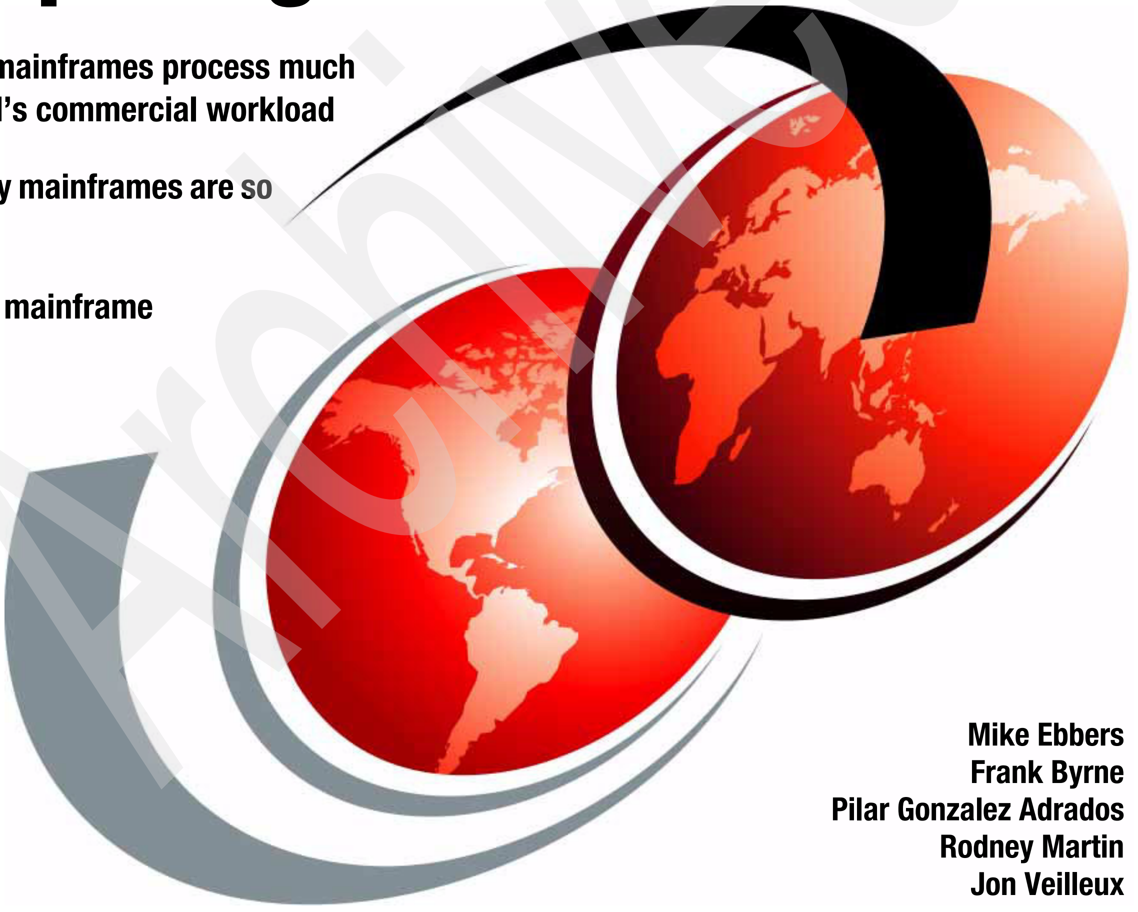
IBM

# Introduction to the New Mainframe: Large-Scale Commercial Computing

Learn why mainframes process much of the world's commercial workload

Find out why mainframes are so reliable

Understand mainframe popularity



Mike Ebbers
Frank Byrne
Pilar Gonzalez Adrados
Rodney Martin
Jon Veilleux

Redbooks

ibm.com/redbooks

International Technical Support Organization

# Introduction to the New Mainframe: Large-Scale Commercial Computing

December 2006

SG24-7175-00

**Note:** Before using this information and the product it supports, read the information in "Notices" on page 207.

**First Edition (December 2006)**

# Contents

# Preface

Today, mainframe computers play a central role in the daily operations of most of the world's largest corporations. While other forms of computing are used in business in various capacities, the mainframe occupies a prominent place in today's e-business environment. In banking, finance, health care, insurance, utilities, government, and a multitude of other public and private enterprises, the mainframe computer continues to provide the foundation of large-scale computing to modern business.

The reasons for mainframe use are many, but generally fall into one or more of the following categories:

- Capacity
- Scalability
- Integrity and security
- Availability
- Access to large amounts of data
- System management
- Autonomic capabilities

The mainframe owes much of its popularity and longevity to its inherent reliability and stability, a result of continuous technological advances since the introduction of the IBM® System/360™ in 1964. No other computer architecture in existence can claim as much continuous, evolutionary improvement, while maintaining compatibility with existing applications.

This IBM Redbook is designed for readers who already possess a basic knowledge of mainframe computing, but need a clearer understanding of how the concepts ranging from capacity through to autonomic capabilities relate to mainframe planning, implementation, and operation.

For readers who need more introductory information about mainframe concepts, usage, and architecture, we recommend that you complete *Introduction to the New Mainframe: z/OS Basics*, SG24-6366, prior to starting this book. And for more detailed information about z/OS programming topics, refer to the 11-volume IBM Redbook series *ABCs of z/OS System Programming*. All of these publications are available at the following site:

`http://www.redbooks.ibm.com`

## How this text is organized

In this document, we use simplified examples and focus mainly on basic system functions. Hands-on exercises are provided to help you explore the mainframe style of computing

## How each chapter is organized

Each chapter follows a common format:

- Objectives for the reader
- Topics that teach a central theme related to mainframe computing
- Summary of the main ideas of the chapter
- A list of key terms introduced in the chapter
- Questions for review to help readers verify their understanding of the material

## About the authors

**Mike Ebbers** has worked for IBM for 33 years, specializing in mainframe technology. He produces textbooks and IBM Redbooks™ about mainframes, which are available at the following site:

http://www.ibm.com/mainframes/

**Frank Byrne** is a Senior IT Specialist in the United Kingdom who provides day-to-day technical guidance and advice on the usage of z/OS and associated products. He has 39 years of experience in large systems support. His areas of expertise include Parallel Sysplex implementation and DFSMS. Frank has written extensively about Parallel Sysplex usage.

**Pilar Gonzalez Adrados** is an IT Specialist with IBM in Spain, and has more than 25 years of experience in mainframe technology. She has been an IT Education Specialist as well as a System Engineer. Pilar's areas of expertise include z/OS® system management, mainframe hardware, and Parallel Sysplex®.

**Rodney Martin** is a Senior Infrastructure Operator with IBM in Sydney, Australia. He joined IBM eight years ago, and has been operationally involved with MVS™ and z/OS systems for 18 years. Rodney has developed operating procedures and training plans for both IBM Australia and other Australian commercial companies.

**Jon Veilleux** is a Senior Systems Engineer at Aetna Inc. in Middletown, CT. He has 26 years of experience in z/OS installation, support, and problem resolution. Jon's responsibilities include training junior systems programmers and participating in the problem resolution swat team. Jon is Aetna's infrastructure representative to the IBM zSeries® Business Leaders Council.

## Acknowledgements

**Edward Baker** is a Senior Software Engineer with IBM Systems and Technology Group Lab Services. He has more than 28 years of experience in z/OS software development. Ed’s areas of expertise include DFSMS, DFSMShsm™, and DFSMSdss™, as well as various IBM storage hardware and software products.

**Myriam Duhamel** is an IT Specialist in Belgium. She has 20 years of experience in application development and has worked at IBM for 15 years. Her areas of expertise include development in different areas of IBM System z™. Myriam currently teaches courses in DB2® and WebSphere® MQ.

**Andreas Gallus** is an Advisory IT Architect for IBM Global Business Services in Hamburg, Germany. He more than seven years of experience with mainframe technology, along with Linux® for zSeries and z/OS. His areas of expertise include infrastructure architectures and server consolidation.

**Michael Grossmann** is a Systems Engineer and IT Education Specialist in Germany. He has 12 years of experience as a z/OS Systems Programmer and Instructor. Michael’s areas of expertise include z/OS education for beginners, z/OS operations, automation, mainframe hardware and Parallel Sysplex.

**John Kettner** is a Consulting Software Architect in the zSeries Advanced Architecture Group. He has 30 years of mainframe experience and holds a BS in Computer Science from L.I.U. His areas of expertise include zSeries internals, WebSphere product integration, and capacity planning. John has written several IBM Redbooks and contributes to various education programs throughout IBM.

**Bianca Potthast** is a Systems Programmer at LVM Versicherungen, Germany. She has 14 years of experience with z/OS and Storage Systems. Bianca’s areas of expertise include installation and configuration of hardware and storage.

## Reviewers

**Olegario Hernandez,** Chile

**Seymour Metz**, New York, USA

**Joe Macera**, California, USA

1

# The new mainframe

**Objective:**

This chapter explains what the term *mainframe* means and describes how they are used.

After completing this chapter, you will be able to do the following:

- Explain why 60% of the world's business data resides on mainframes
- List typical uses of mainframes
- Describe the benefits of mainframe computing
- Outline the major types of workloads for which mainframes are best suited

## 1.1 What is a mainframe?

Today, the term *mainframe* can be used to describe a *style* of operation, applications, and operating system facilities. Here is a working definition: "A mainframe is what businesses use to host their commercial databases, transaction servers, and applications that require a greater degree of security and availability than is commonly found on smaller-scale machines."

A mainframe is the central data repository or *hub* in a corporation's data processing center, linked to users through less powerful devices such as workstations or terminals. The presence of a mainframe often implies a centralized form of computing, rather than a distributed form of computing. Having data centralized in a single mainframe repository saves users from having to manage updates to more than one copy of their business data. This increases the likelihood that the data is current and has integrity, because there is only *one* version of data.

Early mainframe systems were housed in enormous, room-filling metal boxes or frames, and this is probably how the term "mainframe" originated. The mainframe required large amounts of electrical power and air-conditioning, and the room was occupied mostly by input/output (I/O) devices. Also, a typical installation had several mainframes installed with most of the I/O devices connected to all of the mainframes. During their largest period in terms of physical size, a typical mainframe occupied 2,000 to 10,000 square feet (600 to 3,000 square meters), with some installations being much larger than this.

**Mainframe** Hosts the databases, transaction servers, and applications that require a great degree of security and availability.

Starting around 1990, mainframe processors and most of the I/O devices became physically smaller, while their functionality and capacity continued to grow. Mainframe systems today are much smaller than earlier ones—about the size of a large refrigerator. Furthermore, it is now possible in some cases to run a mainframe operating system on a personal computer that emulates an IBM System z processor. Such emulators are useful for developing and testing business applications before moving them to a mainframe production system.

Clearly, the term mainframe has expanded beyond merely describing the physical characteristics of a system. Instead, the word typically applies to some combination of the following attributes:

- Compatibility with mainframe operating systems, applications, and data.
- Centralized control of resources.
- Hardware and operating systems that can share access to disk drives with other systems, with automatic locking and protection against destructive simultaneous use of disk data.
- A style of operation, often involving dedicated operations staff who use highly organized procedures for backup and recovery, training, and disaster recovery at an alternate site.
- Hardware and operating systems that routinely work with hundreds or thousands of simultaneous I/O operations.
- Clustering technologies that allow the client to operate multiple copies of the operating system as a single system.

  This configuration, known as Parallel Sysplex, is analogous in concept to a UNIX® cluster, but allows for systems to be added or removed as needed while applications continue to run. This flexibility allows mainframe clients to introduce new applications, or discontinue using existing applications, in response to changes in business activity.
- Additional data and resource-sharing capabilities; in a Parallel Sysplex, for example, it is possible for users across multiple systems to access the same databases concurrently.

Advances in hardware pave the way for improvements in operating systems and applications. Likewise, software requirements influence the next generation of hardware development. So the mainframe architecture continues to evolve.

## 1.2 An evolving architecture

**Architecture** The organizational structure of a system.

An *architecture* is a set of defined terms and rules used as a blueprint to build products. In computer science, an architecture describes the organizational structure of a system. An architecture can be reduced to parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components, and subsystems.

Starting with the first large machines, which arrived on the scene in the 1950s and became known as “Big Iron” (in contrast to smaller departmental systems), each new generation of mainframe computers has added improvements in one or more of the following areas of its architecture[1]:

[1] Since the introduction of S/360™ in 1964, IBM has significantly extended the platform roughly every ten years: System/370™ in 1970, System/370 Extended Architecture (370-XA) in 1983, Enterprise Systems Architecture/390® (ESA/390) in 1990, and z/Architecture™ in 2000.

