used .bytes:    12529    0 %
 Number of Events. . . :      633  Active Filter. . . . .:       08
 Lost Events . . . . . :        0  Events Rate  . . . . .:      263


 Command ===> __________________________________________________________________
```

Figure C-4 ENQ / DEQ Main Menu Panel

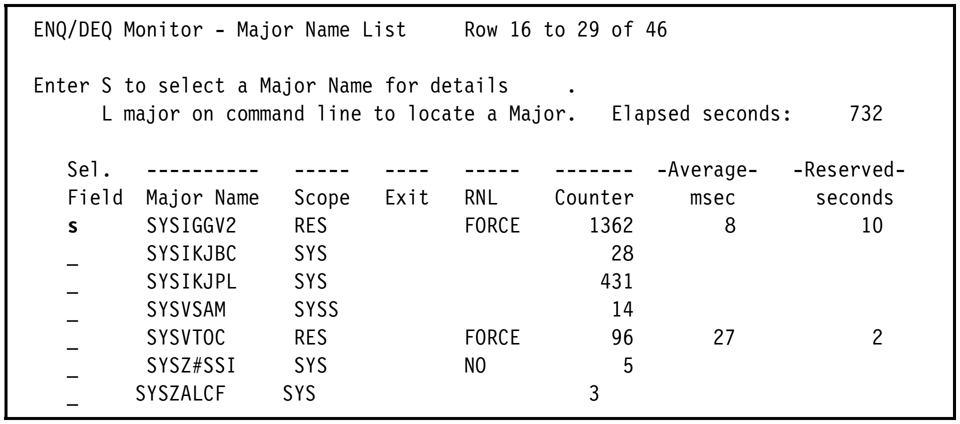
The Main Menu panel summarizes the active GRS options and activity. Selecting option 1, The Monitor shows the ENQ activities listed by major names, RNL action and the number of global and local ENQs if global resource serialization is active, as shown in Figure C-5.



```
             ENQ/DEQ Monitor - Major Name List      Row 1 to 14 of 46

Enter S to select a Major Name for details     .
      L major on command line to locate a Major.   Elapsed seconds:     732

   Sel.   ----------   -----   ----   -----   ------- -Average-   -Reserved-
   Field  Major Name   Scope   Exit   RNL     Counter   msec       seconds
   _      ARCENQG      SYSS                        27
   _      ARCENQL      SYS                          1
   _      ARCGPA       RES            FORCE       281      34             9
   _      ARCGPA       SYS                          7
   _      BLXDASDS     SYS                        165
   _      CHANGEQU     SYSS                        74
   _      DVG221       SYS                         84
   _      DVG221QH     SYS                         42
   _      IGDCDSXS     RES            FORCE        49      49             2
   _      SIBIXFP      SYS                         21
   _      SPFEDIT      RES            FORCE        25     460            10
   _      SPFEDIT      SYSS                        60
   _      SYSDSN       SYS                        331
   _      SYSIEFSD     SYS                       1292

Command ===> L SYSIGGV2
```

Figure C-5 ENQ / DEQ Monitor: Major Name List Option

Once the Major Name List panel is displayed, you can use *L SYSIGGV2* to find if there are any ENQs for catalog resources. Select the Major name SYSIGGV2, see Figure C-6.



```
ENQ/DEQ Monitor - Major Name List     Row 16 to 29 of 46

Enter S to select a Major Name for details     .
      L major on command line to locate a Major.   Elapsed seconds:     732

   Sel.   ----------   -----   ----   -----   ------- -Average-   -Reserved-
   Field  Major Name   Scope   Exit   RNL     Counter   msec       seconds
   s      SYSIGGV2     RES            FORCE      1362       8            10
   _      SYSIKJBC     SYS                         28
   _      SYSIKJPL     SYS                        431
   _      SYSVSAM      SYSS                        14
   _      SYSVTOC      RES            FORCE        96      27             2
   _      SYSZ#SSI     SYS            NO            5
   _     SYSZALCF     SYS                        3
```

Figure C-6 ENQ / DEQ Monitor: locating a major name

After selecting major name SYSIGGV2 a minor name list is displayed, as shown in Figure C-7.

```
ENQ/DEQ Monitor - Minor Name List       Row 1 to 8 of 24

Minor Name list for:              Major Name  : SYSIGGV2
                                  RNL . . . . : FORCED
                                  Scope . . . : RESERVE
                                  Reserved sec: 10       Avg msec: 8

Enter S to select a Minor Name for Jobnames    .
      L min.  on command line to locate a Minor.   Elapsed seconds:      732

  Interval
- --Rate- ----- ------ ------------------------  ------------ Time ------------
S   min.  Count Volume Minor name (max 24 char)  Avg ms  Min ms  Max ms Tot sec
s     0       4 DB2710 CATALOG.DB2710                 6       6       8       0
_     0       3 CICSTS ICFCAT.CICSTS                 24       6      61       0
_     5      56 SMSN01 ICFCAT.IBMBOOKS                6       5      30       0
_   164       1 IMS610 ICFCAT.IMS610                 62      62      62       0
_     1      29 SMS019 ICFCAT.SMSUCAT1                6       6       8       0
_    22     663 SMS011 ICFCAT.SMSUCAT2                8       6     147       5
_    10     257 SMS015 ICFCAT.SMSUCAT3               10       6     121       2
```

Figure C-7 ENQ / DEQ Monitor: Minor Name List

Selecting a minor name will display the job and program names with an indication of exclusive or shared use:

```
ENQ/DEQ Monitor - Jobname List           Row 1 to 1 of 1



List for Major Name  : SYSIGGV2
         Minor Name  : CATALOG.DB2710
         Minor Length: 20
   --------  --------   ----------       -------- ---       ----------
   Job_name  User_ID    Enqs x Job       Pgm_name E/S       Enqs x PGM
   CATALOG   *                   6       IGGPACDV  S                 6
******************************* Bottom of data ********************************
```

Figure C-8 ENQ / DEQ Monitor: Jobname List

From these displays, you can find which catalogs have high activity and how long it takes to process ENQ serialization requests for these catalogs.

In addition to RMF and the ENQ/RESERVE/DEQ Monitor, the `F CATALOG,REPORT,PERFORMANCE` command can provide data concerning the number of entries into the catalog address space, the number of serialization requests CAS has issued, and how long these requests take before being granted. Figure C-9 shows part of the output of the command:

```
F CATALOG,REPORT,PERFORMANCE
 IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
 IEC359I CATALOG PERFORMANCE REPORT 913
 *CAS*************************************************
 *  -----CATALOG EVENT----   --COUNT--  ---AVERAGE---  *
 *  Entries to Catalog        80,789    17.716 MSEC  *
 *  BCS ENQ Shr Sys          105,214     0.223 MSEC  *
 *  BCS ENQ Excl Sys           1,673     0.455 MSEC  *
 *  BCS DEQ                  125,748     0.155 MSEC  *
 *  VVDS RESERVE CI            5,199     1.302 MSEC  *
 *  VVDS DEQ CI                5,199     0.192 MSEC  *
 *  VVDS RESERVE Shr         135,295     0.188 MSEC  *
 *  VVDS RESERVE Excl            392     0.217 MSEC  *
 *  VVDS DEQ                 135,687     0.150 MSEC  *
```

Figure C-9 F CATALOG,REPORT, PERFORMANCE output

There is more information returned from the command but the above data is what is pertinent to diagnosing catalog contention. This report tells you how many entries have been made into CAS, how many exclusive and shared ENQs have been issued to catalogs and VVDSs since the counters were last reset with `F CATALOG,REPORT,PERFORMANCE(RESET)`. If the command has been issued over a period of time, the report reveals workload and ENQ response time trends.

Catalog Contention Summary

After reviewing the RMF reports, output of the `D GRS` and the `F CATALOG` commands, it should be possible to determine if ENQ response times and/or CAS workload have increased. If efforts to tune GRS do not result in lower or acceptable ENQ response times, or if the GRS ring is operating as efficiently as possible and the elongation is due to increased CAS activity for a particular catalog, the only option available is to investigate the possibility of splitting the catalog in question.

You use automation to issue `F CATALOG` commands in interval time, saving the output `in data sets (using Generation Data Group), and then analyze the outputs. To save the command output in a data set you can use two ways:`

1. SDSF batch, see sample in Figure C-10. Refer to *z/OS SDSF Operation an Customization*, SA22-7670 for more information about SDSF batch.

```
//SDSF   EXEC PGM=SDSF,PARM='++60,132'
//ISFOUT DD SYSOUT=A
//OUTPUT DD DSN=...
//ISFIN DD *
ULOG
  /F CATALOG,REPORT,PERFORMANCE
REFRESH
REFRESH
REFRESH
REFRESH
REFRESH
PRINT FILE OUTPUT
PRINT
PRINT CLOSE
```

Figure C-10 F CATALOG Command in SDSF batch

2. For REXX, see samples in Example C-2, Example C-4 on page 440, Example C-5 on page 440, and the JCL in Example C-3 on page 439. You can customize these samples REXX and JCL and use a scheduler, like OPC, to run periodically and collect catalog performance data.

Example: C-2 CATPERF REXX to issue F CATALOG command

```
/*                   REXX                                    */
/* THIS IS A REXX EXEC TO ISSUE F CATALOG,REPORT,PERFORMANCE */
/* COMMANDS AND WRITE THE OUTPUT TO A DATA SET               */
/*                                                           */
/***********************************************************/
'ALLOC F(OUTP) DA(REPORT.OUT) OLD'
SOLD=0
UNSOLD=0
X=MSG("ON")
/**********************************************************/
/*                   REXX                                   */
/* THIS will check CONSPROF profile and enter CONSOLE       */
/* mode to issue F CATALOG,REPORT commands.                 */
/**********************************************************/
X=OUTTRAP('CONDIS.','*')
"CONSPROF"
X=OUTTRAP('OFF')
IF RC > 0 THEN DO
   SAY "ERROR IN CONSPROF."
   EXIT
END
PARSE VAR CONDIS.1 W1 W2 W3 W4 W5
IF SUBSTR(W1,1,3) = 'IKJ' THEN
  DO
```

```
      IF RIGHT(W2,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF SOLDISPLAY(NO)"
          SOLD=1
        END
      IF RIGHT(W4,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(NO)"
          UNSOLD=1
        END
  END
IF SUBSTR(W1,1,3) <> 'IKJ' THEN
  DO
      IF RIGHT(W2,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF SOLDISPLAY(NO)"
          SOLD=1
        END
      IF RIGHT(W4,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(NO)"
          UNSOLD=1
        END
  END
X=MSG('off')
z=0
"CONSOLE ACTIVATE"
ADDRESS CONSOLE "F CATALOG,REPORT,PERFORMANCE"
DO WHILE rc = 0
  rc='GETMSG'(MVSCMD.,,,,5)
    IF rc = 0 THEN DO
    I = 1
      DO       I = 1 TO MVSCMD.0
       IF z = 0 THEN
          DO
             date = mvscmd.mdbgdstp
             time = mvscmd.mdbgtimh
             QUEUE "Date: "||date||"   Time: "||time
           END
       QUEUE mvscmd.i
       z = 1
      END
    END
END
DO i = 1 to queued()
   ADDRESS MVS 'EXECIO 1 DISKW OUTP'
END
ADDRESS MVS 'EXECIO 0 DISKW OUTP (FINIS'
X=MSG('ON')
```

```
'FREE FI(OUTP)'
"CONSOLE DEACTIVATE"
IF SOLD=1 THEN ADDRESS "TSO" "CONSPROF SOLDISPLAY(YES)"
IF UNSOLD=1 THEN ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(YES)"
EXIT
```

Example: C-3 JCL to run REXX in batch and copy the output to a GDG

