Mainframe

The term 'MainFrame' brings to mind a giant room of electronic parts that is a computer, referring to the original CPU cabinet in a computer of the mid-1960's. Today, Mainframe refers to a class of ultra-reliable large and medium-scale servers designed for carrier-class and enterprise-class systems operations. Mainframes are costly, due to the support of symmetric multiprocessing (SMP) and dozens of central processors existing within in a single system. Mainframes are highly scalable. Through the addition of clusters, high-speed caches and volumes of memory, they connect to terabyte holding data subsystems.

Mainframe computer

Mainframe is a very large and expensive computer capable of supporting hundreds, or even thousands, of users simultaneously. In the hierarchy that starts with a simple microprocessor at the bottom and moves to supercomputers at the top, mainframes are just below supercomputers. In some ways, mainframes are more powerful than supercomputers because they support more simultaneous programs. But supercomputers can execute a single program faster than a mainframe. The distinction between small mainframes and minicomputers is vague, depending really on how the manufacturer wants to market *its machines.*

Modern mainframe computers have abilities not so much defined by their single task computational speed (usually defined as MIPS — Millions of Instructions Per Second) as by their redundant internal engineering and resulting high reliability and security, extensive input-output facilities, strict backward compatibility with older software, and high utilization rates to support massive throughput. These machines often run for years without interruption, with repairs and hardware upgrades taking place during normal operation.
Software upgrades are only non-disruptive when Parallel Sysplex is in place, with true workload sharing, so one system can take over another's application, while it is being refreshed. More recently, there are several IBM mainframe installations that have delivered over a decade of continuous business service

as of 2007, with hardware upgrades not interrupting service. Mainframes are defined by high availability, one of the main reasons for their longevity, because they are typically used in applications where downtime would be costly or catastrophic. The term Reliability, Availability and Serviceability (RAS) is a defining characteristic of mainframe computers. Proper planning (and implementation) is required to exploit these features.

In the 1960s, most mainframes had no interactive interface. They accepted sets of punch cards, paper tape, and/or magnetic tape and operated solely in batch mode to support back office functions, such as customer billing. Teletype devices were also common, at least for system operators. By the early 1970s, many mainframes acquired interactive user interfaces and operated as timesharing computers, supporting hundreds or thousands of users simultaneously along with batch processing. Users gained access through specialized terminals or, later, from personal computers equipped with terminal emulation software. Many mainframes supported graphical terminals (and terminal emulation) by the 1980s. Nowadays most mainframes have partially or entirely phased out classic terminal access for end-users in favor of Web user interfaces. Developers and operational staff typically continue to use terminals or terminal emulators.
Historically, mainframes acquired their name in part because of their substantial size, and because of requirements for specialized heating, ventilation, and air conditioning (HVAC), and electrical power. Those requirements ended by the mid-1990s with CMOS mainframe designs replacing the older bipolar technology. In a major reversal, IBM now touts its newer mainframes' ability to reduce data center energy costs for power and cooling, and the reduced physical space requirements compared to server farms.

Characteristics of mainframes

Nearly all mainframes have the ability to run (or host) multiple operating systems, and thereby operate not as a single computer but as a number of virtual machines. In this role, a single mainframe can replace dozens or even hundreds of smaller servers. While mainframes pioneered this capability, virtualization is now available on most families of computer systems, though not to the same degree or level of sophistication.

The distinction between supercomputers and mainframes is not a hard and fast one, but supercomputers generally focus on problems which are limited by calculation speed while mainframes focus on problems which are limited by input/output and reliability ("throughput computing") and on solving multiple business problems concurrently (mixed workload).

The mainframe is the backbone of many industries that are the lifeblood of the global economy. More mainframe processing power is being shipped now than has ever been shipped. Businesses that require unparalleled security, availability, and reliability for their applications depend on mainframe.

Mainframes were designed initially for high-volume business transactions and, for more than 40 years, have been continually enhanced to meet the challenges of business data processing. No computing platform can handle a diversity of workloads better than a mainframe.

But aren't "insert-your-favorite-alternative-platform" computers cheaper/faster/easier to operate? The answer is: It all depends. A student who is composing his term paper does not have the same information needs as a bank that needs to handle millions of transactions each day, especially because the bank also needs to be able to pass security and accounting audits to verify that each account has the correct balance.