- More and faster processors
- More physical memory and greater virtual memory addressing capability
- Dynamic capabilities for upgrading both hardware and software
- More sophisticated automated hardware error-checking and recovery
- Enhanced devices for input/output (I/O) and more and faster paths (channels) between I/O devices and processors
- Sophisticated I/O attachments, such as LAN adapters with extensive inboard processing
- Increased ability to divide the resources of one machine into multiple, logically independent and isolated systems, each running its own operating system
- Enhanced clustering technologies, such as Parallel Sysplex, and the ability to share data among multiple systems.

Despite the continual change, mainframe computers remain the most stable, secure, and compatible of all computing platforms. The latest models can handle the most advanced and demanding client workloads, yet continue to run applications that were written in the 1970s or earlier.

With the expanded functions and added tiers of data processing capabilities such as Web-serving, autonomic computing, disaster recovery, and grid computing, mainframe manufacturers are seeing a growth in annual sales.

While the mainframe computer has retained its traditional, central role in the information technology (IT) organization, that role is now defined to include being the primary hub in the largest distributed networks. In fact, the Internet itself is based largely on numerous interconnected mainframe computers serving as major hubs and routers.

"I predict that the last mainframe will be unplugged on March 15, 1996."

--Stewart Alsop, Infoworld, March 1991

As the image of the mainframe computer continues to change, you might ask: Is the mainframe computer a self-contained computing environment, or one part of the puzzle in distributed computing? The answer is both: A self-contained processing center, powerful enough to process the largest workloads in one secure "footprint," but one that is also just as effective when implemented as the primary server in a corporation's distributed server farm. In effect, the mainframe computer is the ultimate server in the client-server model of computing.

## 1.3 Mainframes in our midst

Mainframes tend to be hidden from the public eye. They do their jobs dependably and with almost total reliability. They are highly resistant to most forms of insidious abuse that afflict personal computers, such as e-mail-borne viruses, Trojan horses, and the like. By performing stably, quietly, and with negligible downtime, mainframes set the standard by which all other computers should be judged. But at the same time, this lack of attention tends to allow them to fade into the background.

So, how can we explore the mainframe's capabilities in the real world? How can we learn to interact with a mainframe, learn its capabilities, and understand its importance to the business world? Major corporations are eager to hire new mainframe professionals, but there's a catch: some previous experience would help.

You would not know this from monitoring the general trends that computer professionals publicize, but mainframe computers play a central role in the daily operations of most of the world's largest corporations. While other forms of computing are also used in business in various capacities, the mainframe occupies a coveted place in today's environment, where e-business reigns. In banking, finance, health care, insurance, public utilities, government, and a multitude of other public and private enterprises, the mainframe computer continues to form the foundation of modern business.

Why has this one form of computing taken hold so strongly among the world's largest corporations? In this chapter, we look at the reasons why mainframe computers continue to be the popular choice for large-scale business computing.

## 1.4 Who uses mainframe computers?

Just about everyone has used a mainframe computer at one point or another, whether they realize it or not. For example, if you have ever used an automated teller machine (ATM) to interact with your bank account, you have used a mainframe.

In fact, until the mid-1990s, mainframes provided the *only* acceptable way of handling the data processing requirements of a large business. These requirements were then (and are often now) based on large and complex batch jobs, such as payroll and general ledger processing.

The mainframe owes much of its popularity and longevity to its inherent reliability and stability, a result of careful and steady technological advances since IBM introduced System/360 in 1964. No other computer architecture can claim as much continuous improvement while maintaining compatibility with previous releases.

Because of these design strengths, the mainframe is often used by IT organizations to host their important *mission-critical* applications. These applications typically include client order processing, financial transactions, production and inventory control, payroll, as well as many other types of work.

Many of today's busiest Web sites store their production databases on a mainframe host. New mainframe hardware and software are ideal for Web transactions because they are designed to allow huge numbers of users and applications to rapidly and simultaneously access the same data without interfering with each other. This security, scalability, and reliability is critical to the efficient and secure operation of contemporary information processing.

Corporations use mainframes for applications that depend on scalability and reliability. For example, a banking institution could use a mainframe to host the database of its client accounts, for which transactions can be submitted from any of thousands of ATM locations worldwide.

Businesses today rely on the mainframe to:

- Perform large-scale transaction processing (up to thousands of transactions per second)
- Support thousands of users and application programs concurrently accessing numerous resources
- Manage terabytes of information in databases
- Handle large-bandwidth communications

The roads of the information superhighway often lead to a mainframe.

## 1.5 Factors contributing to mainframe use

The reasons for the use of mainframes are many, but most generally fall into one or more of the following categories:

- Capacity
- Scalability
- Integrity and security
- Availability
- Access to large amounts of data
- Systems management
- Autonomic capabilities

In the following sections we highlight each of these categories. They are covered in more detail later in this publication.

## 1.5.1 Capacity

As computing power has increased, with a corresponding decrease in cost, the services provided by large commercial organizations have changed dramatically. The banking industry was an early user of computers, but the functions they performed—such as check clearing—were not visible to the client. The industry now has ATMs that replace tellers and allow 24-hour banking, and Internet banking, which enables clients to directly manage their accounts from the comfort of their homes. By establishing profiles of their clients, banks are able to target products to specific individuals, which is very cost effective.

In order to provide these services to clients, generate the reports required to run the business, and to profile clients, you have to use a system with sufficient capacity. The problem is how to define capacity in terms of the hardware and software of a computing system.

The following analogy with a water supply system will help you identify some aspects of "capacity" as used in the computer industry. The capacity of a reservoir is obviously the amount of water that it can contain. However, this is merely a definition of its storage capacity. In order to supply homes and businesses with water, pipes and pumps need to be added. The capacity of a pump is the amount of liquid water that can be moved in a particular time. The capacity of a pipe is the amount of liquid that can be passed through it. The number of clients that can be served by the reservoir, that is, its capacity, is determined by how much water can be supplied and how quickly. If the demand is greater than the capability to supply, then the pressure will drop.

The reservoir in a large-scale computing system is the corporate data and the clients for this data are programs.

The z/OS software and IBM System z server hardware combine to provide the necessary power to store and transmit large volumes of data and process that data as required, in a timely manner.

## 1.5.2 Scalability

**Scalability**
The ability of a system to retain performance levels when adding processors, memory, and storage.

By *scalability*, we mean the ability of the hardware, software, or a distributed system to continue to function well as it is changed in size or volume; for example, the ability to retain performance levels when adding processors, memory, and storage. A scalable system can efficiently adapt to work, with larger or smaller networks performing tasks of varying complexity.

As a company grows in employees, clients, and business partners, it usually needs to add computing resources to support business growth. One approach is to add more processors of the same size, with the resulting overhead in managing this more complex setup. Alternatively, a company can consolidate its many smaller processors into fewer, larger systems. Using a mainframe system, many companies have significantly lowered their total cost of ownership (TCO), which includes not only the cost of the machine (its hardware and software) but also the cost to run it.

IBM System z mainframes exhibit scalability characteristics in both hardware and software, with the ability to run multiple copies of the operating system software as a single entity called a *system complex* or *sysplex*.

## 1.5.3 Integrity and security

One of the most valuable resources of a company is its data: client lists, accounting data, employee information, and so on. This critical data needs to be securely managed and controlled, and simultaneously made available to users authorized to see it. Mainframe computers have extensive capabilities to simultaneously share the data among multiple users, and still protect it.

In an IT environment, *data security* is defined as protection against unauthorized access, transfer, denial of service, modification, or destruction, whether accidental or intentional. To protect data and to maintain the resources necessary to meet the security objectives, clients typically add a sophisticated security manager product to their mainframe operating system. The client's security administrator often bears the overall responsibility for using the available technology to transform the company's security policy into a usable plan.

A secure computer system prevents users from accessing or changing any objects on the system, including user data, except through system-provided interfaces that enforce authority rules. Mainframe computers can provide a very secure system for processing large numbers of heterogeneous (or mixed) applications that access critical data. The "z" in the name of IBM System z stands for zero downtime, indicating that it is always available.

## 1.5.4 Availability

*Reliability, Availability,* and *Serviceability* (often grouped together as RAS) have always been important in data processing. When we say that a particular computer system "exhibits RAS characteristics," we mean that its architecture places a high priority on the system remaining in service at all times. Ideally, RAS is a central design feature of all aspects of a computer system, including the applications.

RAS has become accepted as a collective term for many qualities of hardware and software that are prized by users of mainframes. The terms are defined as follows:

**Reliability** The system's hardware components have extensive self-checking and self-recovery. The system's software reliability results from extensive testing and the ability to make quick updates for detected problems.

**Availability** The system can recover from a failed component without impacting the rest of the running system. This applies to hardware recovery (by automatically replacing failed elements with spares) and software recovery (layers of error recovery provided by the operating system).

**Serviceability** The system can determine why a failure occurred. This allows for the replacement of elements (hardware and software), while impacting as little of the operational system as possible. It also implies well-defined units of replacement, either hardware or software.

A computer system is "available when its applications are available. An available system is one that is reliable, that is, it rarely requires downtime for upgrades or repairs. And, if the system is brought down by an error condition, it must be serviceable—easy to fix—within a relatively short period of time.

Mean time between failures (MTBF) refers to the availability of a computer system. The mainframe and its associated software have evolved to the point that clients often experience *years* of system availability between system downtimes. Moreover, when the system is unavailable because of an unplanned failure or a scheduled upgrade, this period is typically very short. The remarkable availability of the system for processing the organization's mission-critical applications is vital in today's 24-hour, global economy. Along with the hardware, mainframe operating systems exhibit RAS through such features as storage protection and a controlled maintenance process.

Beyond RAS, a state-of-the-art mainframe system might be said to provide *high availability* and *fault toleranc*e. Redundant hardware components in critical paths, enhanced storage protection, a controlled maintenance process, and system software designed for unlimited availability all help to ensure a consistent, highly available environment for business applications in the event that a system component fails. Such an approach allows the system designer to minimize the risk of having a *single point of failure* undermine the overall RAS of a computer system.

## 1.5.5 Access to large amounts of data

The capacity to store data does not necessarily equate to the ability to process it in a timely manner. Tape media is capable of supporting sequential processing of data, but is of no value for random processing.

Disk media is capable of supporting sequential or random processing, but there are other considerations when high volumes of requests to read or write data are made. Very large disks are not generally a good solution because this usually results in the access mechanism being overloaded and queuing to occur. A larger number of smaller disks gives better response times to I/O requests.

In all cases, there has to be sufficient bandwidth to allow data to be transferred between main storage and the I/O device. The size of the volumes depends also on the kind of data to be stored and the kind of application that needs access to the data. A very careful investigation about the best disk layout should be made for each individual installation.

### 1.5.6 Systems management

*Systems management* is a collection of disciplines to monitor and control a system's behavior. These cover such areas as performance, workload, configuration, operations, problem management, network, storage, security, and change management techniques. Some of these functions are performed by the operating system or appropriate subsystems, which are provided in specialized tools marketed by various software companies. Good systems management plays a vital part in the reliability of mainframes.

### 1.5.7 Autonomic capabilities

The term "autonomic" comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations automatically without any external help. Similarly, a good way to handle IT complexity is to create computer systems that can respond to changes in their environment, so the systems can adapt, heal, and protect themselves. Only then will the need be reduced for constant human maintenance, fixing, and debugging of computer systems.

## 1.6 Typical mainframe workloads

Most mainframe workloads fall into one of two categories: batch processing, or online transaction processing, including Web-based applications (see Figure 1-1 on page 11).

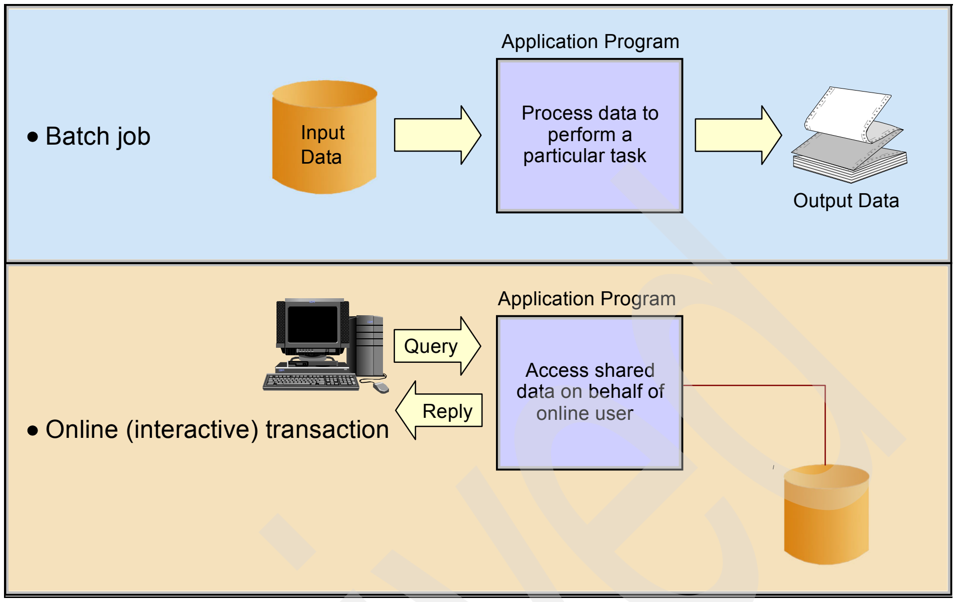

Batch job
Input Data
Application Program
Process data to perform a particular task
Output Data
Online (interactive) transaction
Query
Reply
Application Program
Access shared data on behalf of online user

*Figure 1-1 Typical mainframe workloads*

These workloads are discussed in several chapters in this document; the following sections provide an overview.