```
//jobname  JOB jobcard info
//TSOCMD1 EXEC PGM=IKJEFT01,REGION=4096K
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
 EXEC 'userid.CATREP.PERF(CATDATE)'
/*
//IEBGEN1  EXEC PGM=IEBGENER,COND=(1,NE)
//SYSPRINT  DD  SYSOUT=X
//SYSIN     DD  DUMMY
//SYSUT1 DD DSN=userid.REPORT.OUT,DISP=SHR
//SYSUT2 DD DSN=userid.CATREP.OUT(+1),DISP=(NEW,CATLG),
//    LIKE=userid.REPORT.OUT
//TSOCMD2 EXEC PGM=IKJEFT01,REGION=4096K
//SYSTSPRT DD  SYSOUT=*
//SYSUADS  DD  DSN=SYS1.UADS,DISP=SHR
//SYSUADN  DD  DSN=SYS1.UADS,DISP=SHR
/*
//IEBGEN1  EXEC PGM=IEBGENER,COND=(1,NE)
//SYSPRINT  DD  SYSOUT=X
//SYSIN     DD  DUMMY
//SYSUT1 DD DSN=userid.REPORT.OUT,DISP=SHR
//SYSUT2 DD DSN=userid.CATREP.OUT(+1),DISP=(NEW,CATLG),
//    LIKE=userid.REPORT.OUT
//TSOCMD2 EXEC PGM=IKJEFT01,REGION=4096K
//SYSTSPRT DD  SYSOUT=*
//SYSUADS  DD  DSN=SYS1.UADS,DISP=SHR
//SYSUADN  DD  DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD  DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN  DD  *
 EXEC 'userid.CATREP.PERF(CATPERF)'
/*
```

Example: C-4 CATDATE REXX

```
/*    rexx   */
date = date('j')
'ALLOC F(OUTP) DA(REPORT.OUT) SHR'
ADDRESS MVS 'EXECIO * DISKR OUTP (FINIS'
DO x = 1 to queued()
   PARSE PULL record
   IF SUBSTR(record,1,4) = 'Date' THEN DO
     rdate = SUBSTR(record,9,5)
     IF rdate ¬= date THEN CODE=1
   END
END
'FREE FI(OUTP)'
IF CODE = 1 THEN EXIT 1
EXIT
```

Example: C-5 CATDATE REXX

```
/*    rexx   */
date = date('j')
'ALLOC F(OUTP) DA(REPORT.OUT) SHR'
ADDRESS MVS 'EXECIO * DISKR OUTP (FINIS'
DO x = 1 to queued()
   PARSE PULL record
   IF SUBSTR(record,1,4) = 'Date' THEN DO
     rdate = SUBSTR(record,9,5)
     IF rdate ¬= date THEN CODE=1
   END
END
'FREE FI(OUTP)'
IF CODE = 1 THEN EXIT 1
EXIT
```

D

Information APARs

In this appendix we list several information APARs that deal with VSAM data collection and problem determination.

II12927 - Documentation for VSAM problems

```
Item II12927
  APAR Identifier ...... II12927      Last Changed ........ 02/11/14
  INFO APAR TO PROVIDE GUIDELINES FOR DOCUMENTATION NEEDED BY VSAM
  CATALOG, IDCAMS LEVEL2 TO SOLVE COMMON CATEGORIES OF PROBLEMS.

  Symptom ...... DD DOC               Status ........... INTRAN
  Severity ................... 4      Date Closed .........
  Component .......... INFOV2LIB      Duplicate of ........
  Reported Release ......... 001      Fixed Release ............
  Component Name V2 LIB INFO ITE      Special Notice
  Current Target Date ..              Flags
  SCP ...................
  Platform ............

  Status Detail: Not Available

  PE PTF List:

  PTF List:


  Parent APAR:
  Child APAR list:


  ERROR DESCRIPTION:
  This info apar outlines the documentation requirements for VSAM,
  CATALOG, RLS and IDCAMS L2 diagnosis of general problems.
  Specific problem scenarios may require additional documentation.
  This is designed to assist customers in determining what
  documentation to gather before reporting the problem to IBM.
  #############################################

  Questions before the documentation-gathering process:
  1. Did this problem just occur "out of the blue" or did
     something recently change in your environment, such as the
     installation of software or hardware maintenance?
  2. Did the problem occur only one time?
  3. Can you recreate the problem?
  .
  .
  The following are documentation requirements for CATALOG
  problems:

  ABENDS
  - SVC dump of the abend
  - Syslog (at the time of the abend)
  .
  BROKEN DATA SET
      (Please see II08859 for more details)
  .
  - SYSLOG/JOBLOG (including all messages, JCL and commands)
  - EXAMINE (both ITEST and DTEST) output
  - LISTCAT ALL output
  - DIAGNOSE output
  .
  .
  BROKEN CATALOG
  - LISTCAT ENT(catname)ALL CATALOG(catname)
  - DIAGNOSE vvds
    DIAGNOSE bcs
    DIAGNOSE COMPARE vvds to bcs
    DIAGNOSE COMPARE bcs to vvds
  ***Please see Chapter 22 of the DFSMS/MVS
     ACCESS METHOD SERVICES for the INTEGRATED
     CATALOG FACILITY for DIAGNOSE examples***
  - EXAMINE (both ITEST and DTEST)
  - any error messages from joblog
  .
  The following are documentation requirements for VSAM problems:
  .
  PERFORMANCE

  See II10752.
  - provide a clear and adequate description
  - what is not performing properly
  - what is the basis for the performance expectation
  - what changes appear to trigger the performance change
  * maintenance, release, application?


  - dump of CAS (and associated user ASIDs) before any
    recovery actions...if Catalog is involved
  - performance report
  - comparison report (what was performing in the last release)
  *** We will only diagnose if the problem is defect-related ***
  .
  HANG/WAIT/LOOP
  - dump of CAS (and associated ASIDs)...if Catalog is involved
  - dump from all the systems in the sysplex
  .
  The following are DOC requirements for RLS (SMSVSAM) problems:
  .
  1. To dump SMSVSAM ASIDs on all systems at the same time.
  .
```

```
DUMP COMM=(Some meaningful dump title),
Rnn,JOBNAME=(SMSVSAM),CONT
Rnn,SDATA=(GRSQ,RGN,ALLNUC,LPA,LSQA,CSA,PSA,SQA,SUM,SWA,TRT)
Rnn,DSPNAME=('SMSVSAM',*),
Rnn,REMOTE=(SYSLIST=*('SMSVSAM'),DSPNAME,SDATA),END

2. This will dump SMSVSAM and XCF on the local system and dump
     SMSVSAM XCF w/ DSPNAME and SDATA on all the systems in the
  sysplex.
  DUMP COMM=(SMSVSAM and XCF)
  RXX,JOBNAME=(SMSVSAM,XCFAS),DSPNAME=('XCFAS'.*,'SMSVSAM'.*),CONT
  RYY,SDATA=(GRSQ,RGN,ALLNUC,LPA,LSQA,CSA,PSA,SQA,SUM,SWA,TRT), CONT
  R ZZ,REMOTE=(SYSLIST=(*,('XCFAS','SMSVSAM',DSPNAME,SDATA)))
  .
  To DUMP the SMSVSAM asid, SMSVSAM data spaces and CICS regions
  involved.
  .
  DUMP COMM=(CICS & SMSVSAM HANG)
  R nn,JOBNAME=(SMSVSAM, CICS1, CICS2, CICS3, ETC),
  R nn,DSPNAME=('SMSVSAM'.*), END
  R nn,SDATA=(PSA,NUC,SQA,LSQA,SUM,RGN,GRSQ,LPA,TRT,CSA),CONT
  R nn,REMOTE=(SYSLIST=*('SMSVSAM'),DSPNAME,SDATA),END
  where CICS1, CICS2, CICS3, ETC are the names of the CICS regions.
  .
  The following are documentation requirements for IDCAMS problems:
  .
  - Syslog (including all messages, JCL and commands) from the batch job that
received the OPEN error
  - VERIFY output
  - LISTCAT ALL output
  - EXAMINE (ITEST and DTEST)
  - DIAGNOSE
```

II13326 - Common problems with SHCDS

```
Item II13326
  APAR Identifier ...... II13326      Last Changed ........ 02/11/04
  COMMON PROBLEMS FROM SHCDS DEFINITION AND USAGE. PLEASE REVIEW
  THIS INFO APAR IF RECEIVE ABEND0F4 RSN673F0614 673F0614.

  Symptom ...... IN INCORROUT         Status ........... INTRAN
  Severity .................. 4       Date Closed .........
  Component .......... INFOV2LIB      Duplicate of ........
  Reported Release ......... 001      Fixed Release ............
  Component Name V2 LIB INFO ITE      Special Notice
  Current Target Date ..              Flags
```

```
SCP ..................
Platform ...........

Status Detail: Not Available

PE PTF List:

PTF List:


Parent APAR:
Child APAR list:


ERROR DESCRIPTION:
Due to a number of problems we've been seeing in the field
from new customers, we decide to put together this informational
APAR to help guide you through the process of setting up and
trouble-shooting problems with SMSVSAM/RLS shared control
dataset (SHCDS).
.
PART 1: SHCDS DEFINITION
.
Here is a checklist that you need to run through to make sure
you have defined your SHCDS properly:
1- See 14.1.6  Defining Sharing Control Data Sets in DFSMSdfp
   Storage Admin. Reference for details. Also see section
   14.1.10  Establishing Authorization for VSAM RLS in DFSMSdfp
   Storage Administration Reference.
2- SHCDS must be a VSAM Linear data set.
3- CISIZE for SHCDS must be 4096.  Make sure that if you are
   using a Dataclass you are getting 4096.
4- Shareoptions must be (3,3).
5- Secondary extents are strongly recommended.
6- Initial size of the SHCDS needs to be large enough.
7- When defined, the SHCDS does not need to be catalogued on
   all systems in the sysplex. If it is cataloged, it must be in
   a catalog available when SMSVSAM initializes. The user
   may issue the command, Vary SMS,SHCDS(dsname),NEW
   and Vary SMS,SHCDS(dsname),NEWSPARE from any
   system in the sysplex, not just from the system where the
   SHCDS was defined and catalogued originally.  SMSVSAM
   will attempt to re-catalog the SHCDS if it cannot find the
   SHCDS in a catalog on that system.
8- SMSVSAM must be authorized to update SYS1.DFPSHCDS.*
   data sets.
   If you protect SYS1.* data sets be sure SMSVSAM is
   able to access  SYS1.DFPSHCDS.* for update.
9- Follow the SHCDS naming convention.  The SHCDS name must
```