Mainframes aren't for every computing task. Businesses opt for mainframes and mainframe operating systems when they have large volumes of data, large transaction volumes, large data transfer requirements, a need for an extremely reliable system, or many differing types of workloads that would operate best if they were located on the same computer. Mainframes excel in these types of environments.

Mainframe Features

Enterprise class mainframe integration goes beyond the basic capability to connect to mainframe datasources to distributed platforms. It requires that the underlying mainframe integration architecture be engineered to extend the characteristics of mainframe quality of service – high performance, *scalability, security, reliability and manageability – to the distributed applications* *using the mainframe resources. Shadow's exploitation of IBM's specialty engines, the zIIP and zAAP, offers improved performance while lowering the Total Cost of Operation (TCO) by diverting processing intensive data and SOA integration workloads from the General Purpose Processor to the unmeasured, non-speed restrictive environment of the specialty engines.*

The mainframes are equipped with large number of high processing embedded CPUs which offer it greatest processing capacity.

Huge Memory Capacity:

The memory capacity embedded within mainframe is very huge so as to support large scale operations of mainframe applications. In some cases it can even amount to more than 8 Gigabyte.

Systems Management Performance Analysis and Tuning - Enables rapid views of workload levels running through Shadow. Organizes data into tables, charts and graphs for high-value capacity planning and trending analysis. Application Debugging - Enables detailed tracing and logging with multiple levels of analysis to aid design time and run-time problem diagnosis with 100+ capture points. Automation - Allows automated management of large-scale Shadow implementations for improved availability and throughput. Comprehensive execution of events monitored with a wide range of automated actions available. Monitoring and Control - Provides online, real-time visibility and measurement of all Shadow-related activity to preserve service. Multiple resource utilization thresholds available with a broad range of automated responses for resource breaches. security. Security Optimization and Management (SOM) – enables the use of cached credentials for improved performance and lower costs while maintaining the integrity of the security infrastructure with the real-time purging invalidated credentials as security policies change Auditing - Identifies ultimate, unique end-user of mainframe resources in a standard thread-pooled application platform environment. Encryption - Optimizes encryption of information traveling between the application platform and the mainframe. Manages digital certificates on the mainframe. Virtual Authentication - Provides authentication flexibility to reduce vulnerabilities of program-initiated authentication requests to a mainframe. Subsystem Security Integration - Identifies ultimate end-user of mainframe resources in a standard, thread-pooled application platform environment. Mainframe and Distributed Security - Fully integrates with RACF, CA-TopSecret and CA-ACF2 to provide a robust security model. Distributed platforms support for SSL and client-side, certificate-based authentication. Transactions Distributed Transaction Management - Supports prevalent J2EE and .NET distributed transaction management. Built to X/Open XA transaction management standards with support for BEA WebLogic, IBM WebSphere, BEA Tuxedo and Microsoft Transaction Services thread management and choreography. Support for Enterprise Transactions - Full "two-phase commit" (2PC) Transaction Support. Extensive support for z/OS Resource Recovery Services (RRS). All XA 2PC semantics are supported and exposed directly or via application server transaction manager support. z/OS Resource Recovery Services - Enables transaction management via extensive exploitation of z/OS Resource Recovery Services (RRS). All updates made via Shadow within the same global transaction are handled with a single-phase transaction. Scalability Load Balancing and Failover - Ensures ultimate levels of scalability and availability. Connection Virtualization - Allows unlimited concurrency of connections from distributed applications. Transactional Activity Blocking - Reduces lock duration for complex transactions and significantly improves response times for Update and Insert intensive applications policy. Accounting and Chargeback Support - All integration activity supported by Shadow is logged to the z/OS Systems Measurement Facility (SMF). Various levels of granularity are supported, from session-based accounting to accounting at the interaction level.

Mainframes systems can share the workload among different processors and input/output devices. This increases the processing ability and power of the mainframe for efficient performance.

The mainframe system and application work efficiently in critical and complex business applications. This is because such operations demands the system to be available at all times with 100% throughput, and which is completely achievable and provided by mainframe system. It is achieved by mainframe system by its great design where Memory chips, memory busses, I/O channels and power supplies are provided with a pair more, ensuring the system availability throughout.

Serviceability:

Most of the components in the mainframe system are replacable with the system operating concurrently.