## 1.6.1 Batch processing

A key advantage of mainframe systems is the ability to process terabytes of data from high-speed storage devices and produce valuable output. For example, mainframe systems make it possible for banks and other financial institutions to produce end-of-quarter processing in an acceptable time frame (runtime) when such reporting is necessary to clients (such as quarterly stock statements or pension statements) or to the government (financial results). With mainframe systems, retail stores can generate and consolidate nightly sales reports for review by regional sales managers.

An equivalent concept can be found in a UNIX script file or Windows® CMD file, but these facilities do not provide the same level of control and I/O performance that is available on a z/OS system.

The applications that produce these statements are *batch* applications—they are processed on the mainframe without user interaction. A *batch job* is submitted on the computer, which reads and processes data in bulk—perhaps terabytes of data. A batch job may be part of a group of batch jobs that need to process in

**Batch work**
Data is processed on the mainframe without user interaction.

sequence to create a desired outcome. This outcome may be output such as client billing statements.

z/OS provides a robust suite of scheduling and control software that is not generally available on other platforms. This software, along with built-in workload management, gives z/OS the ability to run and prioritize mixed workloads (batch and online) efficiently, based on user-specified criteria.

While batch processing is possible on distributed systems, it is not as commonplace as on mainframes, because distributed systems often lack the following elements:

- Sufficient data storage
- Available processor capacity (*cycles*)
- I/O bandwidth
- Sysplex-wide management of system resources and job scheduling

Mainframe operating systems are usually equipped with sophisticated job scheduling software that allows data center staff to submit, manage, and track the execution and output of batch jobs[2].

Batch processes usually have the following characteristics:

- Large amounts of input data are processed and stored (perhaps terabytes or more), large numbers of records are accessed and updated, and a large volume of output is produced.
- Interactive response time is usually not the primary requirement. However, batch jobs often must complete within a “batch window,” which is a period that typically has less intensive online activity prescribed by a service level agreement.
- Information is generated about large numbers of users or data entities (for example, client orders or a retailer’s stock on hand).
- A scheduled batch process can consist of the execution of hundreds or thousands of jobs in a pre-established sequence.

During batch processing, multiple types of work can be generated. Consolidated information such as profitability of investment funds, scheduled database back-ups, processing of daily orders, and updating of inventories are common examples. Figure 1-2 on page 13 shows a number of batch jobs running in a typical mainframe environment.

[2] In the early days of the mainframe, punched cards were often used to enter jobs into the system for execution. “Keypunch operators” used card punches to enter data, and decks of cards (or batches) were produced. These were fed into card readers, which read the jobs and data into the system. As you can imagine, this process was cumbersome and error-prone. Today, it is possible to do a file transfer (FTP), the equivalent of punched card data to the mainframe in a PC text file, or to enter the data directly on the mainframe.

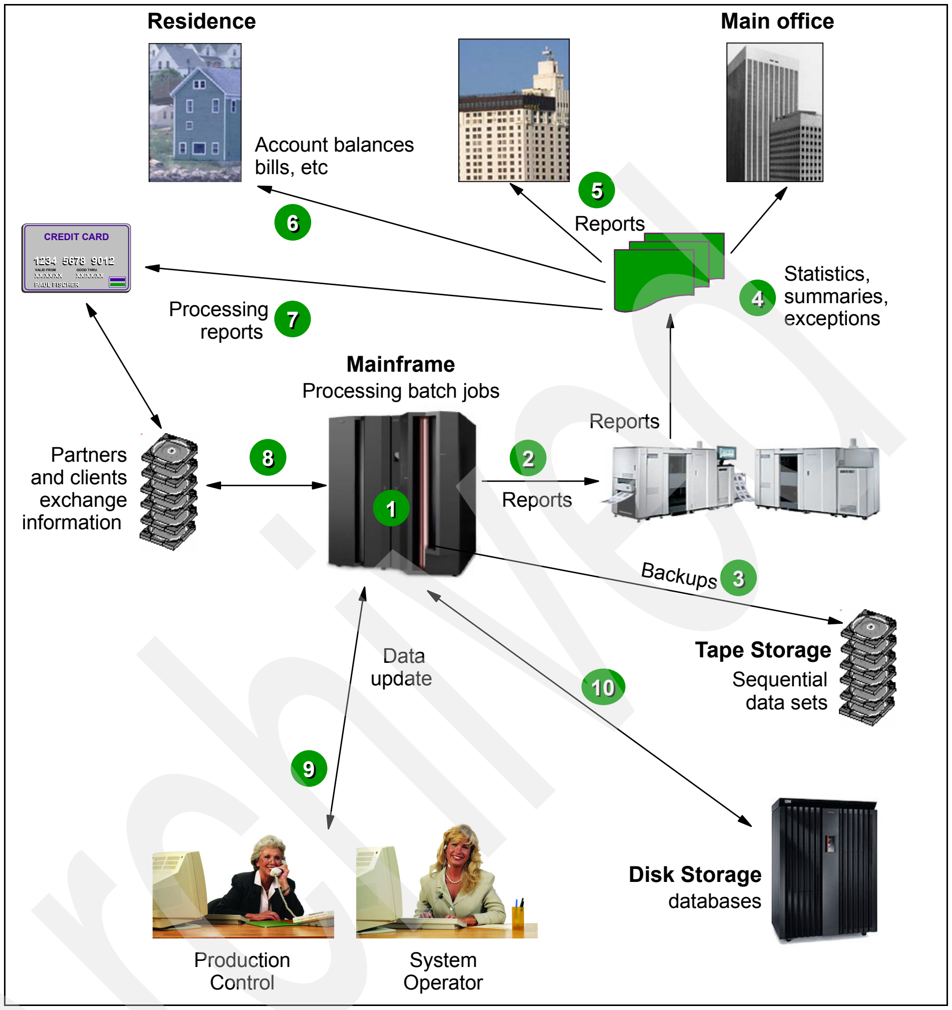


*Figure 1-2 Typical batch use*

In Figure 1-2, consider the following elements at work in the scheduled batch process in a banking application:

1. At night, numerous batch jobs executing programs and utilities are processed. These jobs consolidate the results of the online transactions executed during the day.
2. The batch jobs generate reports of business statistics.
3. Backups of critical files and databases are made before and after the batch window.

4. Reports with business statistics are sent to a specific area for analysis during the following day.
5. Reports with exceptions are sent to the branch offices for follow-up actions.
6. Monthly account balance reports are generated and sent to all bank clients.
7. Reports with processing summaries are sent to the partner credit card company.
8. A credit card transaction report is received from the partner company.
9. In the Production Control department, the System Operator is monitoring messages presented on the system console or an automation interface. Appropriate actions are then taken to ensure the successful execution of the batch jobs.
10. Jobs and transactions are reading or updating the database (the same database used by online transactions) and many files are written to tape.

## 1.6.2 Online transaction processing

**Online transaction processing (OLTP)** Transaction processing that occurs interactively with the end user. It requires a system with fast response time, continuous availability, security, and data integrity.

*Online transaction processing* (OLTP) is transaction processing that occurs interactively with the end user. Mainframes serve a vast number of transaction systems. These are often mission-critical applications that businesses depend on for their core functions. Transaction systems must be able to support an unpredictable number of concurrent users and transaction types. Most transactions are executed in short time periods—fractions of a second, in some cases.

One of the main characteristics of a transaction system is that the interactions between the user and the system are very brief. The user will perform a complete business transaction through brief interactions, with an acceptable response time required for each interaction. These systems are currently supporting mission-critical applications; therefore, continuous availability, high performance, and data protection and integrity are required.

Online transactions are familiar to most people. Examples include:

- ATM machine transactions such as deposits, withdrawals, inquiries, and transfers
- Supermarket payments with debit or credit cards
- Purchase of merchandise over the Internet

For example, whether from a bank branch office or using Internet banking, clients are using online services when checking an account balance or redirecting funds.

In fact, an online system has many of the characteristics of an operating system:

- Managing and dispatching tasks
- Controlling user access authority to system resources
- Managing the use of memory
- Managing and controlling concurrent access to data files
- Providing device independence

Some industry uses of mainframe-based online systems include:

- Banks - ATMs, teller systems for client service
- Insurance - Agent systems for policy management and claims processing
- Travel and transport - Airline reservation systems
- Manufacturing - Inventory control, production scheduling
- Government - Tax processing, license issuance and management

Multiple factors can influence the design of a company's transaction processing system, including:

- Number of users interacting with the system at any one time.
- Number of *transactions per second* (TPS).
- Availability requirements of the application (for example, must the application be available 24 hours a day, seven days a week, or can it be brought down briefly one night each week?)

So how might users in such industries interact with their mainframe system? Before personal computers and intelligent workstations became popular, the most common way to communicate with online mainframe applications was with 3270 terminals. Sometimes known as "dumb" terminals, they had enough intelligence to collect and display a full screen of data rather than interacting with the computer for each key stroke, thereby saving processor cycles. The characters were displayed as green text on a black screen, so the mainframe applications were nicknamed "green screen" applications. This application remains today, even though terminals with color text have been in common use for several decades.

Based on these factors, user interactions vary from installation to installation. Many installations are now reworking their existing mainframe applications to include Web browser-based interfaces for users. This work sometimes requires new application development, but it can often be done with vendor software purchased to "re-face" the application. In such cases, the end user often does not realize that there is a mainframe behind the scenes.

**Note:** There is no need to describe the *process* of interacting with the mainframe through a Web browser in this document, because it is exactly the same as any interaction a user would have through the Web. The only difference is the machine at the other end!

Online transactions usually have the following characteristics:

- Small amount of input data, few stored records accessed and processed, and small amount of data as output
- Rapid response time, usually less than one second
- Large numbers of users involved in concurrent transactions
- Round-the-clock availability of the transactional interface to the user
- Assurance of security for transactions and user data

In a bank branch office, for example, clients use online services when checking account balances or making an investment.

Figure 1-3 illustrates a series of common online transactions using a mainframe.

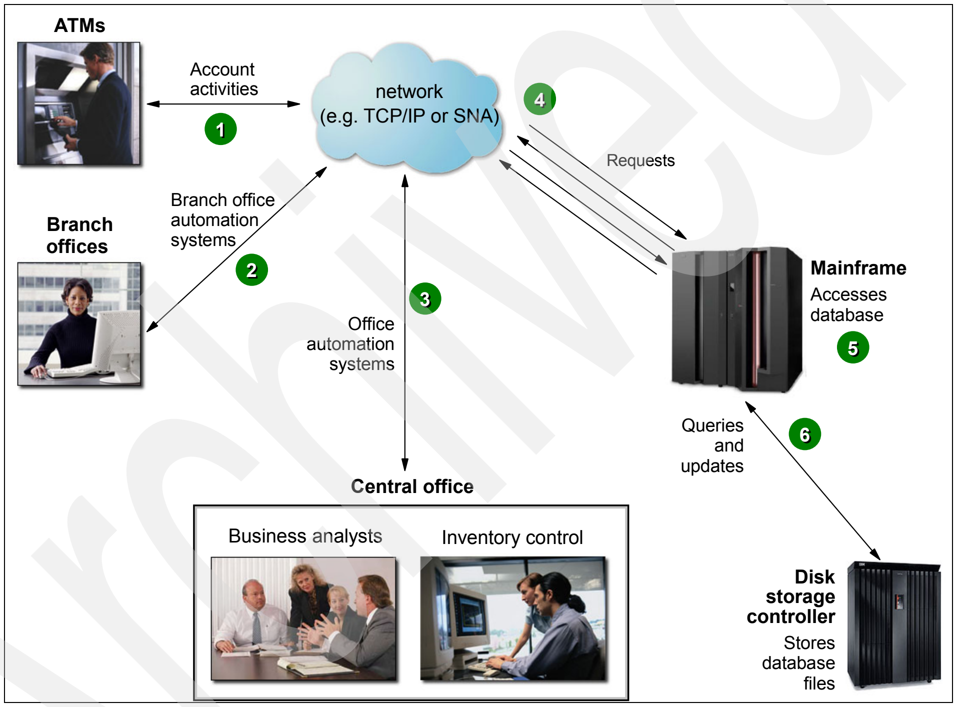


*Figure 1-3 Typical online use*

1. A client uses an ATM, which presents a user-friendly interface for various functions: withdrawal, query account balance, deposit, transfer, or cash advance from a credit card account.
2. Elsewhere in the same private network, a bank employee in a branch office performs operations such as consulting, fund applications, and money ordering.
3. At the bank's central office, business analysts tune transactions for improved performance. Other staff use specialized online systems for office automation

to perform client relationship management, budget planning, and stock control.