```
   match the volume that it resides on.  That is,
   SYS1.DFPSHCDS.firstnam.Vvolser resides on volume volser
Note:  APAR OW49746 prevents addition of a SHCDS if the SHCDS is
       not linear or if the CISIZE is not 4096.
.
The SHCDS may be defined using IDCAMS.  Here is an example:
//*---------------------------------------------------
//* ALLOCATE ON XP0301 - GUARANTEED SPACE IS SXPXXS04
//*---------------------------------------------------
//ALLOCLD1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.ACTIVE3.VXP0301) LINEAR -
        STORCLAS(SXPXXS04)                               -
        SHAREOPTIONS(3 3)   CYL(20 20) VOLUME(XP0301) )
/*
.
PART 2: COMMON USAGE PROBLEMS
.
For suspected problems with the SHCDS, look for messages


starting with IGW6, such as IGW615I.  Also look for messages
that indicate a security problem with your SHCDS.
.
A- Problem:  Sharing Control Data Sets are added, but when
   when SMSVSAM is recycled or the system is IPLed, the SHCDS
   are deleted during initialization.
   Possible solutions:
   1- SHCDS is not correctly defined.  See "SHCDS Definition"
      above.  Redefine SHCDS.
   2- SMSVSAM does not have proper access to the SHCDS.
      Examine the Syslog for error messages indicating problems
      with the SYS1.DFP.  Set up access properly.
B- Problem:  SMSVSAM initialization is not proceeding.
   Use the command, D SMS,SMSVSAM.  If the response indicates
   that SMSVSAM is in SHC_Ph2_Init:
.
IGW420I DISPLAY SMS,SMSVSAM
DISPLAY SMS,SMSVSAM - SERVER STATUS
SYSNAME:  SYSTEM2  UNAVAILABLE ASID: 00FC STEP: SHC_Ph2_Init
.
You should have received some IGW6* messages, such as
IGW611A or IGW609A, prior to this point.  Look for these
messages in the Syslog. Issue the command, D SMS,SHCDS.
Examine the "Status" column.   Make sure that you have at least
2 active SHCDS and 1 Spare SHCDS.  A common error is to forget
to add a Spare SHCDS or to add three actives instead of
2 actives and a spare.
```

.
PART 3: COMMON TASKS
.
You may want to swap in a new set of SHCDS, because you want to increase the size of the SHCDS or you want to change the volume where the SHCDS resides. You must use the VARY SMS,SHCDS command when making changes to the SHCDS in order for SMSVSAM to know about the SHCDS. Do not delete an active or spare SHCDS even if SMSVSAM is not active. Do not move a SHCDS from one volume to another, because the SHCDS naming convention depends on the volume to match the SHCDS name.
.
NOTE: Do not delete and redefine an active or spare SHCDS without first deleting the SHCDS from SMSVSAM using the command - V SMS,SHCDS(shcdsname),DELETE. SMSVSAM remembers the SHCDSs from one SMSVSAM recycle to the next. You must tell SMSVSAM that the SHCDS is no longer an active or spare SHCDS.
.
A common error is to delete and redefine the SHCDS without deleting the SHCDS from SMSVSAM - for example, when SMSVSAM is not active. The correct way to communicate changes for the SHCDS is through the command - V SMS,SHCDS. Once you have deleted a SHCDS using the command V SMS,SHCDS(shcdsname),DELETE, you can then safely use IDCAMS DELETE and DEFINE to modify your SHCDS.
.
NOTE: Do not move a SHCDS from one volume to another, because the SHCDS naming convention depends on the volume to match the SHCDS name.
Remember that once you've added 2 Active SHCDS and 1 Spare SHCDS you will not be allowed delete them when using the command, Vary SMS,SHCDS(dsname),delete. To make a swap, you must add the new SHCDS first and delete the old SHCDS to make sure you stay within criteria. For example, if you have:
.
```
16.40.09 SYSTEM1           d sms,shcds
IGW612I 16:40:10    DISPLAY SMS,SHCDS
Name                      Size    %UTIL Status  Type
TOOSMALL.VXP0301        7200Kb     5%  GOOD    ACTIVE
TOOSMALL.VXP0302        7200Kb     5%  GOOD    ACTIVE
WRONGVOL.VXP0201        7200Kb     5%  GOOD    SPARE
```
.
Issue these commands:
```
V SMS,SHCDS(JUSTRITE.VXP0301),NEW
V SMS,SHCDS(JUSTRITE.VXP0302),NEW
V SMS,SHCDS(RIGHTVOL.VXP0202),NEWSPARE
```

```
.
Now you have:
SYSTEM1           d sms,shcds
IGW612I 16:45:49    DISPLAY SMS,SHCDS
Name                      Size    %UTIL Status  Type
TOOSMALL.VXP0301         7200Kb     5%  GOOD    ACTIVE
TOOSMALL.VXP0302         7200Kb     5%  GOOD    ACTIVE
JUSTRITE.VXP0301        14400Kb     2%  GOOD    ACTIVE
JUSTRITE.VXP0302        14400Kb     2%  GOOD    ACTIVE
WRONGVOL.VXP0201         7200Kb     5%  GOOD    SPARE
RIGHTVOL.VXP0202         7200Kb     5%  GOOD    SPARE
.
Then you can issue:
V SMS,SHCDS(TOOSMALL.VXP0301),DELETE
V SMS,SHCDS(TOOSMALL.VXP0302),DELETE
V SMS,SHCDS(WRONGVOL.VXP0201),DELETE
.
Which will leave you with:
SYSTEM1           d sms,shcds
IGW612I 16:47:07    DISPLAY SMS,SHCDS
Name                      Size    %UTIL Status  Type
JUSTRITE.VXP0301        14400Kb     2%  GOOD    ACTIVE
JUSTRITE.VXP0302        14400Kb     2%  GOOD    ACTIVE
RIGHTVOL.VXP0202         7200Kb     5%  GOOD    SPARE
See 4.56.12  Changing the State of Coupling Facility Cache
Structures and Volumes in MVS System Commands for information
on the VARY command.
.
PART 4: SYMTOMS OF HAVING PROBLEMS WITH SHCDS DEFINITION/USAGE
.
The following symptoms result from this set of steps.
Please note that this is NOT a valid way to change the
way the SHCDSs are defined.  See the section above for the
recommended way of redefining your SHCDS.
- Terminate SMSVSAM
- Delete all SHCDSs (could also happen if delete some SHCDSs)


- Redefine SHCDSs
- Bring up SMSVSAM
.
DUMP00 TITLE=COMPID=DF122,CSECT=IGWXSS90+0890,DATE=10/13/01,
         MAINTID= NONE   ,ABND=0F4,RC=00000024,RSN=67260989
.
DUMP01 TITLE=COMPID=DF122,CSECT=IGWXSS91+0628,DATE=10/13/01,
        MAINTID= NONE   ,ABND=0F4,RC=00000024,RSN=67610382
RSN67610382 67610382
.
--------------------------------------------------------------
```

```
SHCDS has CISIZE=6K and is added successfully.
Things can go along fine for a while and then you get :
DUMP00 TITLE=COMPON=MEDIA MANAGER, COMPID=DF106, ISSUER=ICYFRR
    DUMP TAKEN TIME=15.06.59 DATE=05/31/2002
    SYSTEM ABEND CODE=0C4  REASON CODE=00301314
    MODULE=IEANUC01 CSECT=ICYSTOR
--------------------------------------------------------------
Non-linear SHCDS - defined as a PS or whatever is default:
On system2:
15.46.35 SYSTEM2   IGW602E ADD SHARE CONTROL DATA SET FAILED,
SYS1.DFPSHCDS.NONLINNC.VSPLXP2 IS NOT A VSAM LINEAR DATA SET
On system1:
15.46.35 SYSTEM2   IGW602E ADD SHARE CONTROL DATA SET FAILED,
SYS1.DFPSHCDS.NONLINNC.VSPLXP2 IS NOT A VSAM LINEAR DATA SET
.
then later:
DUMP00 TITLE=COMPON=MEDIA MANAGER, COMPID=DF106, ISSUER=ICYFRR
         SYSTEM ABEND CODE=0C4  REASON CODE=00301314
         MODULE=IEANUC01 CSECT=ICYSTOR
--------------------------------------------------------------
During Server initialization, insufficient access authority
to SHCDS. Previously defined SHCDS are deleted.
ICH408I JOB(SMSVSAM ) STEP(SMSVSAM ) 744
  SYS1.DFPSHCDS.ACTIVE2.VSPLXPK CL(DATASET ) VOL(USRPAK)
  INSUFFICIENT ACCESS AUTHORITY
  FROM SYS1.DFPSHCDS.* (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
IEF196I IEC161I 040(056,006,IGG0CLFT)-002,IEESYSAS,SMSVSAM,SYS00
IEC161I 040(056,006,IGG0CLFT)-002,IEESYSAS,SMSVSAM,SYS00001,,,
IEF196I IEC161I SYS1.DFPSHCDS.ACTIVE2.VSPLXPK
IEC161I SYS1.DFPSHCDS.ACTIVE2.VSPLXPK
.
IEA794I SVC DUMP HAS CAPTURED: 760
DUMPID=001 REQUESTED BY JOB (SMSVSAM )
DUMP TITLE=COMPID=DF122,CSECT=IGWXSI20+0490,DATE=04/16/00,MAINT
         ID= NONE   ,ABND=0F4,RC=00000024,RSN=67510404
RSN67510404 67510404
.
*IGW611A SHARE CONTROL DATA SET NEVER ASSIGNED
*IGW609A NO SPARE SHARE CONTROL DATA SETS EXIST. IMMEDIATE ACTIO
 REQUIRED
----------------------------------------------------------------
During Server initialization, insufficient access to SHCDS
During initialization on a new system, a previously defined


SHCDS might be deleted.
With OW49746, we will fail the initialization process.
IEF196I IEF237I 081F ALLOCATED TO SYS00002
```

```
IEF196I ICH408I JOB(IEESYSAS) STEP(SMSVSAM )
IEF196I   SYS1.MVSRES.MASTCAT CL(DATASET ) VOL(USRPAK)
IEF196I   INSUFFICIENT ACCESS AUTHORITY
IEF196I   ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
ICH408I JOB(IEESYSAS) STEP(SMSVSAM ) 785
  SYS1.MVSRES.MASTCAT CL(DATASET ) VOL(USRPAK)
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
IGW601E ADD SHARE CONTROL DATA SET FAILED, 790
UNABLE TO ALLOCATE SYS1.DFPSHCDS.ACTIVE.VSPLXPK
----------------------------------------------------------------
During initialization on a new system, a previously defined
SHCDS might be deleted.
Instead, we will fail the initialization process.
IGW619I ACTIVE SHARE CONTROL DATA SET 949
SYS1.DFPSHCDS.ACTIVE2.VSPLXPK ADDED.
IEF196I IEF237I 081F ALLOCATED TO SYS00002
IGW619I ACTIVE SHARE CONTROL DATA SET 951
SYS1.DFPSHCDS.ACTIVE.VSPLXPK ADDED.
IGW601E ADD SHARE CONTROL DATA SET FAILED, 952
UNABLE TO ALLOCATE SYS1.DFPSHCDS.SPARE.VXP0201
IEF196I IEF237I 081F ALLOCATED TO SYS00003
IGW619I SPARE SHARE CONTROL DATA SET 954
SYS1.DFPSHCDS.SPARE.VSPLXPK ADDED.
----------------------------------------------------------------
During initialization on a new system, a SHCDS that is on an
offline volume, will be deleted.
14.33.02 SYSTEM1          *IGW615W SHARE CONTROL DATA SET
 SYS1.DFPSHCDS.ACTIVE4.VXP0302 HAS FAILED
.
PART 5: FALLBACK PROCEDURE
.
For some SHCDS errors, FALLBACK is the only way to correct the
problem and to get the SMSVSAM to intialize successfully again.
If you get repeated abends with the prefix '67' in the RSN code,
then you know it's time to do a FALLBACK.
.
NOTE: *** IPL would NOT help because SHCDS is remembered from
          one IPL to another ***
.
FALLBACK procedure is documented in z/OS V1R3.0 DFSMSdfp Storage
Administration Reference. In a nutshell, FALLBACK will reformat
the SHCDS and hence clear all the errors.
FALLBACK involves:
  - Terminate all servers in the plex using the command
    "VARY SMS,SMSVSAM,TERMINATESERVER"
  - Issue "VARY SMS,SMSVSAM,FALLBACK"
  - Reply to the WTOR that you are sure you want to do FALLBACK
  - Reactivate the server one at a time using -
```

```
      "VARY SMS,SMSVSAM,ACTIVE"
      One the first system, you will be asked to specify 2 ACTIVE
      and 1 SPARE SHCDS.


      To add an active SHCDS: "VARY SMS,SHCDS(SHCDS_name),NEW"
      To add a spare SHCDS: "VARY SMS,SHCDS(SHCDS_name),NEWSPARE"
  .
  So far, we know the following errors would require FALLBACK:
    - Abend0F4 RC24 RSN675D0355
    - Abend0F4 RC24 RSN67260989
    - Abend0F4 RC24 RSN675F0398


  LOCAL FIX:
```

BDC000010564

```
Item BDC000010564
Source..........: PDDBO  PDDBO
Last updated....: 10/26/1998
Abstract........: VSAMRLS MIXED ENVIRONMENT

USERS: VRLs

PROBLEM SUMMARY:
 SMSVSAM address space hangs.   Vary S active also hangs.   Display of
server shows SHV PH2 Init step

SOLUTION:
 One of the systems was missing catalog entry for the SHCDS shared
control datasets.