Tightly Coupled Clustering Technology:

The mainframe system supports Tightly Coupled Clustering Technology also known as Parallel Sysplex. This feature facilitates operating up to 32 systems as a single system image. Even if one system fails, the work continues automatically on the next live system and there is loss of work and performance is thereby increased. Besides this makes the maintenance unit easier as one or more system which need to be maintained for hardware or software can be removed and this does not affect the remaining systems as the remaining systems continues with its processing as usual and after the maintenance the removed system can be again plugged in. Parallel Sysplex helps in acheiving reliability, availability and serviceability.

Apart from the above mentioned features Mainframe also offers:

Supports Maximum I/O connectivity
Have capability of providing Maximum I/O bandwidth
Have the greatest ability of producing fault tolerant computing.
Capacity to manage Large Users

From the above features mainframe system proves to be a best resource and application for critical and complex business operations which gives scalability, security, reliability and availability of system throughout and also proves to be cost-effective solution achieved with greater efficiency and performance which is very important for any business operation.

In this situation one might have a feeling of supercomputer to be similar terms to mainframe systems. But there are some distinct differences. Though there are some similarities like both supercomputers and mainframes support and handle parallel processing the way it is handled differs between them.

Overview on CICS

CICS (Customer Information Control System) is an online transaction processing (OLTP) program from IBM that, together with the COBOL programming language, has formed over the past several decades the most common set of tools for building customer transaction applications in the world of large enterprise mainframe computing. A great number of the legacy applications still in use are COBOL/CICS applications. Using the application programming interface (API) provided by CICS, a programmer can write programs that communicate with online users and read from or write to customer and other records (orders, inventory figures, customer data, and so forth) in a database (usually referred to as "data sets") using CICS facilities rather than IBM's access methods directly. Like other transaction managers, CICS can ensure that transactions are completed and, if not, undo partly completed transactions so that the integrity of data records is maintained.

IBM markets or supports a CICS product for OS/390, Unix, and Intel PC operating systems. Some of IBM's customers use IBM's Transaction Server to handle e-business transactions from Internet users and forward these to a mainframe server that accesses an existing CICS order and inventory database.

CICS was not the first Transaction Processing Monitor (TPM), that honour falls to the predecessors of TPF in the airline industry. The production of CICS was influenced by these early TPM such as the SABRE project. CICS is the work of many people, but like all things started with one, Ben Riggins. Riggins saw that the many opportunities that things like the SABRE project had brought to the Airline industry could equally well be applied to the Public Utility market in the United States. PUCICS (Public Utility Customer Information Control System) was basically written to fulfill that need. CICS, as it eventually became know, first saw the light of day as a free program in 1968, although you did need to have bought one of those new, powerful 32 bit IBM 360s. It is written in assembler and was really only a terminal control program with some database capability. CICS eventually goes through several versions picking up more and more of the attributes of a TPM along the way (for example elementary Transaction Control by the early 70's).

By 1979 it was realised that the assembler based CICS with its poor multi-threaded single memory model could not continue into future, and a new CICS, the basis of the current Mainframe version, was written in a high level language. This rework was so extensive that it was incorporated gradually and it was not until the late 80's that it was finally complete when version 3 was introduced. In the early 90's the CICS Application Programming Interface was extended to the AS400 and OS/2 platforms, with the Encina Tool Kit version being announced on AIX, IBM's UNIX like operating system in 1992.

CICS Components and Transactions

CICS (Customer Information Control System) is a transaction server that runs primarily on IBM mainframe systems under z/OS and z/VSE. CICS is a transaction manager designed for rapid high-volume online processing. This processing is mostly interactive (screen-oriented), but background transactions are possible.

CICS can run online applications on the mainframe computer because CICS acts almost like a separate operating system: it manages its own memory address space, runs its own file management functions, and manages the concurrent execution of multiple transaction applications.

While CICS has its highest profile among financial institutions such as banks and insurance companies, over 90 percent of Fortune 500 companies are reported to rely on CICS (running on z/OS) for their core business functions, beside many governments. CICS is used in bank-teller applications, ATM systems, industrial production control systems, insurance applications and many other types of interactive application.