4. All requests are directed to the mainframe computer for processing, and a returned response.
5. Programs running on the mainframe computer perform updates and inquires to the database management system (for example, DB2).
6. Specialized disk storage systems store the database files.

## 1.7 Summary

Today, mainframe computers play a central role in the daily operations of most of the world's largest corporations. While other forms of computing are used in business in various capacities, the mainframe occupies a vital place in today's e-business environment. In banking, finance, health care, insurance, utilities, government sectors, and in a multitude of other public and private enterprises, the mainframe computer continues to provide the foundation for modern business transaction processing.The mainframe owes much of its popularity and longevity to its inherent reliability and stability, a result of continuous technological advances since the introduction of the IBM System/360 in 1964. No other computer architecture in existence can claim as much continuous, evolutionary improvement, while maintaining compatibility with existing applications.

The term *mainframe* has evolved from a physical description of large IBM computers to the categorization of a style of computing. One defining characteristic of mainframes has been continuing compatibility spanning decades.The roles and responsibilities found in a mainframe IT organization are wide-ranging and varied. It takes skilled personnel to keep a mainframe computer running smoothly and reliably. It might seem that there are far more resources needed in a mainframe environment than with small, distributed systems. But if roles are fully identified on the distributed systems side, a number of the same roles exist there, as well.

| **Key terms in this chapter** | | | | |
|---|---|---|---|---|
| Architecture | Data integrity | Infrastructure | Punched card | Security |
| Batch processing | High availability | Mainframe | RAS | Systems management |
| Compatibility | IBM System z | OLTP | Scalability | Transactions per second |

## 1.8 Questions for review

To help test your understanding of the material in this chapter, complete the following review questions:

1. Explain how businesses make use of mainframe processing power, the typical uses of mainframes, and how mainframe computing differs from other forms of computing.
2. Outline the major types of workloads for which mainframes are best suited.

## 1.9 Topics for further discussion

1. What is a mainframe today? How did the term arise? Is it still appropriate?
2. What characteristics, positive or negative, exist in a mainframe processing environment because of the roles that are present in a mainframe installation? (Efficiency? Reliability? Scalability?)
3. Most mainframe installations have implemented very rigorous systems management, security, and operational procedures. Have these same procedures been implemented in distributed system environments? Why or why not?

2

# Capacity

**Objectives:**

This chapter describes how the factors that determine the capacity requirements of a large scale commercial computing environment are addressed by the IBM System z platform.

After completing this chapter, you will be able to describe the following:

- Mixed workloads
- SLA
- PR/SM™ and LPAR
- Parallel Sysplex
- Storage concepts and management
- Measuring capacity

## 2.1 Introduction

A large-scale commercial computing environment is one that exists to process very large volumes of corporate data. The following topics will use the example of the banking industry where data relating to client accounts may be processed for many reasons, including:

- Cash withdrawal from an ATM
- Check processing
- Online inquiry
- Telephone inquiry
- Statement printing
- Overnight accounting
- Data mining
- Money laundering legislation

Much of the data is also subject to regulatory controls that determine how long information must be kept.

## 2.2 What do we mean by capacity?

The term *capacity* has several definitions, including:

1. The potential or suitability for holding, storing, or accommodating
2. The facility or power to produce, perform, deploy, or simply process

A large bank most likely will have millions of clients. Many of these may hold several accounts, and the data relating to these accounts has to be available to multiple functions, which may be reading or updating it.

In order to meet the first definition, there needs to be sufficient disk storage to allow fast access—and a less expensive and secure media for long-term storage. An example of such a media is a *tape*, which is usually used because of its lower cost per byte and its portability.

To meet the second definition, there needs to be sufficient computing capacity to run the programs that will process the data. The data could be stored and processed on a single server—or dispersed and stored on many servers—as discussed in 2.4, “A few servers versus many servers” on page 21.

## 2.3 Elements of a system required for capacity

Any computer system, from a single user personal computer to a multi-thousand user IBM System z mainframe, performs work in the following stages:

1. Load a program from a storage device (normally a disk) to main storage.
2. Process the instructions in the program.
3. Load any required data from disk or tape into main storage.
4. Process the data and send the results to the desired endpoint.

This requires the following elements of hardware:

- Processors
- main storage
- Access input/output storage (for example, disk or tape). This includes also the connection to that device (channels or network).

For maximum capacity, these three elements must be in balance and have software capable of exploiting their capabilities. Because commercial environments are dynamic, with new function often being introduced, the challenge is to keep these three elements in balance. In other words, they play their roles according to the scheme illustrated in Figure 1-2 on page 13.

### The relationship between the elements

A program needs a central processor (CP) to execute its instructions. If additional CPs are added, more programs can run concurrently. Programs require main storage for themselves and their data. If more programs are to be run, more main storage is required.

Commercial programs typically process large volumes of data. The more programs that are running, the greater the bandwidth required to access that data, but also the more main storage is needed to load and process that data.

## 2.4 A few servers versus many servers

In the banking example, the storing of information relating to only one client's accounts usually will not be particularly large and could just as easily be stored and retrieved from a small distributed system as from a large central system.

## 2.4.1 Many servers

If data can be stored and processed on a single server, a large number of small servers could be a more appropriate solution than a centralized data store. If data needs to be accessed and updated from different servers, however, the following problems have to be addressed:

- Data retention may be required by law. When data is changed, does the owning system or the updating system create a copy?
- Because the data may need to be processed for both reading and updating, a data locking mechanism has to be implemented and all servers processing the data need to participate in this locking scheme.
- Connectivity has to be established between the various servers.

The system management of such a setup can be problematic and it often impacts many people.

## 2.4.2 Few servers - the IBM System z approach

To circumvent the problems concerning distributed data and system management, a more centralized approach is used in most mainframe computing centers. The IBM System z philosophy is that the best utilization of the capacity of a server is obtained by running mixed workloads.

The IBM System z architecture has evolved over many years. The design has had to adhere to the following criteria:

- Large volumes of commercial data are best held in one place.
- Applications should not need to be rewritten as new technology is introduced.
- The total system—that is, hardware, system software, and applications—must be extremely robust in the areas of reliability and availability.

The concept of running a mixed workload on a single system derives from the fact that, with modern processor chips, it is very unlikely in a commercial environment that one program could keep a processor fully utilized over a long period of time.

There are several approaches to achieving better processor utilization, such as:

- Level 1 (internal) and level 2 (external) processor cache
- Preloading of programs into main storage
- Preloading of data into main storage
- Running several copies of an application

### IBM System z input/output architecture

If large amounts of data are to be held in one place, then enough bandwidth must be provided to enable that data to be accessed in a timely manner.

The communication bus between a IBM System z server and its input/output (I/O) devices uses a channel architecture. This architecture is implemented in dedicated microprocessors that communicate across fiber optic cables (the channel) to the control units that operate the I/O devices without affecting the server.

Note that the I/O devices are *not* directly connected to the channel; one or more of them are connected to a control unit. This design allows new function to be introduced in just the microprocessor and control units. Additionally, the design has implications for scalability, as discussed in Chapter 3, “Scalability” on page 37.

The performance of an I/O control unit, such as a disk control unit, has an impact on capacity. The design and evolution of these units is discussed in Chapter 6, “Accessing large amounts of data” on page 109.

## 2.5 Mixed workloads

If large amounts of data are to be stored in one place, a large-scale commercial computing environment can be expected to provide service to the following:

- Online clients
- Online in-house users
- Batch jobs

All of these have the same basic requirement, which is to have access to processors, main storage, and disk or tape storage. The time frame in which a unit of work must be completed is the distinguishing factor.

Online clients, such as Internet users, have an expectation of fast response time. If that expectation is not met, they could switch to a competitor’s site. In-house users, such as application developers, require a responsive system in order to be productive, but their needs will not take precedence over those of the clients.

Batch jobs are generally the least demanding regarding a specific completion time, as long as they are completed at a certain time. They can usually give up resources to the work in the previous two categories.

The relationship between capacity, workload, and response time demands of each of the categories should be defined in a service level agreement, as described in the following section.

## 2.6 Service level agreement

A service level agreement (SLA) is an agreement between a service provider and a recipient, generally the server owner and a business unit. There should be several SLAs in place to cover the various aspects of the business that will be run on the server. For capacity management, having correct and precise definitions of SLAs is very important, because these SLAs are the baseline against which the capacity demands are measured and compared.

For example, an Internet application may have an SLA that defines the maximum allowable response time for an online transaction, and one that defines the time at which daily trading reports have to be available. The first of these SLAs requires sufficient capacity on the server to be able to respond to peak traffic, while the second allows the workload to run when resources are available.

Examples of terms from a service level agreement are:

- 95% of ATM transactions are completed in less than one second.
- 90% of daily reports are completed by 6 a.m.

A SLA has wider implications than just the performance of the server. Take, for example, the requirements of an ATM transaction. For a client who is withdrawing cash, the processing consists of:

1. Insert the card.
2. Type in the PIN number.
3. Select from the menu.
4. Enter the amount.
5. Collect the cash.
6. Collect the card.
7. Print a receipt.

The processing of this request involves the ATM validating the card, prompting for input, and sending the request to the server. The server has to check if the card is lost or stolen, implement a cryptographic process, and then check the account to verify whether funds are available. The withdrawal is noted but no action is taken to update the account, because the cash has not yet been dispensed. When the ATM signals that the money has been dispensed, then the account can be debited.

So for this SLA, the ATM and the communications network are involved, as well as the server.

## 2.7 Managing the system to the SLA

The SLA for a batch job is generally an agreement as to when the output from the job will be available. The earliest start time will be when the data to be processed is available.

The 24x7 applications, such as for Internet processing, should be available at all times, and the SLA will apply to the response to each client enquiry, referred to as a *transaction*.

Both the batch program and the 24x7 transaction server application are run in the same environment. Each has an address space on the same System z server. Therefore, it is important to manage the system or server resources in the most efficient way to achieve both SLAs.

### 2.7.1 Managing CPU

The z/OS Operating System provides for a sophisticated feature called Workload Manager (WLM). WLM supplies the means for managing CPU usage to meet goals or Service Level Objectives (SLOs).

Through Service Class definitions written in a rule-based *policy*, the client selects which workload is important over others. The *importance* of the workload will allow the job request to obtain more or less CPU. The CPU management for a job or workload in incorporated in the policy.

CPU usage is defined within the Service Class as Service Units (SUs). Using the policy definition, the WLM administrator can put in place the rules for managing the minimum and maximum CPU consumption for a job or workload. The WLM Policy is the overall method used to manage resource usage for a sysplex.

### 2.7.2 Managing disk

There is a unique element of z/OS called Data Facility Storage Management Subsystem (DFSMS). DFSMS provides all essential disk, storage and device management functions of the system. In a system-managed storage environment, DFSMS automates and centralizes storage administration based on the policies that an installation defines for availability, performance, space utilization, and security.

Storage management policies reduce the need for users to make multiple detailed decisions that are unrelated to their business objectives. DFSMS provides functions that reduce the occurrence of system outages, and enhance disaster recovery capabilities and system security. DFSMS also improves business efficiency by providing better system performance and throughput, as well as usability enhancements that increase storage administration productivity.

For more information about disk management, refer to Chapter 6, “Accessing large amounts of data” on page 109.

**Note:** DFSMS provides functions that reduce the occurrence of system outages, and enhance disaster recovery capabilities and system security. DFSMS also improves business efficiency by providing better system performance and throughput, and usability enhancements that increase storage administrator productivity.

## 2.7.3 Storage concepts: the address space

The z/OS software was designed to provide security and integrity, while also allowing communication between functions and access to common services. To help achieve this goal, the memory of the mainframe is divided into *address spaces*. Each program or subsystem of the z/OS operating system is loaded into its own private address space.

A program can be located in the physical memory of the system or on DASD storage. The combination of the two is called *virtual memory* or *virtual storage*. Here is a definition of virtual storage from Wikipedia, the free encyclopedia:

“Virtual storage is the storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.”[1]

The virtual storage available to a program is known as an address space, as shown in Figure 2-1 on page 27. This provides an environment for each batch job, interactive user, and transaction server.

Each address space has the potential to address up to 16 exabytes of virtual storage, but the installation standard is more likely to be about 32 megabytes (MB). Virtual storage is managed in units of 4 kilobytes (KB), known as *pages*. Pages can have attributes associated with them for security purposes. For more information about this topic, refer to Chapter 4, “Integrity and security” on page 65.