PROBLEM DETAILS:
 TYPE: USER
 COMPID: 5695DF122
 RELEASE: 1D0

CUSTOMER:
We are using VSAM RLS across two systems.  One is 5.2.2 (SMS 1.2) the
other was just upgraded from OS/390 1.3 (SMS 1.3) to OS/390 2.5 (SMS
1.4).  We cannot get SMSVSAM to come available.  I forced the SMSVSAM
```

```
address space down on the 5.2.2 system to try to recover it, but this
did not help.

CUSTOMER:
We tried applying UW48355 to our 5.2.2 system to see if that would
help, to no avail.

IBM STATUS:
I left a voice message requesting that the pmr be updated with additi-
onal info.
1.  Does the RLSINIT parm specify YES
2.  What is being displayed from the D SMS,SMSVSAM,ALL
3.  Has RLS ever been intialized before on these systems?
4.  Were there any messages or Abends
5.  How are the SHCDS defined (make sure there are 2 active and 1 spare)
6.  Does Vary SMS,SMSVSAM,ACTIVE abend/message or hangs.........
          Please update the pmr with as much info as possible.

CUSTOMER:
As I stated earlier, these are NOT new systems, just that one was upg-
raded, so all the definitions are still there.  The parmlib members
were copied over.

 D SMS,SMSVSAM,ALL

 IGW420I DISPLAY SMS,SMSVSAM 934

 DISPLAY SMS,SMSVSAM - SERVER STATUS

   SYSNAME:  ESAJ     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init

   SYSNAME:  SYSI     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

 DISPLAY SMSVSAM     - JOB STATUS

   SUBSYSTEMS CONNECTED:        0 BATCH:        0
 DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)

   CONNECT STATUS:

     SYSNAME:  ESAJ     ................. RSN: 00000000 RbldNotActive

     SYSNAME:  SYSI     ................. RSN: 00000000 RbldNotActive

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

This is what we get when we do a display.  We receive no error message
on doing the V SMS,SMSVSAM,ACTIVE.

IBM STATUS:
From the status of each status it appears that the step phase is stuck
in SHC_PH2_Init which is dealing with the SHCDS (Share Control
Datasets).
1. How are the SHCDS defined (esp shareoptions)
2. Please issue D SMS,SHCDS and paste text in pmr.
3. Is this the first time that the server has been re-activated since
   it has been up on both systems?

CUSTOMER:
Yes, I believe that this has not worked since the upgrade.  Here's the
output:

 D SMS,SMSVSAM,ALL

 IGW420I DISPLAY SMS,SMSVSAM 934

 DISPLAY SMS,SMSVSAM - SERVER STATUS

   SYSNAME:  ESAJ     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init

   SYSNAME:  SYSI     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................
   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

   SYSNAME:  ........ ........... ASID: .... STEP: ....................

 DISPLAY SMSVSAM     - JOB STATUS

   SUBSYSTEMS CONNECTED:        0 BATCH:        0

 DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)

   CONNECT STATUS:

     SYSNAME:  ESAJ     ................. RSN: 00000000 RbldNotActive

     SYSNAME:  SYSI     ................. RSN: 00000000 RbldNotActive

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

     SYSNAME:  ........ ................. RSN: ........ ................

CUSTOMER:
Sorry, I pasted the wrong stuff:

 D SMS,SHCDS
 IEE932I 692
 IGW612I 14:26:23    DISPLAY SMS,SHCDS
 Name             Size   %UTIL Status  Type
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
 -----------------0Kb  0%    N/A     N/A
```

IBM STATUS:
In the D SMS,SHCDS output it is not showing the names of your Share control datasets. Did you erase them from the output or is this the output pasted from the console? If so, it appears that there are no SHCDS listed for this system. Can you see them from a listcat?

CUSTOMER:
Since one of the systems was not upgraded, are these defined in both sides?

IBM STATUS:
There should be only 1 set of Share control datasets. It sounds as if the XCF strucutre was deleted and redefined. This is one of the ways that the names of the SHCDS files are dropped. Also I noticed in your first update that you have an DFSMS R120 MVS 5.2.2 system. Maybe this was a typo, but RLS cannot be activated on a DFSMS R1B0 system.

CUSTOMER:
That DFSMS is R130 on MVS 5.2.2.

CUSTOMER:
We recataloged the datasets. Tried 'cycling' SMSVSAM to no avail. IPLd both systems last night. The user tried one immediately and all is well, the other I'm still waiting on an answer. So looks as though the datasets got uncataloged on the conversion. Of course no one knows how. Is there a way to decipher what the error codes mean so that the next time something like this happens we can find it easier?

IBM STATUS:
The reason codes are not documented but a clue is when the steps of the D SMS,SMSVSAM,ALL show SHC_PH2_INIT or anything with SHC means that the share control datasets were not recognized.

CUSTOMER:
I displayed the shcds's and show three (3) active shcds's. When I do the display on smsvsam,all, i show one active and one unavailable sys-tem. The unavailable system is in the shc_ph2_init state. I varied SMSVSAM active (v sms,smsvsam,active) and the status did not change, i.e. one available and one unavailable. My commands were issued on the unavailable system. Do you have any further suggestions on getting it to activate? I noticed Monday am following IPL's of both systems over the weekend that both showed active, but somehow one of the sys-tems evidently fell out. The SHCDS datasets are all cataloged on that system and add successfully when added via command.

IBM STATUS:
Can you paste the output from D SMS,SHCDS issued on the unavailable system in the PMR?

```
CUSTOMER:
 98244 10:52:23.16 CSTMXE4  00000290  D SMS,SHCDS
98244 10:52:23.22          01000090  IEE932I 264
                       264 00000090  IGW612I 10:52:23    DISPLAY
SMS,SHCDS
                       264 00000090  Name             Size   %UTIL
Status  Type
                       264 00000090  ACTIVE.VAFXBDD   2880Kb  8%  GOOD
  ACTIVE
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A
                       264 00000090  -----------------0Kb  0%    N/A
 N/A  D SMS,SMSVSAM,ALL
98244 10:57:08.14 STC01761 00000090  SQAMON - SQA SIZE -  6756K; SPILL=
 5140K
98244 10:57:08.95          01000290  IXL014I IXLCONN REQUEST FOR
STRUCTURE IGWLO
                       503 00000090  JOBNAME: SMSVSAM ASID: 000A
CONNECTOR NAME:
                       503 00000090  CFNAME: CFPART01
98244 10:57:08.95          01000290  IXL030I CONNECTOR STATISTICS FOR
LOCK STRUC
                       504 00000090  CONNECTOR SYSI:
                       504 00000090        000200F6
                       504 00000090        00000000 00000000 00000000
00000000
                       504 00000090        00000000 00000000 00000000
00000000
                       504 00000090        00000000 00000000 00000000
00000000
                       504 00000090        00000001 00000000 00000000
00000000
                       504 00000090        00000000 00000000 00000000
00000000
```

```
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000002 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
98244 10:57:08.95          01000290  IXL031I CONNECTOR CLEANUP FOR LOCK
STRUCTUR
                       505 00000090  CONNECTOR SYSI, HAS COMPLETED.  D
SMS,SMSVSAM,ALL
98244 10:57:08.14 STC01761 00000090  SQAMON - SQA SIZE -  6756K; SPILL=
 5140K
98244 10:57:08.95          01000290  IXL014I IXLCONN REQUEST FOR
STRUCTURE IGWLO
                       503 00000090  JOBNAME: SMSVSAM ASID: 000A
CONNECTOR NAME:
                       503 00000090  CFNAME: CFPART01
98244 10:57:08.95          01000290  IXL030I CONNECTOR STATISTICS FOR
LOCK STRUC
                       504 00000090  CONNECTOR SYSI:
                       504 00000090       000200F6
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000001 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000002 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
                       504 00000090       00000000 00000000 00000000
00000000
98244 10:57:08.95          01000290  IXL031I CONNECTOR CLEANUP FOR LOCK
STRUCTUR
                       505 00000090  CONNECTOR SYSI, HAS COMPLETED.
00200F6 00000000 00000000 00000000 00000000 00000000
0090  IGW420I DISPLAY SMS,SMSVSAM 506
0090  DISPLAY SMS,SMSVSAM - SERVER STATUS
0090    SYSNAME:  ESAJ      AVAILABLE ASID: 000A STEP:
SmsVsamInitComplete
```

```
0090    SYSNAME:  SYSI     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090    SYSNAME:  ........ ........... ASID: .... STEP:
....................
0090
0090  DISPLAY SMSVSAM     - JOB STATUS
0090    SUBSYSTEMS CONNECTED:       0 BATCH:       1
0090
0090  DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)
0090    CONNECT STATUS:
0090      SYSNAME:  ESAJ     ACTIVE           RSN: 02010407
RbldNotActive
0090      SYSNAME:  SYSI     ................. RSN: 00000000
RbldNotActive

When SYSI fails to connect, like this display, it's always in the
SHC_PH2_INIT status. Attempting to v sms,smsvsam,active has no positve
results.  If an IPL takes place, it will normally sync up.  One other
note: At IPL when it doesn't sync up, we get the message that the
SHCDS is not duplexed; immediate action is required.  Adding another
or tow SHCDS's does not change the status, i.e. it does not activate.

IBM STATUS:
It appears from the output from the D SMS,SHCDS display that only 1
SHCDS exist.  Now this is strange when there should be at least 3
share control datasets(2 active and 1 spare).  As for system SYSI it
appears to be stuck in share control phase init which means that the
Share control datasets are not being recognized to that system.

Do both systems diplay the same ouput for the D SMS,SHCDS?  If not,
please update etr with output for both systems.  Are there 3 SHCDS de-
fined?

CUSTOMER:
IBM:
    Getting back to problem.
Today I issued some SMS display commands and one of the two systems
showed available and one showed unavailable.  I display the SHCDS's
and only one showed active.  I added a second (duplex) and then a
third (spare) which then caused smsvsam to activate and now shows to
```

```
be available on both systems.  I have a user currently testing and
should have results fairly soon.  How does SMS acquire the SHCDS data-
sets?  Since the name is SYS1.DFPSHCDS, does it look for those names?
Once I activate the SHCDS's, do I need to activate them each time
SMSVSAM is stopped (i.e. IPL)?

IBM UPDATE:
The short answers to your question is-
Yes,SMS does look for the data set names.
No,you do not need to activate the Share Control Data Sets every time
a SMSVSAM is stopped.

After talking to Renee it seems that something keeps taking your SC
data sets out of commission and that you're trying to pinpoint why-
true?

IBM STATUS:
One of the systems was missing Catalog entry for shared D/S.
```

BDC000007033

```
Item BDC000007033
Source..........: PDDBO  PDDBO
Last updated....: 02/26/1998
Abstract........: SMSVSAM address space does not complete initialization

USERS: VRLS

PROBLEM SUMMARY:
Delete / Moved SHCDS share control ds SMSVSAM would not initialize

SOLUTION:
Applied uw42626 and issued FALLBACKSMSVSAMYES

PROBLEM DETAILS:
 TYPE: N/A
 COMPID: 5695DF122
 RELEASE: 1C0

CUSTOMER:
Running OS390 R3 at 9710 put level.  Any attempt to bring up
the SMSVSAM address space and it never seems to complete initializa-
tion.  Display it you get the following:
DISPLAY SMS,SMSVSAM - SERVER STATUS
  SYSNAME:  TLP1     UNAVAILABLE ASID: 000A STEP: SHC_Ph2_Init
  SYSNAME:  ........ ........... ASID: .... STEP: ............
```

```
DISPLAY SMSVSAM     - JOB STATUS
  SUBSYSTEMS CONNECTED:        0 BATCH:        0

DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)
  CONNECT STATUS:
    SYSNAME:  TLP1     ................. RSN: 00000000 RbldNotActive
    SYSNAME:  ........ ................. RSN: ........ .............
Here are the SMS parms.