Transactions

A CICS transaction is a set of operations which together perform a task. Usually, the majority of transactions are relatively simple tasks such as requesting an inventory list or entering a debit or credit to an account. On IBM System z servers, CICS easily supports thousands of transactions per second, making it a mainstay of enterprise computing. CICS applications comprise transactions which can be written in numerous programming languages, including COBOL, PL/I, C, C++, IBM Basic Assembly Language, REXX, and Java. Each CICS program is initiated using a transaction identifier. CICS screens are sent as a construct called a map, a little program created with BMS (Basic Mapping Support) assembler macros. The end user inputs data, which is made accessible to the program by receiving a map from CICS. CICS screens may contain text that is highlighted, has different colors, and/or blinks. An example of how a map can be sent through COBOL is given below.

```
EXEC CICS
   SEND MAPSET(MPS1) MAP(MP1)
END-EXEC.
```

CICS is actually more than just one product (indeed more that one product from one company). IBM have four different code bases all of which are called CICS. In addition to CICS IBM have two further TPM, IMS and TPF. Six Transaction Process Monitors, not bad for one company, but it is rather large. Here's the, currently, exhaustive, in our estimation, list of CICSen.
Mainframe CICS : Currently available for z/OS (quondam MVS/ESA) operating system on IBM and IBM look alike mainframe architectures. Both the successors to the original 360 and the newer MPP based 390 ones are supported. The current release is now called CICS TS 3/2. This is the definitive CICS, it has a slight variant which can be used on the older VSE operating system. This CICS now has the ability to run Enterprise Java Beans (EJB) in conjunction with `ordinary' CICS programs. More can be found at the OS/390 CICS web site.
iSeries (quondam AS/400) CICS (now withdrawn) : This was found on the IBM's AS/400 series computers, but is now withdrawn.
CICS/ETK (officially TXSeries for Multiplatform) :"CICS/ETK" is not really the official name since it has various names on the different platforms that it finds itself. This is the CICS that was formally built on top of the Encina Tool Kit (ETK). It is found on various UNIXs, AIX, DIGITAL/UNIX (formally known as OSF/1, now withdrawn), HP-UX, SINIX (now withdrawn) and Solaris. It is

also latterly found in Windows NT (where it was once known as Transaction Server) and now Windows 2003. This is the CICS that ZOIS is familiar with (and specialises in) and its current release is v6.2. The oft-promised Linux version still seems to be in the works.
As part of the reorganisation that took place when IBM fully purchased Transarc responsibility for this particular CICS moved to Transarc. It was then bundled with Encina itself (not just the Tool Kit) and the resultant package called TXSeries. It was, for a time, further bundled up in Websphere Enterprise Edition (WSEE), IBM's major offering in the Enterprise Java Beans (EJB) server market. Subsequently the Encina component was withdrawn and The IBM TXSeries web site has more.
CICS/Small-Systems (now sort-of withdrawn) : This isn't a fully functional Transaction Monitor in the true sense, it makes a good job of implementing the API but cannot participate in Transactions (Logical Units of Work). It was available on OS/2 and on Windows NT, where it was the Official CICS for NT prior to version 4. This CICS made a comeback as part of the Visual Age for COBOL development suite(on Windows platforms).

CICS is a client/server system, with clients making requests of servers. In CICS the servers are known as Application Servers or AS. These are individual operating system processes running under the control of the Application Manager or AM. The AS's are grouped into domains known as Regions. The Regions support a number of Programs known as Transaction Programs, or simply (and confusingly for the non-CICS person as Transactions). These programs are loaded dynamically into the Application Servers upon demand and are identified by often cryptic four letter names (four letters is the maximum).
CICS's Transactions are managed by the TRAN component of the Encina Tool Kit and thus it is the individual ASs which track the Transactions in a separate thread of control, much as Encina. The Application Servers are permanently connected to the Resource Managers (such as databases). With the retirement of Encina the TRAN component has been internalised and is now known as CICS/OLTP.

Introduction to JCL

Job Control Language (JCL) is a scripting language used on IBM mainframe operating system to instruct the system on how to run a batch job or start a subsystem. The term "Job Control Language" can also be used generically to refer to all languages which perform these functions, such as Burroughs' WFL and ICL's OCL. This article is specifically about IBM's JCL.

JCL (job control language) is a language for describing jobs (units of work) to the MVS, OS/390, and VSE operating systems, which run on IBM's S/390 large server(mainframe) computers. These operating systems allocate their time and space resources among the total number of jobs that have been started in the computer. Jobs in turn break down into job steps. All the statements required to run a particular program constitute a job step. Jobs are background (sometimes called batch) units of work that run without requiring user interaction (for example, print jobs). In addition, the operating system manages interactive (foreground) user requests that initiate units of work. In general, foreground work is given priority over background work.