[1] See http://en.wikipedia.org/wiki/Virtual_storage

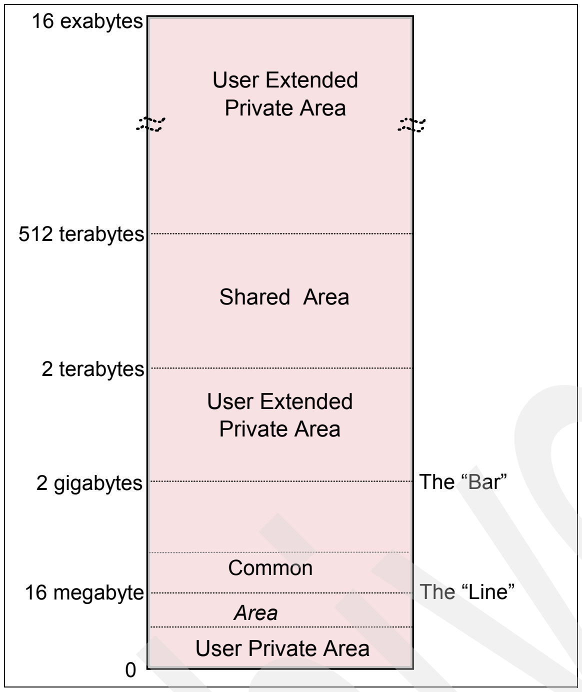


*Figure 2-1 An address space showing the “Bar” and the “Line”*

The mechanism used for associating a real address to a virtual address allows each function to have the unique use of all of the storage—apart from that marked as *common*. The common areas allow data to be viewed from every address space, and provide a way to communicate between address spaces.

The divisions of the storage must be viewed from a historical perspective in order to be meaningful. As previously mentioned, the IBM System z has evolved over many years and through many technological advances, which have increased storage addressability by many orders of magnitude. It has been a hallmark of the platform that programs which ran on the early systems will run without alteration on the most current systems.

The milestones in virtual storage addressing have been:

- 24-bit
- 31-bit
- 64-bit

The earliest ancestor of z/OS supported a $2^{24}$ addressing structure; the common areas were positioned at the topmost addressable storage, known as the *line*. The next technology leap introduced $2^{31}$ addressability (bit 0 was used for compatibility with the previous architecture, for example z/VSE™ runs in $2^{31}$ addressability); the common areas extended above the line.

The latest advance has been the introduction of addressability to $2^{64}$ and the topmost 31-bit address became referred to as the *bar*. The 24-bit and 31-bit architectures allow both program instructions and data to be in any part of the storage.

In 64-bit addressing, however, only data can reside above the bar. The rationale for this is to allow more data to be held in main storage, which will decrease the time to access it, and therefore increase the capacity of the system.

This may appear to restrict the combined size of programs in the address space to somewhat less than 2 gigabytes. However, the architecture also allows for programs to be executed, subject to security rules, in other address spaces, which lifts this restriction.

### 2.7.4 Real storage management

main storage, in this context referred to as *real storage*, is managed by a Real Storage Manager (RSM™) in elements of 4 kilobytes, known as *frames*. Each frame of real storage holds a page of virtual storage.

Real storage is not directly addressed by application programs. RSM uses tables to establish a link between a virtual address and a real address. A hardware mechanism known as Dynamic Address Translation (DAT) converts the virtual addresses seen by a program into real addresses that the CPs can access.

The majority of programs have sections that are frequently executed and have data objects that they refer to on a regular basis. The RSM, in common with other systems, tries to keep as many of the pages of programs and data that have been used in storage, as a performance aid. Eventually, all of real storage will be full of virtual pages. To accommodate new ones, the pages that have not been used recently are copied to a disk file. This is referred to as a *page-out*, and if the virtual page is referred to at a later time, it will be paged in. Avoiding page-in and page-out activity aids the performance of a program and it is therefore a capacity requirement to have sufficient real storage to achieve this.

## 2.8 Architecture, running work, and capacity

A z/OS system is capable of driving the servers' processors at 100% for sustained periods of time. This is not difficult on any system if there is sufficient resources for workload. The problem is to make sure that the high-importance units of work achieve their objectives.

One technique for achieving this is to allow the high-importance work to run until it must give up control of the CP (for example, while waiting for an I/O operation). The I/O is considered an interrupt and at this point a context switch occurs. A *context switch* is where the executing unit of work status is kept in special hold areas called register save areas (RSA). After the executing work status is saved off, the interrupt processing can begin. The RSA is used to provide a return point back to the executing program to pick up execution where it left off after the interrupt is processed.

In the case of an I/O interrupt, the I/O request may have been to retrieve a record from a data set. The return data from the I/O can then be used by the executing program and continue execution to process the information. The interrupt provides the means to give up control to other executing work so that their instruction requests can be processed; this is known as *multi-tasking*.

z/OS is able to achieve balance between processor utilization and allowing as many units of work as possible to meet their objectives, defined in WLM. z/OS does so by exploiting the hardware architecture of the IBM System z server.

The significant architectural elements of the IBM System z server are the Program Status Word (PSW) and six types of interrupts. The combination of these allows control of the work that runs on the CPs, as well as being the focal point for system security. In the following sections, we describe these elements in more detail.

## Program status word (PSW)

The program status word (PSW) is the heart of the interface between the z/OS software and the IBM System z hardware. It represents a program running on a central processor (CP), and as such, there is one PSW per active CP.

One function of the PSW is to hold the address of the instruction that is currently being executed on a CP. A program is given access to a CP by z/OS issuing a special instruction, the LOAD PSW. After this has occurred, the program is allowed to run until it loses control either voluntarily or involuntarily—either way involves an interrupt.

## Interrupts

As mention, the z/OS architecture supports six types of interrupts:

- Machine check
- Restart
- Program check
- I/O
- Supervisor call (SVC)
- External

Each of these interrupts indicates an event that requires action on the part of the z/OS system software.

| | |
|---|---|
| **Machine check** | Indicates a problem with the server hardware |
| **Restart** | z/OS use only, rarely seen |
| **Program check** | An error or other event has occurred in a program |
| **I/O** | An I/O operation has ended |
| **SVC** | A supervisor call has been issued (the detail of this will be discussed later) |
| **External** | Caused by an event such as a time period expiring |

To pass control to the z/OS software, the hardware uses architecture-defined storage locations to do the following:

- Store the current PSW, which allows z/OS to resume the program at the point of the interrupt
- Store any other relevant information, such as the device associated with an I/O interrupt
- Load a new PSW, which gives control to the function in z/OS that will deal with the interrupt

## Supervisor call (SVC)

It is part of the architecture that some of the IBM System z instructions can only be executed if a particular authority level is in place, which is indicated in the PSW. This means that there are functions that a normal program is unable to perform (for example, starting I/O operations).

A supervisor call (SVC) is a form of application program interface (API). SVCs contribute to the overall integrity of the system, because only an SVC can perform functions that could possibly harm the operating system.

To request one of these restricted functions, the supervisor call is used. It has both a hardware and software component. In the hardware, the SVC is an instruction that any program is allowed to issue. It causes an SVC interrupt. The numerical parameter associated with it allows z/OS to identify the function required.

An example of this is SVC 0, which is a request to perform I/O. The program that handles SVC 0 requests verifies that the calling program is allowed to access the device, and then starts the I/O operation on the behalf of the calling program.

## Selection of work to run on a CP

Before explaining how CP resources are allocated, we need to explain how a unit of work in an address space is represented.

When the first program is loaded into an address space, a Task Control Block (TCB) is created to represent it. The TCB indicates whether the program is waiting for work to do, or has work and requires a CP to execute instructions.

When a task has work to do, its TCB is put on a queue and a z/OS function known as the Dispatcher allocates it to a CP. A program can call other programs that are represented by the same TCB, or a new TCB can be created. With this mechanism, a single address space can have units of work running concurrently on multiple CPs.

## Dispatching priority

The standard management of the z/OS system is based on a priority system, which is determined by the initial importance of a unit of work. This means that a TCB is placed on a queue based on its priority, which works well for the 24x7 transaction server because it starts at a high priority. However, lower priority units of work can struggle to get CP resources at busy times.

If an SLA sets the target for a transaction response time at two seconds and one second is being achieved, then the transaction server could run at a lower priority. If the response times go over the two-second target, then the transaction server would need an increase in priority.

To dynamically alter priorities, the Workload Manager (WLM) is used (see 3.3.6, "Workload Manager (WLM)" on page 56, for more detailed information about this topic). WLM has information relating to the SLA, and it keeps track of how successful the unit of work is in meeting its objectives. Based on this information, the WLM can alter the dispatching priorities and allow lower importance work access to the CPs.

## How CP utilization is balanced

In order to balance processor utilization between high importance and low importance work, the system needs to be able to interrupt a unit of work.

As mentioned, a program is allowed to run until it loses control either voluntarily or involuntarily. The voluntary method is to inform z/OS by issuing an SVC code 1; z/OS will then look for other work that is ready to run on the CP.

Involuntary loss of control occurs if an I/O interrupt occurs. This allows z/OS to evaluate the performance of the units of work in the system. However, an I/O interrupt will affect the internal caching of the CP, which can reduce its effectiveness. z/OS can set indicators that determine whether a CP can be interrupted by I/O interrupts, and therefore not all CPs will be interrupted in this manner.

This creates the potential for a low-priority unit of work to have the use of a CP even though more important tasks need to run in order to meet their objectives. The z/OS solution to this is to make use of the external interrupt facility. There are several reasons for this interrupt occurring, one of which is associated with a timed event. Before a task is dispatched, a timed event is set up, and if the task has not lost control before the time period has elapsed, it will do so when the associated external interrupt occurs.

By using this mechanism, WLM is able to allow low importance work to run when the high importance work is achieving its objectives.

## 2.9 Several servers on one physical machine

As previously mentioned, the optimum use of processor resources in a commercial environment is obtained by having a mixed workload. This was not possible on the early generations of mainframes, because they did not have sufficient processor power or memory to run multiple or very large workloads at the same time. Dedicated machines were the norm.

However, as processor speed and storage density increased, combined with a decrease in cost, this higher capacity allowed for the merging of workloads. Additionally, with fewer physical machines the infrastructure costs are reduced and connectivity is simplified. However, although it is *possible* to run enterprise's workloads on a single operating system, it is not necessarily the preferred method.

### 2.9.1 The LPAR

Most online environments need to have sufficient capacity to handle their peak workload volumes and are therefore sized such that they are not, generally, fully utilizing the resources that they are allocated. For reasons such as systems management and security, it is often preferable to have separate environments for system testing, application development, and production systems. In order to meet the requirement for function separation and to optimize processor utilization, the logical partition (LPAR) was introduced.

An LPAR is a subdivision of a machine's resources, which shares use of the central processors (CPs) with other LPARs but has main storage dedicated for its use. From the point of view of the programs running in an LPAR, the other LPARs do not exist (unless you connect them using a Coupling Facility, channel, hiperpipe, OSA or shared DASD). There is no common storage for them to use for communications. A hardware interface allows the definition of the amount of main storage to be allocated to an LPAR and the number of processors that it can use concurrently. We refer to a *processor* as the resources assigned and operating system running in the LPAR.

Each LPAR can access one or all of the processors installed on the machine and is given a relative weight. The weight is used by a microcoded supervisor function, the Processor Resource/Systems Manager™ (PR/SM), to determine which LPAR has use of a CP and for how long. The weight given to an LPAR and the number of CPs it is allowed to use usually has a relationship to the SLAs associated with the workload running on it.

The channel architecture, mentioned in 2.4.2, “Few servers - the IBM System z approach” on page 22, allows for LPARs to share the use of a channel. Therefore, the LPARs sharing a workload do not need their own cables to access the I/O devices, and the configuration is simplified.

## 2.9.2 Planning for downtime

All large systems must have the capacity for planned or unplanned outages, as rare as they may be in a mainframe environment.

### Software

All software systems require maintenance activity at some time, either to implement new levels or to fix problems in an existing level. There is also the possibility of failure.

Continuity of service can be provided by sharing the critical workload across two or more LPARs, on one or more physical machines. A careful choice of the time to take the system down (a scheduled downtime) should make it possible to meet the SLAs from the remaining systems. In the event of an unscheduled outage the service can still be provided, although possibly degraded, until the failing system is recovered.

### Hardware

Two or more physical machines are required to provide continuity of service in the event of a machine loss due to failure, maintenance, or repair, that requires the whole machine. In such a configuration, the LPARs sharing the critical workload have to be placed on different physical machines.

## 2.10 Parallel Sysplex

While the combination of IBM System z hardware and software provides excellent availability and reliability characteristics, both planned and unplanned outages can occur. If 24x7 availability is required, then more than one LPAR must be configured to share the critical workload; this is known as a Parallel Sysplex,

Parallel Sysplex has aspects that relate to capacity, availability, and scalability, and it is described in more detail under those topics. Briefly, it is a form of clustering that allows up to 32 systems to be linked together and share data. The components required to do this, in addition to the servers and I/O devices, are a time source and a Coupling Facility.