IGD031I SMS PARAMETERS 483
ACDS     = SCS5.DFSMS.ACDS
COMMDS   = SCS5.DFSMS.COMMDS
INTERVAL = 15     DINTERVAL = 150
BMFTIME = 3600    CACHETIME = 3600
SMF_TIME = YES    CF_TIME = 3600
LOCAL_DEADLOCK = 30    GLOBAL_DEADLOCK = 10
REVERIFY = NO     ACSDEFAULTS = NO
DSNTYPE = PDS     USE_RESOWNER = YES
PDSESHARING = NORMAL
OVRD_EXPDT = NO   RLS_MAX_POOL_SIZE = 50MB
SYSTEMS    = 8    RLSINIT = YES
HSP_SIZE   = 256MB
TRACE    = OFF    SIZE = 128K     TYPE = ERROR
  JOBNAME = *    ASID = *
  TRACING EVENTS:
    MODULE = ON   SMSSJF = ON   SMSSSI = ON   ACSINT = ON
    OPCMD  = ON   CONFC  = ON   CDSC   = ON   CONFS  = ON
    MSG    = ON   ERR    = ON   CONFR  = ON   CONFA  = ON
    ACSPRO = ON   IDAX   = ON   DISP   = ON   CATG   = ON
    VOLREF = ON   SCHEDP = ON   SCHEDS = ON   VTOCL  = ON
    VTOCD  = ON   VTOCR  = ON   VTOCC  = ON   VTOCA  = ON
    RCD    = ON   DCF    = ON   DPN    = ON   TVR    = ON
    DSTACK = ON

Used the command: V SMS,SMSVSAM,ACTIVE

IBM STATUS:
When the Vary SMS,SMSVSAM,ACTIVE command is issued what are the messages
received?  Issue a Display SMS,SHCDS this will
tell if any Share control datasets were defined and what is the sta-
tus.  Are there 2 primary and 1 spare SHCDS?

CUSTOMER:
System was IPLed this morning and the SMSVSAM address space is up
and is getting CPU time.  No messages seen.
A D SMS,SHCDS and it never comes back.  Looking through the console
log at ipl time it does show the share datasets:
IEF196I IEF237I 09BB ALLOCATED TO SYS00001
IGW619I ACTIVE SHARE CONTROL DATA SET 162
```

```
SYS1.DFPSHCDS.PRIMARY.VSYSP16 ADDED.
IEF196I IEF237I 09BC ALLOCATED TO SYS00002
IGW619I ACTIVE SHARE CONTROL DATA SET 164
SYS1.DFPSHCDS.SECONDRY.VSYSP17 ADDED.
IEF196I IEF237I 09BD ALLOCATED TO SYS00003
IGW619I ACTIVE SHARE CONTROL DATA SET 166
SYS1.DFPSHCDS.SPARE.VSYSP18 ADDED.

A V SMS,SMSVSAM,ACTIVE and nothing comes back.
A D SMS,SMSVSAM,ALL and no messages or displays are shown .

CUSTOMER:
Just to give you a little more background.  Previously the SMSVSAM
address space use to come up and get Abend0e0, but would never fully
initialize.  However we always could do various displays with no prob-
lem.  This changed to our current problem after we deleted our coup-
ling facility datasets and reallocated them.  We changed the MAXSYSTEM
parm from 8 to 4, that's why we did it.  This morning we wanted to
changeour share datasets because we want to move them.  I turned
RLSINIT to NO abd then did FORCE SMSVSAM,ARM to shutdown theSMSVSAM
address space and this all worked.  I turned RLSINIT to YES and then
did V SMS, SMSVSAM, ACTIVE. SMSVSAM took the abend0e0:
SYSTEM COMPLETION CODE=0E0  REASON CODE=00000030
 TIME=09.29.25  SEQ=00135  CPU=0041  ASID=0020
 PSW AT TIME OF ERROR  075C1000   81E76252  ILC 4  INTC 30
   NO ACTIVE MODULE FOUND
   NAME=UNKNOWN
   DATA AT PSW  01E7624C - 47F0F006  063AB240  00E05180
   GPR  0-3  00000200  7F148470  FF147028  7F147028
   GPR  4-7  FF147028  00000000  00000050  007DE920
   GPR  8-11 81E7625C  00000050  7F148420  01EA079F
   GPR 12-15 20000000  00000000  81E768C8  01E7624C
 END OF SYMPTOM DUMP
It took 2 of these abends.  The SMSVSAM address space shows up and
getting CPU time.  We do not see any other messages.  I then try to
add one of the new share datasets: V SMS,SHCDS(PRIMARY.VSCSPX1),NEW.
Nothing comes back and no matter what displays I put in they don't
come back.  This is where we stand.

IBM STATUS:
From the indication of your messages It is because you do not have a
SPARE Share control dataset recognized.  The msgs following would in-
dicate that a SPARE exist;
IGW619I ACTIVE SHARE CONTROL DATA SET 060
===========>SYS1.DFPSHCDS.prime.V603RES ADDED.
IEF196I IEF237I 023D ALLOCATED TO SYS00002
IGW619I ACTIVE SHARE CONTROL DATA SET 062
===========>SYS1.DFPSHCDS.SECON.V603LIB ADDED.
IEF196I IEF237I 0230 ALLOCATED TO SYS00003
```

```
******>IGW619I SPARE SHARE CONTROL DATA SET 064
===========>SYS1.DFPSHCDS.SPARE.V603RES ADDED.
```

It will indicate SPARE. The messages in your previous update display ACTIVE. So what needs to be done is that one of the SHCDS needs to be deactivated and reactivated as a SPARE with the Vary command using the Newspare keyword...

CUSTOMER:
When we shutdown SMSVSAM this morning and then came back and got the Abend0E0, here are the messages received:

```
IGW619I ACTIVE SHARE CONTROL DATA SET 734
SYS1.DFPSHCDS.PRIMARY.VSYSP16 ADDED.
IEF196I IEF237I 09BC ALLOCATED TO SYS00002
IGW619I ACTIVE SHARE CONTROL DATA SET 736
SYS1.DFPSHCDS.SECONDRY.VSYSP17 ADDED.
IEF196I IEF237I 09BD ALLOCATED TO SYS00003
IGW619I SPARE SHARE CONTROL DATA SET 738
SYS1.DFPSHCDS.SPARE.VSYSP18 ADDED.
IEF196I IEA995I SYMPTOM DUMP OUTPUT
IEF196I SYSTEM COMPLETION CODE=0E0  REASON CODE=00000030
```

So the system does recognize the spare and looking back at ipl time it also shows the SPARE message, my cut and paste probably is bad.

IBM UPDATE:
I discussed this problem with my developer and he can see from your update a couple of things.

1. Need to make sure that the PTFs UW42626 and UW39292 are both applied.
2. Also the way that the SHCDS were moved was incorrect. Due to the info that is kept in the SHCDS and on the system, SMSVSAM will look for those same SHCDS. The proper way would be to add in the new dataset and spares (using Vary command) and then delete the old ones(using Vary Command). Do all of this while the SMSVSAM server is active.

Please verify that these PTFs are applied.

CUSTOMER:
PTFs, UW42626 and UW39292 are on. We have used the VARY command for the SHCDSes. We have just ipled our system with RLSINIT(YES), so SMSVSAM was active. I did a D SMS,SHCDS and the display never came back. I then reset RLSINIT(NO) and did a FORCE SMSVSAM,ARM and this shutdown the address space. I then did a VARY SMS,SMSVSAM,FALLBACK and replied to the message FALLBACKSMSVSAMYES. I then received the following:

```
 IGW524I SMSVSAM FALLBACK PROCESSING IS NOW COMPLETE
 IEF196I IGW523A SMSVSAM ADDRESS SPACE FALLBACK IS REQUESTED.   REPLY
 IEF196I 'CANCEL' TO ABORT, 'FALLBACKSMSVSAMYES' TO PROCEED.
*19 IGW523A SMSVSAM ADDRESS SPACE FALLBACK IS REQUESTED.  REPLY 'CANCEL'
```

```
  TO ABORT, 'FALLBACKSMSVSAMYES' TO PROCEED.

I replied FALLBACKSMSVSAMYES and received the message:
IGW524I SMSVSAM FALLBACK PROCESSING IS NOW COMPLETE

I set SMS to RLSINIT(YES) and that is where we sit.  I'm waiting for
the person in charge of this project to come in to run some commands
and we are probably going to try to vary the new SHCDS datasets on.
Before we do anything, is there something you want me to do?

CUSTOMER:
We did V SMS,SMSVSAM,ACTIVE and received:
IGW415I SMSVSAM SERVER ADDRESS SPACE HAS FAILED AND IS RESTARTING
Next we did:
D SMS,SHCDS
IEE932I 708
IGW612I 12:54:16    DISPLAY SMS,SHCDS
Name             Size   %UTIL Status  Type
-----------------0Kb  0%    N/A     N/A
-----------------0Kb  0%    N/A     N/A
-----------------0Kb  0%    N/A     N/A
-----------------0Kb  0%    N/A     N/A
Then we varied the SHCDS:
V SMS,SHCDS(PRIMARY.VSCSPX1),NEW
IGW619I ACTIVE SHARE CONTROL DATA SET 798
SYS1.DFPSHCDS.PRIMARY.VSCSPX1 ADDED.
V SMS,SHCDS(SECONDRY.VSCSPX2),NEW
IGW619I ACTIVE SHARE CONTROL DATA SET 838
SYS1.DFPSHCDS.SECONDRY.VSCSPX2 ADDED.
V SMS,SHCDS(SPARE.VSCSPX3),NEWSPARE
IGW619I SPARE SHARE CONTROL DATA SET 863
SYS1.DFPSHCDS.SPARE.VSCSPX3 ADDED.
IXL014I IXLCONN REQUEST FOR STRUCTURE IGWLOCK00 WAS SUCCESSFUL. 865
JOBNAME: SMSVSAM ASID: 00BE CONNECTOR NAME: TLP1
CFNAME: SCSCF1
IXL015I STRUCTURE ALLOCATION INFORMATION FOR 866
STRUCTURE IGWLOCK00, CONNECTOR NAME TLP1
 CFNAME     ALLOCATION STATUS/FAILURE REASON
 --------   ---------------------------------
 SCSCF1     STRUCTURE ALLOCATED
IGW453I SMSVSAM ADDRESS SPACE HAS SUCCESSFULLY 867
CONNECTED TO DFSMS LOCK STRUCTURE IGWLOCK00
STRUCTURE VERSION:AFD504C2A838AA03 SIZE:20224K bytes
MAXIMUM USERS:4 REQUESTED:4
LOCK TABLE ENTRIES:4194304 REQUESTED:4194304
RECORD TABLE ENTRIES:83481 USED:0
IGW321I No retained locks
IGW321I No Spheres in Lost Locks
IGW414I SMSVSAM SERVER ADDRESS SPACE IS NOW ACTIVE.
```

```
IGW467I DFSMS RLS_MAX_POOL_SIZE PARMLIB VALUE SET DURING 877
SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: TLP1
CURRENT VALUE: 40  1
IGW467I DFSMS DEADLOCK_DETECTION PARMLIB VALUE SET DURING 878
SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: TLP1
THIS SYSTEM IS OPERATING AS A LOCAL DEADLOCK PROCESSOR.
CURRENT VALUE: 30  10  1
IGW467I DFSMS SMF_TIME PARMLIB VALUE SET DURING 879
SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: TLP1
CURRENT VALUE: YES  1
IGW467I DFSMS CF_TIME PARMLIB VALUE SET DURING 880
SMSVSAM ADDRESS SPACE INITIALIZATION ON SYSTEM: TLP1
CURRENT VALUE: 3600  1

Then we did a D SMS,SHCDS
IGW612I 12:57:32    DISPLAY SMS,SHCDS
Name             Size   %UTIL Status  Type
PRIMARY.VSCSPX1  10800Kb  2%  GOOD    ACTIVE
SECONDRY.VSCSPX2 10800Kb  2%  GOOD    ACTIVE
SPARE.VSCSPX3    10800Kb  2%  GOOD    SPARE
-----------------0Kb  0%    N/A     N/A

D SMS,SMSVSAM,ALL
IGW420I DISPLAY SMS,SMSVSAM 915
DISPLAY SMS,SMSVSAM - SERVER STATUS
  SYSNAME:  TLP1       AVAILABLE ASID: 00BE STEP: SmsVsamInitComplete
  SYSNAME:  ........ ........... ASID: .... STEP: ....................
  SYSNAME:  ........ ........... ASID: .... STEP: ....................