One IBM manual compares a set of JCL statements to a menu order in a restaurant. The whole order is comparable to the job. Back in the kitchen, the chefs divide the order up and work on individual dishes (job steps). As the job steps complete, the meal is served (but it has to be served in the order prescribed just as some job steps depend on other job steps being performed first).

JCL statements mainly specify the input data sets (files) that must be accessed, the output data set to be created or updated, what resources must be allocated for the job, and the programs that are to run, using these input and output data sets. A set of JCL statements for a job is itself stored as a data set and can be started interactively. MVS and OS/390 provide an interactive menu-like interface, ISPF, for initiating and managing jobs.

In MVS and OS/390, the part of the operating system that handles JCL is called the Job Entry Subsystem (JES). There are two versions, JES2 and a later version with additional capabilities, JES3.

There are actually 2 IBM JCLs: one for the operating system lineage that begins with DOS/360 and whose latest member is z/VSE; and the other for the lineage from OS/360 to z/OS. They share some basic syntax rules and a few basic concepts, but are otherwise very different.

Introduction to VSAM

Virtual Storage Access Method - VSAM - is a data management system introduced by IBM in the 1970s as part of the OS/VS1 and OS/VS2 operating system. Although there are still datasets that are best managed with the several other (non-VSAM) data management methods, VSAM is a major component of modern IBM operating systems.

Virtual storage access method (VSAM) is an IBM disk file storage access method, first used in the OS/VS2 operating system, later used throughout the Multiple Virtual Storage (MVS) architecture and now in z/OS. Originally a record-oriented filesystem, VSAM comprises four data set organizations: Key Sequenced Data Set (KSDS), Relative Record Data Set (RRDS), Entry Sequenced Data Set (ESDS) and Linear Data Set (LDS). The KSDS, RRDS and ESDS organizations contain records, while the LDS organization (added later to VSAM) simply contains a sequence of bytes with no intrinsic record structure.

IBM uses the term data set in official documentation as a synonym of file, and DASD instead of disk drive. VSAM records can be of fixed or variable length. They are organised in fixed-size blocks called Control Intervals (CIs), and then into larger divisions called Control Areas (CAs). Control Interval sizes are measured in bytes — for example 4 kilobytes — while Control Area sizes are measured in disk tracks or cylinders. Control Intervals are the units of transfer between disk and computer so a read request will read one complete Control Interval. Control Areas are the units of allocation so, when a VSAM data set is defined, an integral number of Control Areas will be allocated.

The Access Method Services utility program IDCAMS is commonly used to manipulate ("delete and define") VSAM data sets. Custom programs can access VSAM datasets through data definitions (DDs) in Job Control Language (JCL) or in online regions such as in Customer Information Control Systems (CICS).
Both IMS/DB and DB2 are implemented on top of VSAM and use its underlying data structures.

Concepts and Facilities

VSAM was, by several accounts, intended to replace all of the earlier data management systems in use by IBM's operating systems. Conventional (non-VSAM) access methods generally provide only a single type of dataset organization. VSAM provides three:

- Key Sequenced Data Set (KSDS), where each record is identified for access by specifying its key value - a sequence of characters embedded in each data record which uniquely identify that record from all other records in the dataset. KSDS datasets are similar to Indexed Sequential Access Method (ISAM) datasets, with many of the same characteristics, but also having distinct advantages over ISAM.
- Entry Sequenced Data Set (ESDS), where each record is identified for access by specifying its physical location - the byte address of the first data byte of each record in relationship to the beginning of the dataset. ESDS datasets are similar to Basic Sequential Access Methid (BSAM) or Queued Sequential Access Method (QSAM) datasets.
- Relative Record Data Set (RRDS), where each record is identified for access by specifying its record number - the sequence number relative to the first record in the dataset. RRDS datasets are similar to Basic Direct Access Method (BDAM) datasets.

VSAM datasets are frequently referred to as clusters. A KSDS cluster consists of two physical parts, an index component, and a data component. ESDS and RRDS clusters consist of only a single component, the data component.

VSAM Catalogs

VSAM maintains extensive information about data sets and the DASD space they occupy. It is kept along with a wide range of other information in a central location called the VSAM catalog.