### Coupling Facility

The Coupling Facility (CF) is either a standalone server or an LPAR that runs a specialized operating system and provides facilities to control data sharing. A CF has high speed links to the LPARs in the Parallel Sysplex, and can store selected data in its main storage. This ability to make shared data available quickly is the major contribution of the Coupling Facility to capacity.

### Common time source

Transaction processing software writes log entries about the actions taken to update data, including timestamps. This is why a common time signal is needed in a clustered eComputing environment. In a Parallel Sysplex, a single high resolution time source is used to ensure that every subsystem of the Parallel Sysplex uses exactly the same time.

## 2.11 Measurements

As stated in 2.3, "Elements of a system required for capacity" on page 21, capacity requires that the CPs, main storage, and data access must be in balance, and it is essential that regular measurements are taken to detect imbalance before performance is impacted.

Both the IBM System z hardware and software produce information that may be recorded and processed by batch jobs to produce reports. The reports need to be read in the context of the SLAs, because the SLAs determine whether or not the capacity is sufficient. If elements of work are missing the SLA targets, then the causative factors must be addressed.

### Central Processor (CP) usage

Because the workload can be a mix of high and low importance, a CP utilization of 100% does not necessarily mean that capacity has been exceeded. In fact, it is not unusual for the CPs on IBM System z servers to be utilized at 100% for long periods of time. However, if the reports indicate that the reason for the failure to meet the SLA was due to waiting for CP resources, then a closer look is warranted. Delays due to waiting for CP resources do not necessarily mean that more CPs are required. Perhaps running part of the workload at a different time might be all that is needed.

### Main storage usage

Delays can be caused if the ratio of virtual storage to real storage is such that workloads are competing for the real storage. As with the CPs, it is not necessarily required to add more resources—there might be too many address spaces running concurrently, and an alteration to the scheduling will be all that is required.

### Access to disk storage

Delays due to elongated response times from disk storage can be the most difficult to identify. This is due to the interrelationship between the components involved, as described in Chapter 6, “Accessing large amounts of data” on page 109. Possible solutions are to add more channels, move data to different volumes, or change the workload mix.

## 2.12 Summary

Capacity is not a simple concept when running a mixed workload in a large-scale commercial computing environment. Different workloads have significantly different requirements. A well-defined set of service level agreements (SLAs) is the bedrock that allows maximum system throughput. SLAs are also helpful for the measurement of capacity bottlenecks and for the planning of capacity upgrades.

| **Key terms in this chapter** | | | | |
|---|---|---|---|---|
| 24-bit | CP | LPAR | PSW | SVC |
| 64-bit | Interrupt | Mixed workload | Real storage | Virtual storage |
| Address space | I/O | Parallel Sysplex | SLA | SVC |
| Bar | Line | | | |

## 2.13 Questions for review

1. Explain the advantages of a mixed workload.
2. Explain the difference between a single server approach (mainframe philosophy) and a multiple server approach. List the advantages and disadvantages of each.
3. List at least three elements of a system required for capacity planning and estimation.
4. Define the term “virtual storage”.
5. How much virtual storage does an address space offer under the latest z/OS version?
6. List the six interrupts of a z/OS system.
7. Explain what an SVC is and how it is used.
8. Explain the dependency between an SLA and capacity.

## 2.14 Topics for further discussion

1. Define some SLAs and discuss the implication of these SLAs to business and capacity planning or systems management.
2. Contrast the LPAR concept to VMware and other virtualization techniques.
3. Explain the differences between parallel sysplex and clustering (for example, Linux clusters).
4. Discuss the dependencies between SLAs and WLM definitions.

3

# Scalability

**Objectives:**

After completing this chapter, you will be able to describe the following:

- What scalability means
- Differences between scaling in and scaling out
- The relationship between mainframe hardware and scalability
- Software scalability levels
- The relationship between Parallel Sysplex and scalability
- Main workload management concepts

## 3.1 Introduction to scalability

Scalability is one of the main characteristics of a large-scale commercial operating system configuration. It is related to the ability of the operating system and its associated infrastructure, such as hardware and software, to expand as the business dictates.

We begin with some definitions:

| | |
|---|---|
| **Scale** | To change the size of an object while maintaining its shape.[1] |
| **Scalability** | The hardware capability of a system to increase performance under an increased load when resources are added.[2] |
| **Scalability** | The software ability to grow with your needs. A “scalable software package” means that you only buy the parts you need, and that the software package has the ability to grow by adding on as you grow.[3] |

Scalability is a characteristic that must be taken into account when designing and implementing a large -scale commercial operating system environment.

The IBM System z server family offers the described scalability. These machines are capable of enormous growth within each series, by making use of advanced clustering technology.

The advanced clustering technology within the IBM System z servers is more commonly referred to as a systems complex (sysplex). A *sysplex* is basically a collection of z/OS systems that cooperate, using certain hardware and software products to process work. This type of clustering technology can provide near-continuous availability and scalability.

The next step up from a basic sysplex is a Parallel Sysplex. This is a sysplex that uses multisystem data-sharing technology. A Parallel Sysplex relies on one or more Coupling Facilities.

A basic description of a Coupling Facility is a mainframe processor, with memory and special channels, and a built-in operating system. The Coupling Facility (CF) has no I/O devices, other than the special channels, and the operating system is very small. The operating system within the CF is nothing like z/OS and has no direct user interfaces.

[1] From Wikipedia Encyclopedia http://en.wikipedia.org/wiki/Scale
[2] From Wikipedia Encyclopedia http://en.wikipedia.org/wiki/Scalability
[3] From The Concise Tech Encyclopedia:http://www.tech-encyclopedia.com/term/scalability

## 3.2 Scalability concepts

By *scalability* we mean the ability of the hardware, software, or a distributed system to deliver the same functionality and to obey the same service level agreements (SLAs) as it is changed in size or volume. For example, the system needs the ability to retain or improve performance levels when adding processors, memory, and storage. A scalable system can efficiently adapt to its workload, with larger or smaller networks performing tasks of varying complexity.

### 3.2.1 Scalability approaches

Scalability is a highly significant issue in electronic systems, databases, routers, networking, and so on, and it implies performance. A system whose performance improves after adding hardware, proportionally (or at least predictably) to the capacity added, is said to be a "scalable system".

Scalability can be measured in three different dimensions:

- Load scalability - A system able to expand and contract its resource pool to accommodate heavier or lighter loads.
- Geographic scalability - A geographically scalable system is one that maintains its usefulness and usability, regardless of how far apart its users or resources are.
- Administrative scalability - No matter how many different organizations need to share a single distributed system, it should still be easy to use and manage.

For example, a scalable online transaction processing system or database management system is one that can be upgraded to process more transactions by adding new processors, devices, and storage, and which can be upgraded easily and transparently without shutting it down. It maintains its performance; that is, it is able to deliver transactions at a rate that grows nearly linearly with the capacity that has been added.

A routing protocol is considered scalable with respect to network size, if the size of the necessary routing table on each node grows as a function of log N, where N is the number of nodes in the network.

Scalability can be achieved in different ways, as follows:

- To *scale vertically* or *scale up* means to add resources to a single node in a system, such as adding memory or a faster hard drive to a computer.
- To *scale horizontally* or *scale out* means to add more nodes to a system, such as adding a new computer to a clustered software application.

A more detailed discussion on what scalability means and why it is important can be found in the paper entitled “Scaling—Up and Out”, which is available at the following site:

http://www.redbooks.ibm.com/abstracts/redp0436.html

The workload management function is a software piece that plays an important role in the management of these different workload types, and thus on the scalability of a system.

## 3.2.2 Scalability influences

Scalability refers to the ability of a system to provide nondisruptive growth path in order to handle the projected explosion in computing demand. Scalability includes not only processor power, but also processor memory, I/O bandwidth, the ability to attach to large number of peripherals, the ability of the software to exploit large systems and the ability to manage a large system.

Almost all platforms today are positioned as scalable systems. The IBM System z server family has the proven capability to support tens of thousands of users, while processing hundreds or even thousands of transactions per second.

There are two broad approaches to processor scalability—vertical and horizontal growth. These are explained in more detail here.

### Vertical growth

Vertical growth is the ability to upgrade the installed server processor capacity to a larger one within the same family, for example, by adding further processor engines to a Symmetric Multiprocessor (SMP).

The IBM System z multiprocessing design is a mature technology with decades of expertise behind it. In order to reduce interprocessor interference, IBM has improved both the hardware design and the software algorithms of the operating systems that support the IBM System z. This means that there is a greater scalability in model upgrades for IBM System z than for other platforms. This design also means that the IBM System z family of processors offers clear, simple upgrade paths between models.

### Horizontal growth through Parallel Sysplex

Horizontal growth is the ability to add processor capacity by adding more servers in a cluster. The IBM System z unique Parallel Sysplex share-all, clustering technology offers near-linear scalability and a capacity far exceeding any single commercial workload, by allowing you to connect up to 32 IBM System z servers and to operate and manage them as a single image.

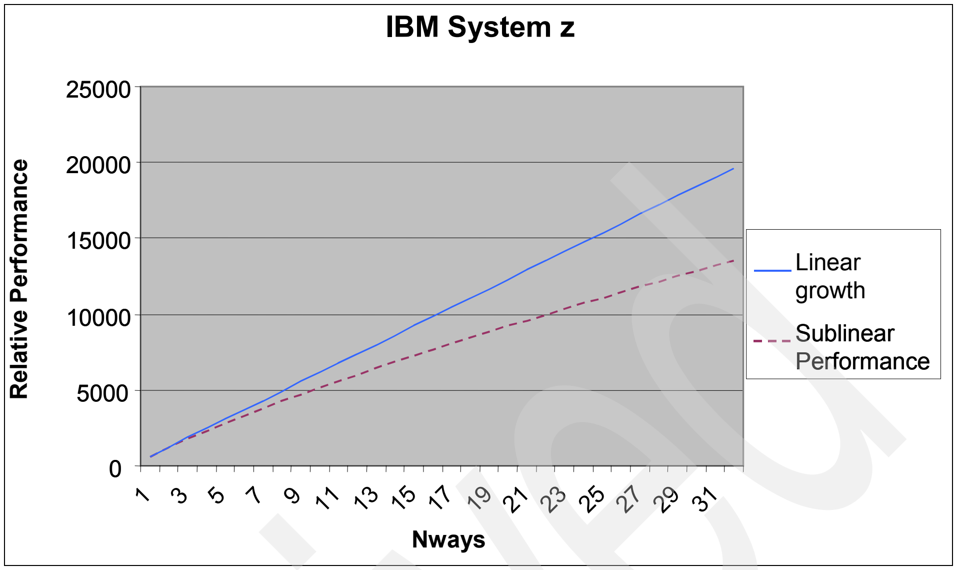


*Figure 3-1 Linear growth compared to Sublinear Performance*

Figure 3-1 shows linear growth compared to sublinear performance. The performance number is based on real-world test results. Nways® represents the number of processors. By "sublinear" we mean that the real (measured) performance growth is less than one would expect from drawing a straight line that is defined by the first two measurement points (in mathematics, sublinear means "less than linear"[4]).

In Figure 3-1 that line is also shown, so the nth CP of a Parallel Sysplex adds less computing power to the system than the one before (n-1). In a linear growth (top line), all CP would add the same amount of computing power.

Clusters, or scale out systems, replicate complete systems interconnected via a variety of interconnect mechanisms. In scale out systems, the performance and scalability are limited by both the speed and efficiency of the intra-node communication and by the effectiveness of the sharing scheme implemented at the software level.

There are many clustering schemes, depending on the machine type used. The IBM System z cluster technology is called Parallel Sysplex, and is described in detail Chapter 5, "Availability" on page 87.

[4] From Wikipedia Encyclopedia http://en.wikipedia.org/wiki/Sublinear

### 3.2.3 Provisioning

Traditionally, to provision meant to stock needed materials or supplies; especially a stock of food.

In information technology, the provisioning concept can be aligned to the scalability concept. It refers to how adaptable a system can be, to add more resources when needed, without disruption.

*Provisioning* is the end-to-end capability to automatically deploy and dynamically optimize resources in response to business objectives in heterogeneous environments. Provisioning helps to respond to changing business conditions by enabling the ability to dynamically allocate resources to the processes that most need them, as driven by business policies. Provisioning of individual elements, such as identities, storage, servers, applications, operating systems, and middleware, is a critical step to being able to then orchestrate the entire environment to respond to business needs on demand.

## 3.3 Scalability implementation on IBM System z

Here we introduce various concepts of how IBM System z is implemented to support scalability.

### 3.3.1 Hardware scalability

z/OS systems base their scalability on the way central processors (CPs) and memory are connected, and on the size and use of the cache memory.

The performance of the IBM System z servers is measured in terms of "throughput", the amount of work the server does in a given period of time. To isolate the processor unit performance from that of other subsystems necessary to make a complete server system, the Internal Throughput Rate (ITR) is measured.

The ITR is the amount of work done during the period in which the processor is actually performing work (rather than waiting for external devices to respond). The ITR is a measure of the number of completed transactions divided by the amount of time the central processor is busy.