DISPLAY SMSVSAM     - JOB STATUS
  SUBSYSTEMS CONNECTED:       0 BATCH:        1

DISPLAY SMSVSAM     - LOCK TABLE STATUS (IGWLOCK00)
  CONNECT STATUS:
    SYSNAME:  TLP1     ACTIVE           RSN: 00000000 RbldNotActive
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................
We did other displays and they all come back.  These new SHCDS data-
sets were allocated on the same system that we brought up SMSVSAM on.
This system is an OS390 R3 put level 9710 system.  Before the data-
sets were allocated on another system in the SYSPLEX.  This system is
an OS390 R2 put level 9605+ system.  The SHCDS datasets were then re-
cataloged to the system that SMSVSAM was started on.  These 2 systems
do not share a master catalog, but they share DASD.  What solved this
problem this time around?

IBM STATUS:
The Fallback is kind of like a reset for SMSVSAM.  All of the infolike
name of SHCDS and lock structure status is all deleted.  SO when this
```

```
completes SMSVSAM can be activated and will retrieve all new info.
Understand that this is to be used as a last resort, because a lot of
pertinent info is deleted like the lock structure data.  It appears
from the displays and the messages that eveything is working.

CUSTOMER:
We seem to have every working.  When I brought up out other system in
OS390 R3 and had the SHCDS datasets cataloged everything seemed to
work fine.
```

II12603

```
Item II12603
  APAR Identifier ...... II12603      Last Changed ........ 02/07/21
  VSAM RECORD-LEVEL SHARING, RLS SMSVSAM INITIALIZATION AND
  RECOVERY CONSIDERATIONS

  Symptom ...... IN INCORROUT         Status ........... INTRAN
  Severity ................... 3      Date Closed .........
  Component .......... INFOV2LIB      Duplicate of ........
  Reported Release ......... 001      Fixed Release ............
  Component Name V2 LIB INFO ITE      Special Notice
  Current Target Date ..              Flags
  SCP ...................
  Platform ............

  Status Detail: Not Available

  PE PTF List:

  PTF List:


  Parent APAR:
  Child APAR list:


  ERROR DESCRIPTION:
  When planning to implement vsam record-level sharing, rls, or
  when planning a smsvsam recovery strategy please carefully
  review the following documentation.
  (SC264920) 'OS390 DFSMSDFP Storage Administration Reference'
            Chapter 14: Administering Vsam Record-Level Sharing
  (SG242073) 'Getting the Most Out of a Parallel Sysplex'
             Section 6.3.4: Batch With CICS and Vsam-RLS
             Section 7.3.1: CICS and Vsam-RLS Considerations
  (Sy277611) 'OS390 DFSMSDFP Diagnosis Reference'
```

```
          Section 20.5: Vsam RLS Diagnostic Aids
(GA227286) 'Parallel Sysplex Recovery'
           Section 2.8: Recovering From a Vsam Rls Address Space
                Failure
(SC264919) 'DFSMS/MVS Planning for Installation'
           Chapter 6: Vsam Record Level Sharing
(GC331681) 'Installation Guide'
           Chapter 18: Preparing for Vsam-RLS
(sc267344) 'OS390 DFSMS Introduction'
           Chapter 4: Using Vsam Record-Level Sharing


LOCAL FIX:
           -----------
INITIALIZATION
The smsvsam server address space may be set to initialize
during ipl or can be activated by means of the
'V SMS,SMSVSAM,ACTIVE' command.
The documentation above outlines what must be setup before
smsvsam can initialize. If the address space fails to come up
successfully, use the following display commands to ascertain
the nature of the initialization problem.
'D SMS,SMSVSAM,ALL' To display smsvsam status within the
sysplex.
'D SMS,SHCDS' To display the status of  the shcds files.
'D XCF' To display coupling facility status information.
.
RECOVERY AND DIAGNOSIS
If the SMSVSAM server abnormally terminates a svcdump will be
written to the SYS1.DUMP dataset and the server should
automatically restart.  If the server does not restart, use
'V SMS,SMSVSAM,ACTIVE' to force the server to reinitialise.
The dump should be used to analyze and resolve the abnormal
termination condition.
It may become apparent that there is a non-terminating problem
with the SMSVSAM server. 'Display GRS' command may display
resource contention involving SMSVSAM. Workload directed to the
SMSVSAM server may not be completing causing slow downs or work
stoppage in jobs utilizing RLS.
When such a hang condition is detected, it is essential to
obtain dumps from every system in the sysplex on which SMSVXAM
is active. These dumps are required by IBM to analyze product
defects leading to the hang condition.
The following example slip can be set disabled and enabled when
a hang condition arises requiring dumps to SMSVSAM. When this
slip matches all systems in the sysplex will produce SVCDumps.
SLIP SET,IF,A=SVCD,N=(IEAVEDS0,0000,7FFF),
     JOBLIST=(GRS,CATALOG,SMSVSAM),
     DSPNAME('GRS'.*,'CATALOG'.*,'SMSVSAM'.*),
      SDATA=(PSA,SQA,CSA,LPA,TRT,SUM,LSQA,RGN,GRSQ,NUC,XESDATA),
     REMOTE=(JOBLIST,DSPNAME,SDATA),END.
As each system in the sysplex will produce a SVCDump
you must ensure the availability of adaquately large sys1.dump
data sets on all systems.
Once the dumps have been taken it may be necessary to terminate
the smsvsam server by use of the 'V SMS,SMSVSAM,TERMINATESERVER'
command. Attempt to identify the system most likely to be the
root of the hang condition and terminate it first. It may be
necessary to terminate additional or even all servers before the
 hang is resolved. Each terminated server should automatically
restart or may be activated by means of the 'V
SMS,SMSVSAM,ACTIVE' command. It may be necessary to terminate
additional or all servers before a terminated server can
successfully restart. You may make use of information returned
from the 'D SMS,SMSVSAM' commands to anallyze initialization
problems.
If the system does not respond to the terminateserver command,


issue 'FORCE SMSVSAM,ARM' to force the address space down.
SMSvsam should be restarted as if terminateserver had been
successful. A re-ipl should be necessary in only the most severe
situations and attempted only as a last resort.
```

II12243

```
Item II12243
  APAR Identifier ...... II12243      Last Changed ........ 00/08/10
  MISC ABENDOF4 IN SMSVSAM FOLLOWING 'V XCF,'SYSNAME''
  VSAMRLS INFOAPAR

  Symptom ...... AB ABENDOF4          Status ........... INTRAN
  Severity ................... 4      Date Closed .........
  Component .......... INFOV2LIB      Duplicate of ........
  Reported Release ......... 001      Fixed Release ............
  Component Name V2 LIB INFO ITE      Special Notice
  Current Target Date ..              Flags
  SCP ...................
  Platform ............

  Status Detail: Not Available

  PE PTF List:

  PTF List:
```

```
Parent APAR:
Child APAR list:


ERROR DESCRIPTION:
The command 'v sms,smsvsam,terminateserver' must be issued to
terminate the smsvsam address space prior to partitioning a
system from the sysplex. If a 'v xcf,'sysname'' is issued while
the smsvsam address space is active miscelaneous abend0f4s can
result. Examples of these abends include:
abend0f4 rc24, rc00000024, rsn62020071 rc62020071 idavstai
abend0f4 rc14, rc00000014 rsn00001122 rc00001122 idavqini
Additional keywords:
msgigw416i
```

BDC000022923

```
Item BDC000022923
Source..........: PDDBO  PDDBO
Last updated....: 04/29/2002
Abstract........: SMSVSAM address looping during connect to IGWLOCK00

USERS: ALL DFSMS SMSVSAM users

PROBLEM SUMMARY:Loop issuing the following messages:
msgixc579i pending deallocation for structure IGWLOCK00
msgigw469i no suitable coupling facility
msgixl013i IXLCONN failed rc0c rsn02010c08.
These messages flood the console

SOLUTION: User specified incorrect structure size
PROBLEM DETAILS:
 TYPE:
 COMPID: 5695DF122
 RELEASE: 1F0

Customer :
Hi Brought up SMSVSAM for the 1st time on Friday. After issuing the
V SMS,SMSVSAM,Active command, SMSVSAM loops trying to connect to the
IGWLOCK00 structure. There was not enough space in the CF to define
the IGWLOCK00 structure.

I have FTPed dump and SYSLOG..

IBM DFP:
Hello
```

```
The dump does not contain all that much of the storage for the address
space.  I see we issue the following messages over and over from about
12:15 to 12:55 when you forced SMSVSAM address space.
msgixc579i pending deallocation for structure IGWLOCK00
msgigw469i no suitable coupling facility
msgixl013i IXLCONN failed rc0c rsn02010c08.
The trace table is insufficient to determine where the loop lies also.
The WTO appears to be issued from 5e25bc0.  This is also not in the
dump nor is the parameter list so I cannot confirm this is where the
problem is.  I can tell you this address is in IGDZILLA+x'34bco' .  If
you could get an AMBLIST of IGDZILLA and send it I can then check to
what module we are in and see if that will help.
Is this recreatable?
If so, I can put together some information to capture better doc.


Customer
Hi AMBLIST sent
I believe we can recreate .


IBM DFP:

received the amblist.  Module is IGDMCSI2 +x'f0' which is the module
that issues all IGW messages.  Not too helpful.  I will check the code
around the SVC 35 to see where we were at that time.  Will update later.
Customer:
HI  Is there any other info you need. ?  I away on holidays until
next year. In the meantime the DFHSM folks would like to continue with
TESTING..

IBM DFP:
Hello
I am still working on this with the MVS folks.  I will update when I
have more information.  Who in the HSM group will be working on this
just in case I need to contact them?
IBM:
thanks for the update.  we are still trying to figure this out and
hopefully will have something for you today.

IBM DFP:
Hello VSAM
I am not sure if you guys are the right people for this or if there is
another queue for SMSVSAM stuff.  Please forward to the correct queue if
it is not you guys.
Please take a look at my update on page 4 of the pmr.  SMSVSAM could not
connect to IGWLOCK00 because the CF structure was full.  However we
continued looping trying to connect.  I am sending the syslog to you
via FTP per II03855.  This should be easy enough to recreate I suspect.
```

```
If there is anything further that you require let me know.


Requeueing to the RLS queue for diagnosis.


VSAM L2:
Hi
  By design, SMSVSAM will continue attempting to connect to the lock
structure.  If there is no room in the coupling facility, room needs to
be made, or the CFRM needs to be changed.

IBM DFP:
Hello
This is totally unreasonable.  How are the operators supposed to take
any action when SMSVSAM is looping trying to connect to the CF?  This
is like saying you shouldn't do anything when you get an X37 abend, just
keep trying eventually the PUT will work?  This  is ludicrous.  IF there
is no space in the CF then fail the request and allow the operators
time to correct it and try again.
Please adddress this as a defect in the code.
Thanks,
action plan: sending to VSAM RLS level 2.
Vsam L2
  Development looked through the code involved.  It is not in a loop.
If SMSVSAM cannot connect to the lock structure during initialization
it will go into a wait after issuing the IGW469i.  It is notified every
time a change is made to the CF and will attempt to re-allocate the
lock structure.  If unsuccessful, it will issue the message and go into
a wait again.  It is our opinion that changing the design to terminate
is not in the best interest for all customers.
  Regards,

IBM DFP:
I took another look at the dump and do see that we are waiting and then
being woken up every once in awhile.  I have asked MVS to take a look
at the trace to see if she can figure out what XCF is changing that is
causing SMSVSAM to wake up and try the connect again.