Master Catalog

VSAM maintains extensive information about data sets and the DASD space they occupy. It is kept along with a wide range of other information in a central location called the VSAM catalog. The VSAM catalog resides on a direct-access device. Some of the data set and data space related information it typically contains is indicated below.

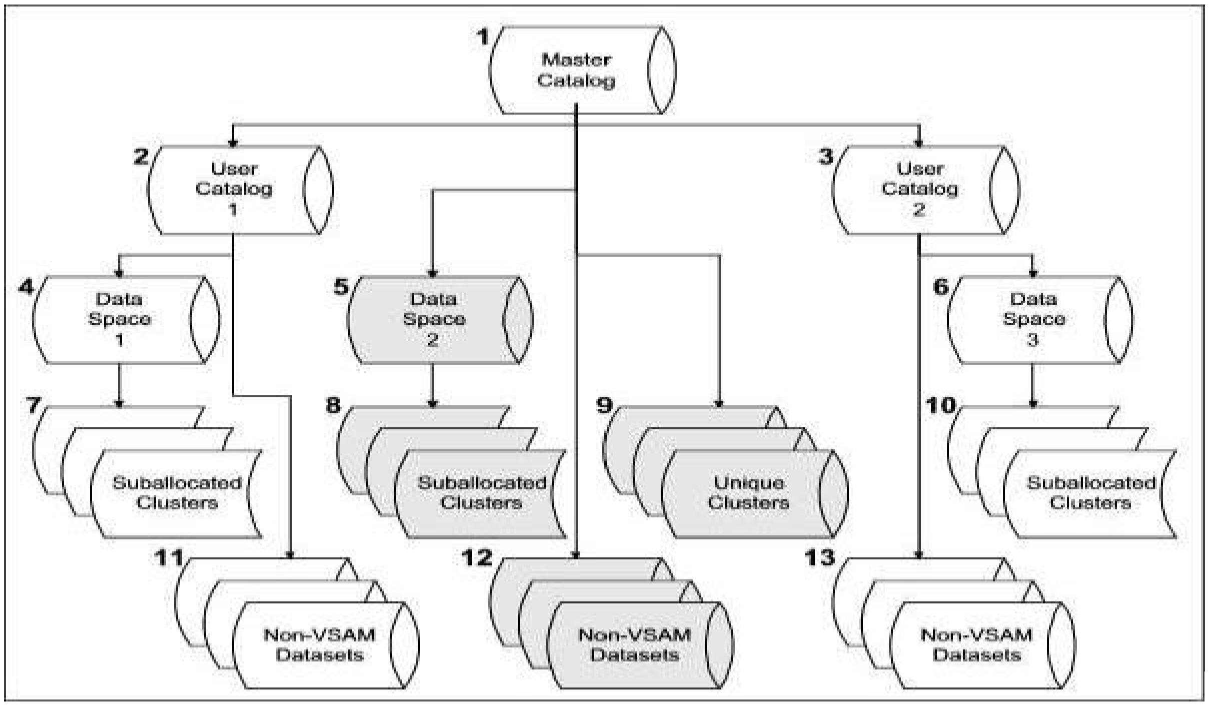
- Location of DASD space reserved for the exclusive use of VSAM
- Amount of space allocated to different VSAM data sets
- Addresses of VSAM data sets
- Attributes of VSAM data sets
- Statistical information about VSAM data sets

The VSAM catalog is very important because a VSAM data set cannot exist until it is first defined in the catalog. This means that information about the data set is recorded in the catalog in the form of entries. Also, a VSAM data set cannot be accessed unless information about it can be obtained from the catalog. Therefore, the catalog must be available for use.

VSAM Structure

In order to access data on a disk, a programmer normally invokes an access method. By using one of the commercially available access methods the programmer is relieved of the many complex and intricate details required to store and retrieve data from disk.

Some of these details include setting record size, determination of available disk space for record storage, writing data to the disk, locating data on disk and retrieving data from the disk. Without an available access method, the programmer would be required to write enormous amounts of code in each program to write and retrieve data from a disk. The importance of choosing a powerful and reliable access method must be given high priority in the field of information processing.