As an example, Figure 3-2 shows a comparison between different models of IBM System z processors. You can see the balanced growth in all the axes.

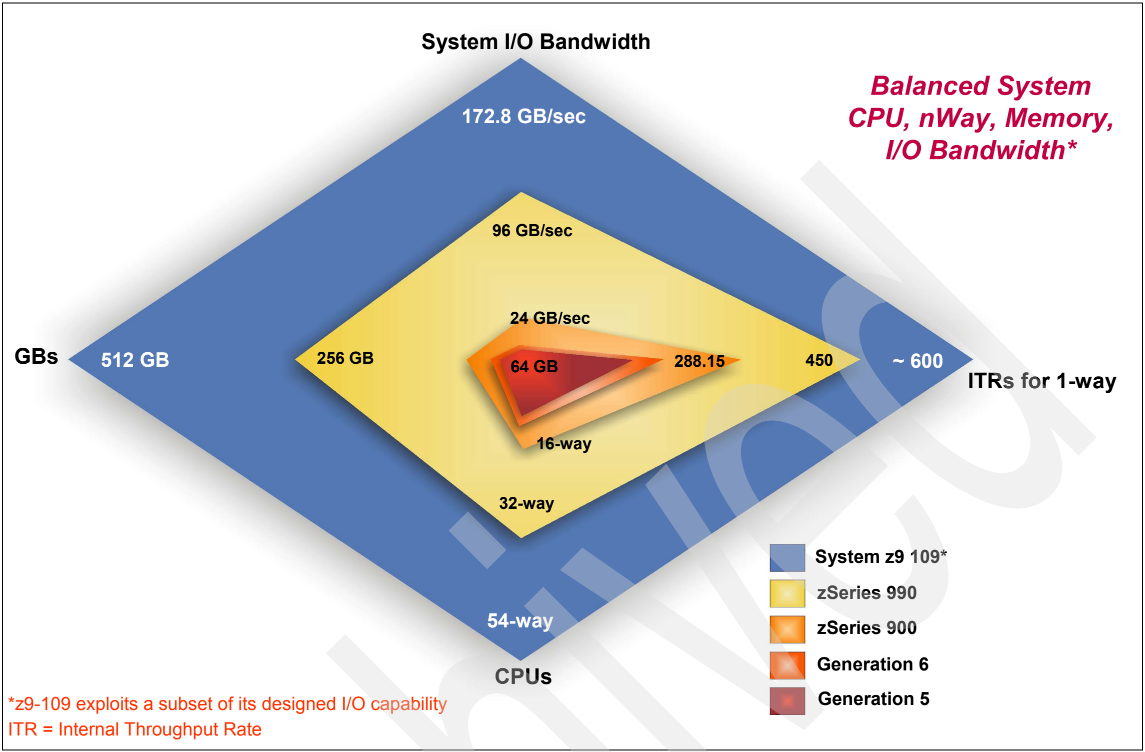


*Figure 3-2 Balanced growth*

Internal Throughput rate (ITR) is computed as:

$$\boxed{ITR = \frac{\text{Unit of Work}}{\text{Processor busy time}}}$$

A Unit of Work (UoW) is normally expressed as a job (or job-step) for batch workloads, and as a transaction or command for online workloads. To be useful, the UoW measured must represent a large and repeatable amount of the total workload, in order to best represent the average. Processor busy time is normally expressed in seconds.

For the purpose of computing an ITR, processor busy time should include all processing time to do work, including the operating system related overhead. On an N-way processor, processor busy time must represent the entire complex of engines as though there were a single resource. Therefore, processor busy time is the sum of the busy times of each individual engine, divided by the total number of engines. Since all processor time is included, “captured” and “un-captured” time considerations are unnecessary.

ITR characterizes processor capacity, since it is a CPU-busy time measurement. As such, ITR lends itself to the “processor comparison methodology”.

To ensure that the processor is the primary point of focus, you must configure it with all necessary external resources (including main storage, expanded storage, channels, control units, I/O devices) in adequate quantities so that they do not become constraints.

One additional point needs to be made regarding processor utilization. When two processors are to be compared for capacity purposes, they should both be viewed at the same loading point—in other words, at equal utilization. It is imprecise to asses relative capacity when one processor is running at lower utilization and the other is running at high utilization.[5]

## IBM System z processors

The IBM System z processors support a highly scalable standard of performance and integration by expanding on the balanced system approach of IBM z/Architecture. By a “balanced system”, we mean similar capacity growth in the three resource areas:

- CP (number of CPs and Cycle Time)
- Memory (size)
- I/O bandwidth (Gbps)[6]

Without such balance, performance can be degraded because of bottlenecks in one or more of these resources.

IBM System z machines are designed to eliminate bottlenecks and, through their 64-bit addressing capability (16 exabyte addresses; this is 16 * $2^{64}$), provide plenty of “headroom” to anticipate (predicted and unpredictable) growth in enterprise applications. zSeries introduces a packaging concept based on books. A *book* contains processor units (PUs), memory, and connectors to I/O cages that contain the I/O channels.

A IBM System z has at least one book, but may have up to four books installed. A full installed system, with four books and its components, is illustrated in Figure 3-3.

[5] Also see “Large Systems Performance Reference”, SC28-1187
[6] Bandwidth in IBM System z is usually measured in bytes per second, as opposed to the Telecommunications Bandwidth, which is usually measured as bits per second.

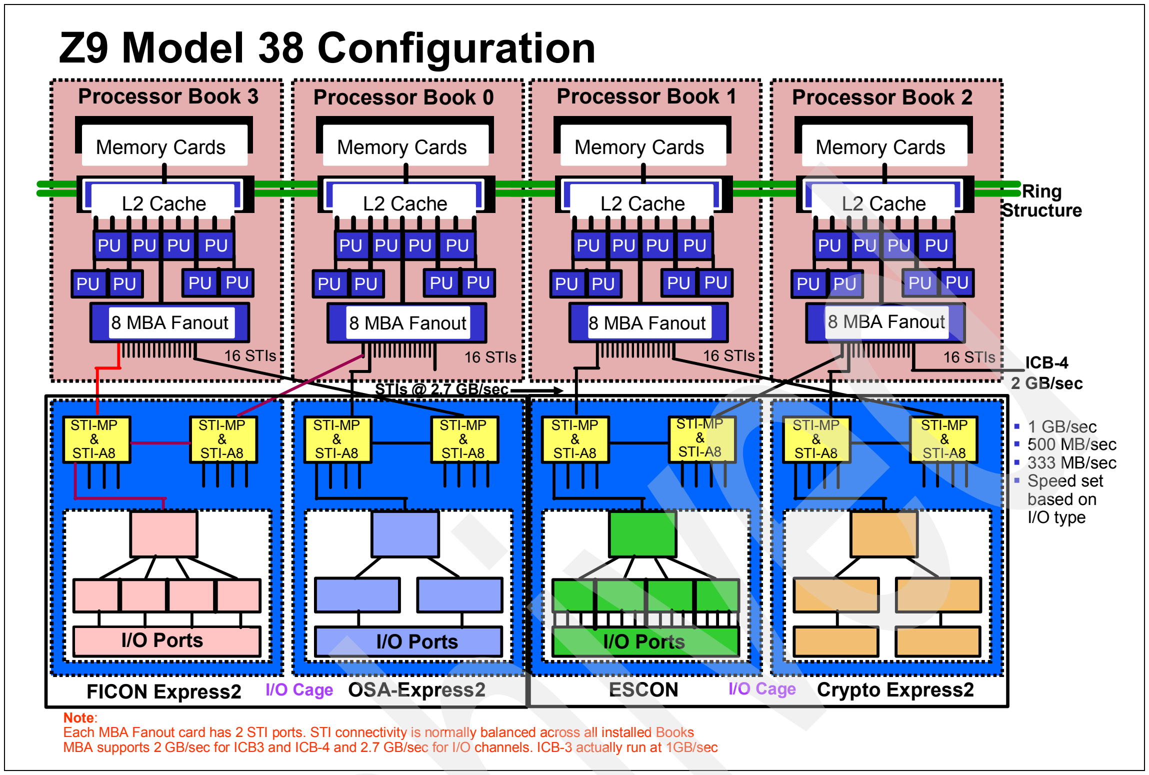


*Figure 3-3 Books and I/O structure in IBM System z*

Each book contains the following:

- 12 (16 on the higher models) processor units (PUs[7]), including up to 8 (12 on the higher models) Central Processors (CPs[8]), 2 System Assistant Processors (SAPs[9]) and 2 spares per IBM System z.

  The book contains a multi-chip module (MCM), which hosts the processor units, memory, and high speed connectors for I/O. A PU is the generic term for the z/Architecture processor on the Multichip Module (MCM) that can be characterized as:[10]

  – A Central Processor (CP) to be used by the operating system.
  – An Internal Coupling Facility (ICF) to be used by the Control Facility Control code (CFCC).
  – An Integrated Facility for Linux (IFL).

[7] A processing unit is the base chip. It is the engine that executes instructions.
[8] CP is the most used flavor of PU. It is used for non-special processing.
[9] SAPs are used to execute I/O.
[10] Refer to the IBM Redbook *IBM System z9 109 Technical Guide*, SG24-7124, for more details.

  - An additional System Assist Processor (SAP®) to be used by the channel subsystem.
  - A System z9™ Application Assist Processor (zAAP).
  - A System z9 Integrated Information Processor (zIIP), which will allow certain functions of DB2 database processing to be off-loaded.
- 16 GB to 128 GB of physical memory. The memory grows in 16 GB increments.
- Four memory bus adapters, supporting up to 16 self-timed interconnects to the I/O cages, with a maximum bandwidth of 43.2 GB per second.

IBM System z has the memory bandwidth required for running large diverse workloads typical with large-scale commercial needs, or with server consolidation.

Each PU has 512 KB of L1 cache and each book has 40 MB of shared level 2 cache. As a result, no matter which PU work is dispatched upon, recent context will most likely be in the shared level 2 cache, thus eliminating the need for an expensive memory fetch.

Other architectures typically do not have this size of shared cache, opting instead for larger level 1 caches on individual PUs. This approach fits well for *mono-application* servers, but in a *multiapplication* environment with high levels of context switching, work is constantly being moved between physical processors, and thus cache contexts are constantly invalidated, requiring time-consuming memory fetches.

When additional books are installed, the shared level 2 caches in the additional books are linked together in two concentric high-speed rings, to allow level 2 caches to be interconnected. Thus, a 32-way IBM System z is truly an SMP environment, where level 2 cache (4x40 MB = 160 MB of cache) is available to all 32 PUs, allowing for efficient processing of large diverse workloads.

So, at the time of writing, a IBM System z processor can scale from 1 CP, 16 GB of physical memory and a bandwidth of 43.2 Gbps to a maximum configuration of 54 CPs, 512 GB of physical memory, and an I/O bandwidth of 172.8 Gbps.

### 3.3.2 Operating system scalability

z/OS operating systems have many characteristics related to scalability that allow them to scale up to 16, or even 32, CPs on a single operating system instance. Scaling efficiency is very close to linear, proving full power for each processor that is added.

## Three levels acquiring resources for an application to grow

The addressing scheme in z/OS allows a theoretical limit of 16 exabytes. As explained in Chapter 2, “Capacity” on page 19, each task or group of associated tasks is executed in an address space (AS). Even though there is a 16-exabyte limit, no single task is able to use the whole extent, and the use of multiple address spaces is more related to integrity issues than to capacity ones.

However, from a scalability viewpoint, this addressing scheme means that for a task running in a single address space, there is plenty of room to grow. Data sharing inside an address space can be done directly (every piece of code has the same addressing) or by means of specialized middleware (for example, DB2). Database Manager (DB2) is an example of a heavy storage user. DB2 handles its largest tables (up to 128 TB) in buffers residing in a single address space.

### *Multiple Virtual Storages*

Figure 3-4 shows the layout of a zOS system.