IBM MVS:
To C/T:
  The customer got the following sequence of messages after issuing
a V SMS,SMSVSAM,Active command:
IXC579I PENDING DEALLOCATION FOR STRUCTURE IGWLOCK00 IN
        COUPLING FACILITY  009672.IBM.02.000000049747
                           PARTITION: 2    CPCID: 00
HAS BEEN COMPLETED.
PHYSICAL STRUCTURE VERSION: B6D0D708 14576381
INFO116: 13060020 01 6A00 0000000B
```

```
TRACE THREAD: 00000DE0.
IXC579I PENDING DEALLOCATION FOR STRUCTURE IGWLOCK00 IN
        COUPLING FACILITY  009672.IBM.02.000000049748
                           PARTITION: 2    CPCID: 00
HAS BEEN COMPLETED.
PHYSICAL STRUCTURE VERSION: B6D0D70A A2CCF662
INFO116: 13060020 01 6A00 00000003
TRACE THREAD: 00000DE0.
IGW469I NO SUITABLE COUPLING FACILITY, 779
NUMSYSTEMS:11 STRUCTURESIZE:500000 LOCKENTRIES:34217728
IXL013I IXLCONN REQUEST FOR STRUCTURE IGWLOCK00 FAILED.
JOBNAME: SMSVSAM ASID: 006D CONNECTOR NAME: K300
IXLCONN RETURN CODE: 0000000C,  REASON CODE: 02010C08
IXL015I STRUCTURE ALLOCATION INFORMATION FOR
STRUCTURE IGWLOCK00, CONNECTOR NAME K300
 CFNAME     ALLOCATION STATUS/FAILURE REASON
 --------   --------------------------------
 CF6L2      INSUFFICIENT SPACE
 CF5L2      INSUFFICIENT SPACE
These messages were repeated over and over again appearing to be in
a loop so the customer took a dump of the situation.   From the dump
(and with assistance from the VSAM c/t) we have determined that
IGWLNI01 repeatedly issues the IXLCONN to connect to IGWLOCK00.
IGWLNI01 checks the return code from the IXLCONN and since it is
non zero it issues IGW469I and then waits on an ECB that is posted
by their ENF35 exit.  I have looked at their ENF35 exit (IGWSSEN2)
and it simply causes the ECB to be posted when it is given control.
It appears that the ENF35 exit is being driven frequently.  This is
why they continually keep trying to issue the IXLCONN.   I have
reviewed all the reasons why the ENF35 exit is given control and
I cannot determine from the dump exactly why the exit is being
driven so frequently. I wasn't sure why the IXC579I messages were
being issued everytime we did the IXLCONN but INFO116 indicates
that module IXCl2ASR is the module that is driving IXCXRHT so
I think these messages are probably normal.   I cannot see why
these would trigger the ENF35...
  Also - in the SYSXES GLOBAL component trace just prior to
the IXCASR COMPLETE entry -  I see several HWLAYER entries for
ENTRY TO IXLERREC but I cannot tie these into the ENF35 signal.
  Any ideas??   Thanks

MVS C/T:
Hi
  You can send us the dump if you like.

IBM DFP:
Hello
I have ftp'd the tersed dump and syslog to you.  Let me know if you need
anything further.
```

```
Thanks


MVS C/T:
  Reveiwing doc.
  Can we find out who is at LMOD IXCI2PVT+027A78?
Thanks

IBM MVS:
Hi we are trying to solve this mystery but we need an AMBLIST
of IXCI2PVT that matches the dump that was taken on K300 on Nov. 30th
at 12:52:42.  I know we are a little late in asking but if you haven't
down any maintenance to XCF load libraries on that system since
Nov. 30th then we should be ok.   You can send it to me


IBM C/T:

  Trying to determine what would cause the apparent repeated ENF35,
though as MVS and I discussed, we cannot see any remnants of them
(ie. no storage containig the ENF plist) in the dump.
  At this point I suspect that the deallocation seen in the MSGIXC579I
is relevant.  L2ASR does an IXCXRHT, and I think IXCL2RHT is doing an
IXCXLSIG(NEWRESOURCEAVAIL) which will result in an ENF35.  What is not
clear at this point is why the str is being deallocated since the
externals indicate it was not allocated because of insufficient CF
space.
  There are some CFRM trace entries that look relevant, and also
some system trace entries where it appears that IXCL2ALF is doing
PC B15.  The lmod map previously requested should help clarify these.
Action Plan:
  Continue on Monday.
Action Taken:
  From discussion with development, this may involve MINSIZE.  The
str may actually get allocated at the size of whatever is available in
the CF, but if it does not meet the MINSIZE requirement, it is then
deallocated.
Action Plan:
  Check STBL in dump for MINSIZE value and review L2ALF code.
Hello MVS.
  Any luck with the lmod map?  I also want to know what if anything
is specified in the CFRM policy for the MINSIZE keyword on defintion
of IGWLOCK00.
Thanks

Customer:
Sorry for the delay. AMBLIST has been XMITTED to TORIBM.CS32587.
The IXCI2PVT was last updated as follows.
```

```
Entry Type:  MOD                            Zone Name: MVSSAQ
Entry Name:  IXCI2PVT                       Zone Type: TARGET

  FMID:    HBB7703                     LASTUPD: HBB7703
  RMID:    JNLSHRK     DISTLIB: AOSXCF

        -------- -------- -------- -------- -------- -------- --------
MOD     IXCI2PVT
SECTS   IXCI2PVT
The JNLSHRK is a usermod provided by Dave Sherman to allow us to SHRINK
our CFRM dataset dynamically. This did not change since dump...

Yes the structure was defined as follows but with 2 or 3 extra zeros
for the various size fields. ie SIZE(5000000)

STRUCTURE NAME(IGWLOCK00) SIZE(5000)
   INITSIZE(2000)
   MINSIZE(1000)
   FULLTHRESHOLD(80)
   ALLOWAUTOALT(YES)
   REBUILDPERCENT(1)
   PREFLIST(CF5L2, CF6L2)

Hope this helps.
IBM MVS:
Received the AMBLIST from Deo into dataset 'isc.pmr07201.b057.amblist'.
MVS C/T it is on it's way to you......thanks again for the help.

IBM MVS:
  What the customer says about the policy does not coincide with what
I see in the dump which shows that SIZE, MINSIZE, and INITSIZE were
all 500000.
  Based on what is in the dump, the flow appears to be that there is
not enough space in the CF to allocate 500000 so L2ALF allocates a
structure at whatever size is avaialble.  But since the allocated size
is less than MINSIZE, the structure gets dealloacted in L2RHT when
L2ASR does an IXCXRHT to clean the checkpoint.  L2RHT also issues the
MSGIXC579I and does an IXCXLSIG(NEWRESOURCEAVAIL) which ultimately
results in L2MSG doing the ENF35.
  I'm reviewing this scenario with development.  I'll keep a secondary
on our queue and provide a further update when I hear back.
Regards

Customer:
Hi sorry to mislead you. What I meant was the various sizes were
greater by 2 or 3 zeros..

MVS C/T:
Hi IBM MVS
```

```
  I saw the comment (which is also what I understood from a prior
update), but it still does not make sense since SIZE, INITSIZE, and
MINSIZE are all the same in the dump.  At this point I don't think it
is relevant to the problem that occurred, so let's just see what
response I get from development.
Regards,
IBM MVS:
Hi MVS C/T.....did you hear back from development?

MVS C/T:
Hi IBM MVS.
  Yes, they say this is user error and the code is working as it should.
If the ENF was not issued, then a hang could occur because the
connector has no initiative to ever try to get connected.
Regards

IBM MVS:
Talked with IBM DFP  In this case the 'user' of the XCF
services is SMS which continually retries the IXLCONN when it receives
the ENF35.   The IXLCONN could never have been satisfied because
the MINSIZE was too large.   They will investigate if the SMS code
could detect the situation.

IBM DFP:

Hello MVS C/T:
We have been able to determine why this appears to be looping.  What is
occurring is is VSAM issues the CONNECT to connect to the structure.
XCF attempts to allocate the structure but there is not enough room
even to allocate the MINSIZE that was specified.  So XCF then does a
deallocate to unallocate the space that was successfully allocated for
this request.
Since the allocated size is less than MINSIZE, the structure gets
deallocated in IXCL2RHT when IXCL2ASR does an IXCXRHT to clean the
checkpoint.  L2RHT also issues the msgixc579i and then does an
IXCXLSIG(NEWRESOURCEAVAIL) which ultimately results in IXCL2MSG doing
the ENF35.  XCF has to do this in order to tell all listeners that
there is now space available in the CF.
The problem is SMSVSAM just keeps reissuing the CONNECT so it appears
that we are looping when in fact we really are not.  What SMSVSAM
should do is attempt the CONNECT again (maybe a couple of times) and
then give up.  Otherwise we would be doing this forever, since in this
specific situation the ral problem was the size that was specified in
the structure was incorrect (and way too big).
Do you think it would be reasonable that we should only retry some
finite number of times and then give up?
Thanks
```

```
Hello MVS DFP:
  I discussed this with Terri again.  We feel that it is best to leave
the code unchanged.  Since RLSINIT(YES) was specified in PARMLIB,
we assume that the user intended SMSVSAM to start up
at IPL and had planned accordingly.
  Regards,

MVS DFP:
I discussed the resolution of this problem with Customer and he agreed
to leave the code as is.   He agreed to close the ETR.
```

RTA000141603

```
Item RTA000141603
Source..........: PDDBO  PDDBO
Last updated....: 09/26/2000
Abstract........: Rebuild of IGWLOCK00 stalled after loss of coupling facility.

USERS: VSAM

PROBLEM SUMMARY:
Rebuild of IGWLOCK00 stalled after loss of coupling facility.
SMSVSAM unable to re-INIT on all systems.

SOLUTION:
Issue V SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE comand.

PROBLEM DETAILS:
 TYPE: WAD
 COMPID: 5695DF122
 RELEASE: 1D0

CUSTOMER:
Lost the CF on which the IGWLOCK00 structure resided.
While the structure was being rebuilt, also lost the SMSVSAM
server on N7B.  This caused all the other SMSVSAM servers
to kill themselves (as expected).  However, once they
restarted, the rebuild of IGWLOCK00 was either stalled
or stopped.  In any event, the structure did not rebuild
onto the surviving CF, and all the connection to the
structure were listed as FAILED-PERSISTENT.  Took a sysplex-
wide dump of SMSVSAM to help determine what happened to the
rebuild.  (Just as a side note, once I recovered the CF,
the structure was successfully recovered).
```

```
The same information could have been gotten with the D SMS,SMSVSAM or
via the D XCF,STR,STRNAME=IGWLOCK00 commands.

The systems will only be able to start if the current copy of IGWLOCK00
recovers from its connectivity problem or if the current structure copy
of IGWLOCK00 is destroyed. This could be done with the command:
V SMS,SMSVSAM,FORCEDELETELOCKSTRUCTURE
or by using the appropriate XCF commands.

This is working as designed.  If connectivity is reestablished everyone
should take off.  If the lock structure is deleted,  then all currently
open VSAM data sets will go into LOST LOCKS state.

CUSTOMER:
Ok. I'll rerun the scenario with the extra step of deleting the lock
structure and I'll let you know what happens.

CUSTOMER:
Re-executed the scenario with the additional step (of deleting
the lock structure).  As seemed to work well.