VSAM Structure

The structure of VSAM evolves around four basic concepts: the master catalog, the user catalog, the data space and , of course, the file itself which is commonly referred to as a cluster. The master catalog is a required file which must be defined using the

IDCAMS program. Every file which is to be used must be under the control of the master catalog. The information stored for each file allows for VSAM access to the file, password authorization and statistics for the many operations which may be performed on the file. In addition, the master catalog contains information concerning the physical environment of the VSAM files which include the type of DASD being used and the location of and allocation of available free space. This information, of course is kept current as files are expanded, reduced, created or deleted. Through the master catalog, VSAM is capable of organizing and distributing the DASD space which has been allocated. VSAM, upon request from an application program will locate, open and close a requested file and in doing so will check passwords and insure that the master catalog information is correct as it pertains to the requested file. Since the master catalog is itself a file, it must be provided the required DLBL an EXTENT statements under DOS/VSE. During IPL, the supervisor is informed of the existence of the master catalog through the CAT command and the master catalog is always assigned the system logical unit SYSCAT. The file-id and the appropriate volume containing the master catalog must be placed on the DLBL and EXTENT statements for the file. The DLBL must specify IJSYSCT as the filename and both the DLBL and EXTENT statements are normally stored in the standard label cylinder.

User catalogs are not required in a VSAM installation but their use is encouraged to provide better data integrity and security. The user catalogs have the same structure and contain most of the same information as the master catalog but appear at a lower level then the master catalog. When user catalogs are being used, the master catalog will point to the appropriate user catalog instead of pointing to the file. The user catalog will then point to the VSAM file. One may think of this as a three level access path. As evidenced in figure 1, the file request will pass through the master catalog on the first level, the user catalog on the second level and finally finish up on the third level at the proper file. The usefulness of implementing user catalogs is that if a user catalog gets destroyed (or partially destroyed) only the files under that catalog will be affected. Therefore, the catalog can be rebuilt without affecting the vast majority of files (and users). Likewise, if the master catalog is destroyed, the user catalogs are not affected. The benefits of implementing user catalogs in a VSAM environment obviously outweigh the additional overhead costs which include extra space and a more complex file access path.

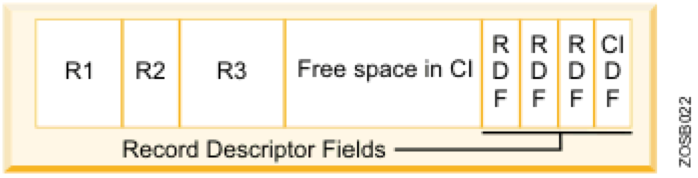
The area on a DASD volume which is allocated to VSAM is called the VSAM data space. This data space may occupy the entire volume or may simply occupy a portion of a specific volume. This space must be defined to the catalog through an Access Method Service command and must be define on the VTOC of the system. By defining this space to the catalog, you are giving VSAM complete control of this area. VSAM will then manage this space according to the requirements of the files when they are created. One interesting note is that the individual files will be defined in the appropriate VSAM catalog but each file will not appear on the VTOC. Instead, the name given to the area of the entire VSAM data space will appear on the VTOC. This area will, more than likely, contain many VSAM files

VSAM Control Interval

VSAM is a kind of record-oriented file system. In this kind of dataset, information is stored as a collection of records. VSAM records can be of any length; they need not be of one set length. They are, however, organized into blocks called Control Intervals, which are measured in bytes. These Control Intervals are further organized into Control Areas, which are measured in much larger units.

Successful processing with this program reduces these tasks from more than hour of work to only a few minutes for a disk. This program is used after the preliminary formatting of the disk is completed using the DOS disk formatting facility which makes the disk ready for VSAM catalog and dataset definitions.

VSAM works with a logical data area known as a control interval (CI) that is diagrammed in Figure 1. The default CI size is 4K bytes, but it can be up to 32K bytes. The CI contains data records, unused space, record descriptor fields (RDFs), and a CI descriptor field.



Simple VSAM control interval

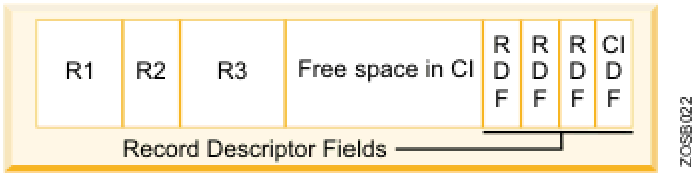
In non-VSAM data management methods, the unit of data that is moved between memory and the storage device is defined by the block. In VSAM, the unit of data that is transferred in each physical I/O operation is defined as a control interval. A control interval contains records, control information, and (in the case of KSDS clusters) possibly free space which may later be used to contain inserted records.

When a VSAM dataset is loaded, control intervals are created and records are written into them. With KSDS clusters, the entire control interval is usually not filled. Some percentage of free space is left available for expansion. With ESDS clusters, each control interval is completely filled before records are written into the next control interval in sequence. With RRDS clusters, control intervals are filled with fixed-length slots, each containing either an active record or a dummy record. Slots containing dummy records are available for use when new records are added to the dataset.

VSAM Control Area

The term Virtual Storage Access Method (VSAM) applies to both a data set type and the access method used to manage various user data types.As an access method, VSAM provides much more complex functions than other disk access methods. VSAM keeps disk records in a unique format that is not understandable by other access methods. VSAM is used primarily for applications. It is not used for source programs, JCL, or executable modules. VSAM files cannot be routinely displayed or edited with ISPF. You can use VSAM to organize records into four types of data sets: key-sequenced, entry-sequenced, linear, or relative record. The primary difference among these types of data sets is the way their records are stored and accessed.

VSAM works with a logical data area known as a control interval (CI) that is diagrammed in Figure 1. The default CI size is 4K bytes, but it can be up to 32K bytes. The CI contains data records, unused space, record descriptor fields (RDFs), and a CI descriptor field.



Simple VSAM control interval

Multiple CIs are placed in a control area (CA). A VSAM data set consists of control areas and index records. One form of index record is the sequence set, which is the lowest-level index pointing to a control interval.

VSAM is a kind of record-oriented file system. In this kind of dataset, information is stored as a collection of records. VSAM records can be of any length; they need not be of one set length. They are, however, organized into blocks called Control Intervals, which are measured in bytes. These Control Intervals are further organized into Control Areas, which are measured in much larger units.

Control intervals are grouped together into control areas. The rules used for filling and writing control areas are similar to those which apply for control intervals. For ESDS and RRDS clusters, control areas are filled with control intervals that contain records. For KSDS clusters, some of the control intervals in each control area may consist entirely of free space that can be used for dataset expansion.

JES : Job Entry Subsystem

Job Entry Subsystem (JES) is a subsystem of the OS/390 and MVS mainframe operating systems that manages jobs (units of work) that the system does. Each job is described to the operating system by system administrators or other users in job control language (JCL). The operating system then sends the job to the JES program. The JES program receives the job, performs the job based on priority, and then purges the job from the system.

There are two versions, JES2 and JES3. JES3 allows central control of the processing of jobs using a common work queue. Both OS/390 and MVS provide an interactive menu for initiating and managing jobs.

JES2 (Job Entry Subsystem 2) is descended from HASP, the Houston Automated Spooling Program, developed by the programmers of IBM as self-initiative and eventually owned and supported by IBM for NASA in the mid 1960s. JES3 (Job Entry Subsystem 3) is similarly descended from the Attached Support Processor (ASP), which was IBM's initially-preferred system for OS/360 "unit record I/O". In the 1970s a notable installation of ASP was at Princeton University controlling an IBM 360/91 mainframe.

JES3 has more network style dependency than JES2; as networking and inter-system dependencies have developed, this has become more practical than the single platform environment and single task processes that JES2 addresses.

HASP is defined as: a computer program that provides supplementary job management, data management, and task management functions such as: scheduling, control of job flow, and spooling. HASP remains within JES2 subsystem as the prefix of most module names and the prefix of all messages sent by JES to the operator. JES2 is a functional extension of the HASP II program that receives jobs into the system and processes all output data produced by the job.

Simply stated, JES is a task that runs under MVS which provides the necessary functions to get jobs into, and output out of, the MVS system, and to control the scheduling of their execution. It is designed to provide efficient spooling, scheduling, and management facilities for the MVS operating system. But none of this explains why MVS needs JES. Basically, by separating job processing into a number of tasks, MVS operates more efficiently. At any point in time, the computer system resources are busy processing the tasks for individual jobs, while other tasks are waiting for those resources to become available. In its most simple view, MVS divides the management of jobs and resources between the JES and the base control program of MVS. In this manner, the JES manages jobs before and after running the program; the base control program manages them during processing.