Additional search words:
LVL1PDDB   (DMD)
50566,180   VSAM
VRLSPDDB97
```

Glossary

Access method control block (ACB). A control block that links an application to a VSAM data set

Active control data set (ACDS). A VSAM linear data set that controls the storage management policy for an installation. It is shared by each system that is using the same SMS configuration to manage storage.

Active lock. A lock obtained by an executing subsystem. Requests for locks incompatible with a held active lock are queued. See also retained lock.

Alternate index. A key-sequenced data set containing index entries organized by the alternate keys of its associated base data records. It provides an alternative way of locating records in the data set on which the alternate index is based.

Application owning region (AOR). A CICS region that does not own terminals or files but which purely runs CICS transactions, relying upon terminal-owning and file-owning regions for terminal and file access.

Automatic class selection (ACS). A facility that allows a DFSMS system to decide which constructs (data class, storage class, management class or storage group) should be associated with a data set based on definition attributes.

Automatic restart manager (ARM). An OS/390 component that provides recovery and restart capabilities.

Back-out. Undo all changes made to a protected resource since the previous sync point.

Basic sequential access method (BSAM). An access method for sequentially organized non-VSAM data that relies on an application program to block and unblock data records.

CEMT. A CICS-supplied transaction used to invoke all the master terminal functions. These functions include inquiring and changing the value of parameters used by CICS, altering the status of system resources, terminating tasks and shutting down CICS.

Cluster. A named VSAM structure consisting of a group of one or more related components.

Consistent read (CR). A read operation where a share lock is obtained and released as part of the read process. This ensures that the reader sees the latest committed copy of the record as at the time of the read.

Consistent read explicit (CRE). A share lock is obtained as part of the read process and held until explicitly released. This ensures that a subsequent read while the lock is still held will always return the same version of the data.

Control area (CA). A fixed-size area in which VSAM stores control intervals. It is the unit of allocation for VSAM data.

Control interval (CI). A fixed-length area in which VSAM stores records. It is the unit of transfer between VSAM and disk storage.

Coupling facility (CF). The hardware that provides services to support data shared between OS/390 systems.

Coupling facility control code (CFCC). The software running within a Coupling Facility that provides Coupling Facility services.

Cross-system coupling facility (XCF). An MVS component that provides services for the exchange of messages between members of a sysplex.

DFHLSCU. A CICS-supplied utility to calculate space needed and average buffer size for log streams.

DFR. An ACB parameter affecting buffer writing when a data set is opened for LSR or GSR. It defers the writing of a buffer until explicitly requested to do so or until a buffer is needed for a read request.

DFSMS. OS/390 component replacing DFSMS/MVS.

DFSMS/MVS. A System/390 licensed program that provides storage data and device management functions. It comprises DFSMSdfp, DFSMSdss, DFSMShsm and DFSMSrmm.

Entry-sequenced data set (ESDS). A data set whose records are loaded without regard to their contents and whose relative byte addresses cannot change. New records are added at the end of the data set.

Exclusive lock. A lock used to permit a resource to be updated. An exclusive lock can only be obtained if no other user holds a lock on the resource. See also *share lock*.

External CICS interface (EXCI). A method of invoking CICS services from outside CICS, for example, by batch jobs.

File owning region (FOR). A CICS region that is configured so that its only function is to own and access CICS files on behalf of other CICS regions.

Forward recovery. The process of applying the records contained in a redo log to redo the changes made to a data set in the event that a data set is lost or damaged and must be recovered from a backup copy.

GET NUP. A macro that reads a record from a VSAM data set and, through specifications in the RPL parameters, informs VSAM that the record will not be updated or deleted.

GET UPD. A macro that reads a record from a VSAM data set and, through specifications in the RPL parameters, informs VSAM that the record will be updated or deleted so that VSAM can obtain the appropriate locks.

GETIX. A VSAM macro that allows you to read an index control interval directly. It is normally used only for index maintenance and repair.

Global resource serialization (GRS). A component of OS/390 used for serializing the use of system resources.

Global shared resources (GSR). A form of VSAM processing that uses a buffer pool that is shared dynamically between users in several address spaces. The buffer pool is used to satisfy I/O requests without a physical I/O.

Harden. Write to a non-volatile medium such as disk or a log stream.

Integrated catalog facility (ICF). A cataloging scheme that comprises basic catalog structure (BCS) catalogs and multiple VSAM volume data sets (VVDSs).

Integrated cluster bus (ICB). A form of link between a processor and a Coupling Facility.

Internal coupling channel (IC). A link between OS/390 and a Coupling Facility running in the same physical processor complex.

IDCAMS. A utility program that provides services in support of VSAM data sets.

Instance. An instance of DFSMStvs is the single version ofDFSMStvs running in one z/OS image and defined by the IGDSMSxx PARMLIB member for that z/OS image.

Inter-system coupling (ISC). A old type of link between a processor and a Coupling Facility, superseded by an ICB link for distances below seven meters with suitable processors.

Java Database Connection (JDBC). A set of interfaces and protocols originally designed for Java applications to be able to access relational databases.

Key-sequenced data set (KSDS). A VSAM data set whose records are loaded by ascending key sequence and controlled by an index.Records are retrieved and stored by keyed access or addressed access and new records are inserted in key sequence because of free space allocated within the data set. Relative byte addresses of records can change because of control interval or control area splits.

Logical partition. A subset of the processor hardware that is defined to support an operating system.

Local shared resources (LSR). A form of VSAM processing that uses a buffer pool that is shared dynamically between all users in one address space. The buffer pool is used to satisfy I/O requests without a physical I/O.

MIPS. Million Instructions per second.

NDF. Not deferred. See DFR.

No read integrity (NRI). A form of read operation performed without locking so that the record returned may contain uncommitted data.

Note string position (NSP). For direct access to VSAM data sets, this RPL parameter requests that VSAM remembers the position in the data set for subsequent sequential access.

Non-shared resources (NSR). A form of VSAM processing that provides static buffering on a per data set basis.

OS/390. OS/390 is an operating system for System/390 processors consisting of more than 50 base elements and integrated optional features including MVS and DFSMS.

Parallel sysplex. A sysplex with at least one coupling facility.

POINT. A macro that allows you to position to a specified record within a VSAM data set in preparation for a subsequent read or write.

Processor resource/system manager (PR/SM™). The feature that allows a processor to use several OS/390 images simultaneously by providing logical partitions.

PUTIX. A macro that allows you to write a VSAM index control interval directly. It is normally used only for index maintenance and repair.

Queued sequential access method (QSAM). An access method for sequentially organized non-VSAM data that automatically blocks and unblocks data records on behalf of an application program.

Record level sharing. A VSAM extension that provides direct access to a VSAM data set from multiple systems, providing cross system locking and buffer invalidation.

Resource recovery services (RRS). An MVS component that provides sync point management facilities to coordinate resource recovery for programs that issue commit and back-out requests to participating resource managers.

Redo. See forward recovery.

Relative record data set (RRDS). A type of VSAM data set containing fixed length records which are accessed by relative record number.

Request parameter list (RPL). A macro that is used by programs to specify parameters for I/O requests to VSAM data sets identified by an ACB.

Resource access control facility (RACF). An IBM licensed program which is a base element of OS/390 and provides access control by identifying and verifying users, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected resources.

Retained lock. The status of a lock when the subsystem executing a transaction fails. A request incompatible with a retained lock is rejected, rather than queued as would be the case for an active (not retained) lock. Retained status should be cleared when recovery is complete.

Share lock. A lock used to serialize read access with update access to the same resource. Many users can hold a share lock, denoting that they are all reading the resource. If there is an exclusive lock held for a resource, a request for a share lock is queued. If there is an share lock held for a resource, another request for a share lock is granted. See also *exclusive lock*.

Sharing control data set (SHCDS). The repository for data required to ensure the integrity of a data sharing environment in the event of the loss of a coupling facility cache or lock structure.

Shunt. The process of putting aside a unit of recovery that can neither committed not backed out due to an error (such as an I/O error on one of the data sets involved).

Shunt log. A secondary undo log in which a record of the before image of records in data sets are written before changing those records. Records are moved to the secondary undo log from the primary undo log when DFSMStvs is unable to complete back-out processing for them, usually because a resource is unavailable.

Sphere. A logical group of related VSAM components containing both data and index.

SOD. Statement of Direction.

Storage Management Subsystem (SMS). An operating environment that helps to automate and centralize the management of storage. SMS provides a storage administrator with control over data class, storage class, management class, storage group and ACS routine definitions.

Sysplex. A collection of MVS or OS/390 systems with a single common time reference that cooperate to process work and communicate using XCF.

Terminal owning region (TOR). A CICS region that is configured to act only as the interface between terminals (or distributed systems appearing as CICS terminals) and an application owning region

Unit of recovery. Work done by or on behalf of a context between one point of consistency and another.

Unit of recovery ID (URID). The unique identification of a unit of recovery.

UPAD. VSAM exit for user processing during a VSAM request.

Variable relative record data set (VRRDS). A type of VSAM data set containing fixed or variable length records which are accessed by relative record number.

Virtual storage access method (VSAM). If you need to check this, stop reading the book.

XCF. See Cross-system coupling facility.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 482. Note that some of the documents referenced here may be available in softcopy only.

- *CICS and VSAM Record Lavel Sharing: Implementation Guide*, SG24-4766
- *CICS and VSAM Record Lavel Sharing: Planning Guide*, SG24-4765
- *CICS and VSAM Record Lavel Sharing: Recovery Considerations*, SG24-4768
- *DFSMS/MVS V1R4 Technical Guide*, SG24-4892
- *Transactional VSAM Application Migration Guide*, SG24-6145
- *OS/390 Parallel Sysplex Configuration: Overview, Volume 1*, SG24-5637
- *OS/390 Parallel Sysplex Configuration Cook Book, Volume 2*, SG24-5638
- *OS/390 Parallel Sysplex Configuration: Connectivity, Volume 3*, SG24-5639

Other publications

These publications are also relevant as further information sources:

- *z/OS DFSMS Managing Catalogs*, SC26-7409
- *z/OS DFSMS Using Data Sets*, SC26-7410
- *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402
- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- *z/OS MVS Setting Up a Sysplex*, SA22-7625
- *z/OS MVS Programming: Resource Recovery*, SA22-7616
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589
- *z/OS MVS Programming: MVS Assembler Services Guide*, SA22-7605
- *z/OS MVS Programming: Authorized Assembler Services*, SA22-7610

- *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348
- *CICS VSAM Recovery Implementation Guide*, SH26-4126
- *CICS Recovery and Restart Guide*, SC33-1698
- *DFSMStvs Administration Guide*, GC26-7483

Online resources

These Web sites and URLs are also relevant as further information sources

- IBM Storage Web site

  `http://www.storage.ibm.com/software/sms/details.html#RLS`
- IBM Coupling Facility sizer

  `http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html`
- z/OS RMF

  `http://www.ibm.com/servers/eserver/zseries/rmf`
- Catalog and VSAM Knowledge Base

  `http://knowledge.storage.ibm.com`
- DFSMS Technical Support

  `http://ssddom02.storage.ibm.com/techsup/webnav.nsf/support/dfsms`
- TRSMAIN Utility

  `http://techsupport.services.ibm.com/390/trsmain.html`
- Flashes

  `http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/Flashes`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

VSAM Demystified

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

VSAM Demystified

Learn the latest VSAM functions and manage VSAM data

Understand, evaluate, and use VSAM properly

Problem determination and recommendations

Virtual Storage Access Method (VSAM) is one of the access methods used to process data. Many of us have used VSAM and work with VSAM data sets daily, but exactly how it works and why we use it instead of another access method is a mystery.

This book helps to demystify VSAM and gives you the information necessary to understand, evaluate, and use VSAM properly. It clarifies VSAM functions for application programmers who work with VSAM. The practical, straightforward approach should dispel much of the complexity associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This IBM Redbook is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions.

This book also builds upon the subject of Record Level Sharing and the new z/OS feature called DFSMStvs.

SG24-6105-01 ISBN 0738453234