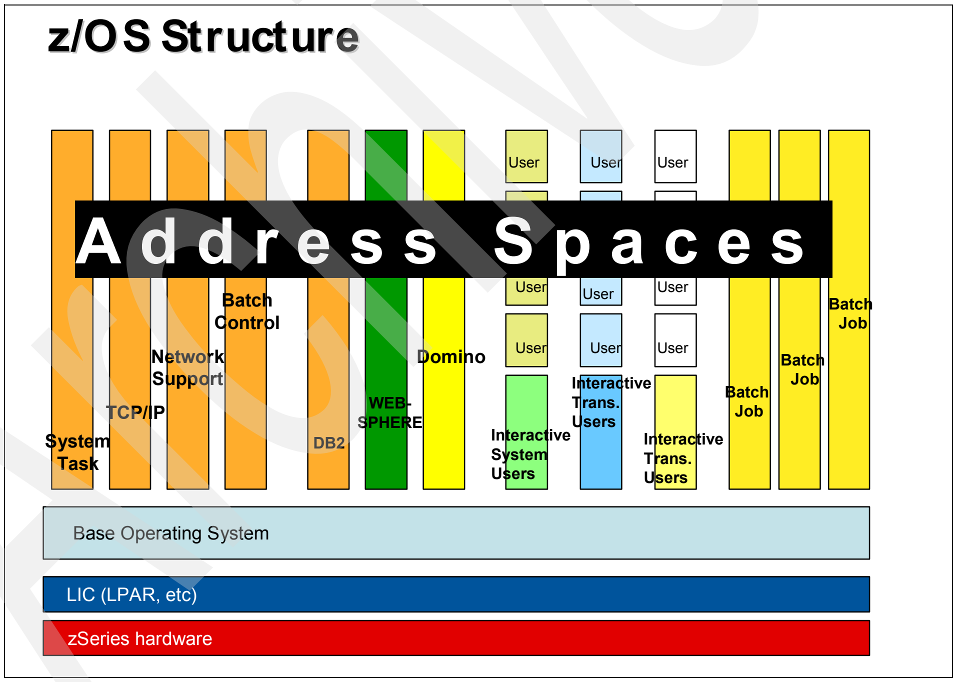


*Figure 3-4 Address spaces in a z/OS system*

The use of multiple address spaces provides a means for security and isolation, as well as a means of adapting the operating system size to the size of the machine it is running on. The number of address spaces that can be started is limited by the available storage. Data can be shared using buffers in the Common Area or using specialized middleware (such as DB2).

#### *Multiple Instances of OS running Single Image (Parallel Sysplex)*

Running more than one system (up to 32) sharing the same data can increase the capacity of the zSeries. Data is shared using either external storage or external memory (Coupling Facility). Usually data is shared by means of middleware that uses such facilities.

## 3.3.3 Parallel Sysplex

A conventional large computer system uses hardware and software products that cooperate to process work. A major difference between a sysplex and a conventional large computer system is the improved scalability and level of availability in a sysplex. Sysplex technology increases the number of CPs that can realize common work. A single operating system can use 16 or 32 CPs, depending on the version. A Parallel Sysplex can have up to 32 members, that makes 16 times 32 CP in one single system. Figure 3-5 shows a Parallel Sysplex.

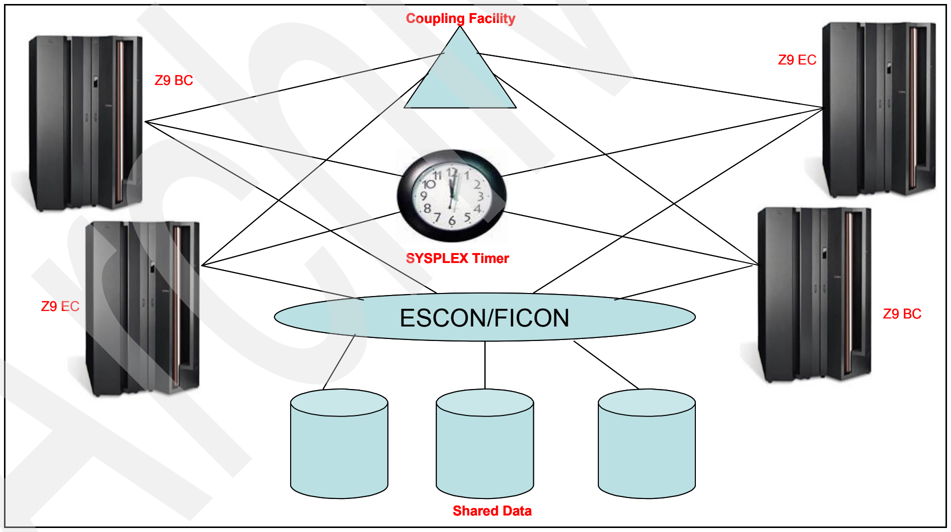


*Figure 3-5 Scalability of IBM System z - Parallel Sysplex*

To facilitate this cooperation, there are specialized hardware and software technologies that have been added to the standard software to enable sysplex capabilities. The hardware technology is covered in 2.10, “Parallel Sysplex” on page 34.

One of the key hardware components of this enabling technology is the Coupling Facility (CF) which makes high-performance multisystem data sharing possible. The CF provides high-speed locking, caching and message list services between the zSeries servers and the z/OS systems within a sysplex using coupling link technology.

## Parallel Sysplex technology

There are technologies related to Parallel Sysplex that allow it to be seen as a single system, resulting in better scalability. [11]

### *Serialization*

In a multitasking, multiprocessing environment, *resource serialization* is the technique used to coordinate access to resources that are used by more than one application. In a sysplex, in order for resources to be available to more than one system, serialization is maintained using an extension of the traditional mainframe serialization scheme. Programs that change data (or other serially reusable resource) need exclusive access to the data. Otherwise, if several programs were to update the same data at the same time, the data could be corrupted (also referred to as a loss of data integrity). On the other hand, programs that need only to read data can safely share access to the same data at the same time.

The most common techniques for serializing the use of resources are enqueuing and locking. These techniques allow for orderly access to system resources needed by more than one user in a multiprogramming or multiprocessing environment.

In the operating system, a serialization component processes the resource requests that programs issue. This component in z/OS is called Global Resource Serialization (GRS). It serializes access to resources to protect their integrity. An installation can connect two or more z/OS systems to form a complex in which the use of the resources shared among the systems is serialized.

When a program requests access to a reusable resource, the access can be requested as *exclusive* or *shared*. When the system grants shared access to a resource, exclusive users cannot obtain access to the resource. Likewise, when the system grants exclusive access to a resource, all other requestors for the resource wait until the exclusive requestor frees the resource.

It is important to note that this scheme is based in two conventions:

- Every resource user uses the same name to name a resource.
- Every resource user asks permission before using a resource.

[11] Refer to *Cluster architectures and S/390 Parallel Sysplex scalability* by G. M. King, D. M. Dias, and P. S. Yu, IBM System Journal Volume 36, Number 2, 1997.

There are several separate but complementary serialization approaches in z/OS. Locking is extremely fast (very short path length), but should be used for only a small number of locks. Enqueuing (ENQ) is slightly slower, but can be used for a great number of resources. These two serialization approaches are explained in more detail here:

- Locking

  Locking serializes the use of system resources by authorized routines and, in a Parallel Sysplex, by processors. A lock is simply a named field in storage that indicates whether a resource is being used and who is using it. In z/OS, there are two kinds of locks: *global locks*, for resources related to more than one address space, and *local lock*s, for resources assigned to a particular address space. Global locks are provided for nonreusable or nonsharable routines and various resources.

  To use a resource protected by a lock, a routine must first request the lock for that resource. If the lock is unavailable (that is, if it is already held by another program or processor), the action taken by the program or processor that requested the lock depends on whether the lock is a *spin lock* or a *suspend lock*:

  – If a spin lock is unavailable, the requesting processor continues testing the lock until the other processor releases it. As soon as the lock is released, the requesting processor can obtain the lock and, thus, get control of the protected resource. Most global locks are spin locks.

  – If a suspend lock is unavailable, the unit of work requesting the lock is delayed until the lock is available. Other work is dispatched on the requesting processor. All local locks are suspend locks.

  To avoid deadlocks, locks are arranged in a hierarchy, and a processor or routine can unconditionally request only locks higher in the hierarchy than locks it currently holds.

  For example, a deadlock could occur if Processor 1 held Lock A and required Lock B; and Processor 2 held Lock B and required Lock A. This situation cannot occur because locks must be acquired in hierarchical sequence.

  Assume, in this example, that Lock A precedes Lock B in the hierarchy. Processor 2, then, cannot unconditionally request Lock A while holding Lock B. It must, instead, release Lock B, request Lock A, and then request Lock B.

  Because of this hierarchy, a deadlock cannot occur on the acquisition of locks as it can occur in the enqueue scheme. This scheme is less scalable than the enqueue one and that is the reason why it is much less used.

  Theoretically its use is open, there is a published interface, but in the real word it is used by systems and subsystems (DB2, transaction managers, and so on). Those represent a controlled number of users.

- Enqueuing (Enq)

  Enqueuing is accomplished by an external and published interface. If the resources are to be modified, the user must request exclusive control; if the resources are not to be modified, the user should request shared control, which allows the resource to be shared by others who do not require exclusive control. If the resource is not available, the requestor is suspended until it becomes available. This can lead to deadlocks that must be detected and in some circumstances solved by the system operator or an automated operator.

  This serialization mechanism is able to scale quite well. In this context, to be scalable means being able to cope with a growth in the number of users without becoming a bottleneck. The enqueuing scheme is scalable for two reasons:

  – Granularity

    The scope of the resource can be as little as needed to prevent too many users from competing for the same entity. This can be better understood with an example. User A wants to update a field Z in record Y on file X on volume V. The enqueue can be done from the more general (volume) to the more particular (field). The users of the ENQ mechanism (DB2, Dataset Management, Security, and so on) use the granularity they need to be scalable.

  – Star configuration

    The queue is in a common area—in the OS Common System Area if there is only one system—or in the external Common Area (CF structure) in a sysplex. The communication is always with a central point so no other member can interfere. The only concern is how many members are queued for the same resource, but this is controlled by means of the granularity.

### *Communication*

The cross-system Coupling Facility (XCF) component of the z/OS operating system provides simplified multisystem management services within a base sysplex configuration. In a base sysplex environment, that is, a system configuration that does not include a Coupling Facility, the various z/OS systems on the same zSeries server communicate using hardware technology known as Channel-to-channel (CTC) connections.

The Cross-System Extended Services® (XES) component of the z/OS operating system is used to access the Coupling Facility environment.

These services allow authorized programs on one system to communicate with programs on the same system or on other systems. If a system fails, system services also provide the capability for batch jobs and other tasks to be restarted on another eligible system in the sysplex.

XCF uses an external file that contains “heartbeat” information: related data about the sysplex, systems, groups, members, and general status information. This file is required to run a Parallel Sysplex with more than one system, and must be duplexed. Another file is required that contains relevant information about the CF including its name, and what structure(s), including their size, are defined in the CF.

These authorized programs can be any program using a published interface with the communication mechanism. There are many operating system components that use it, for example GRS, the serialization software. It uses the provided communication interface between systems to coordinate the requests.

Other users are console support to provide a single point of control, and most of the components that run instances in more than one system of the sysplex (database subsystem, messaging subsystem, security subsystem, and so on).

Communication capability (bandwidth and speed) among members is often one of the more important aids or inhibitors to scalability. As with serialization, the use of common external storage, accessed with links that have a capacity of up to 2 Gbps each, provides better performance.

## Data sharing and Coupling Facility structures

Data is shared in a sysplex using an external storage called the Coupling Facility (CF). In the Coupling Facility, storage is dynamically partitioned into structures, as illustrated in Figure 3-6.

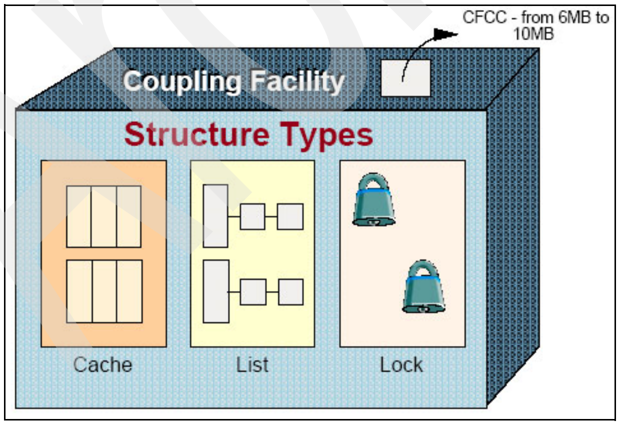


*Figure 3-6 Coupling Facility structures*

z/OS services manipulate data within the structures. There are three types of structure:

| | |
|---|---|
| **Cache structure** | Supplies a mechanism called *buffer invalidation* to ensure consistency of cached data. The cache structure can also be used as a high-speed buffer for storing shared data with common read/write access. |
| **List structure** | Enables authorized applications to share data that is organized in a set of lists, for implementing functions such as shared work queues and shared status information. |
| **Lock structure** | Supplies shared and exclusive locking capability for serialization of shared resources down to a very small unit of data. |

The Coupling Facility can be used by any authorized program. Most operating system components and subsystems, such as database manager or messaging manager, have a copy of their data in the Coupling Facility.

The sysplex scalability depends on how efficient the use of the CF is, and this efficiency basically consists of the access time to the data in the Coupling Facility.

When running a sysplex, two main factors affect performance:

| | |
|---|---|
| **Access time** | To get the data from internal memory takes less time than to get it from external memory. |
| **Access queuing** | When there is only one database manager accessing a database, there is no queuing for a piece of data, except for the internal competence within the DB application, because the database manager is the only user. One of the main reasons to build a sysplex is to split transactional loads involving database access between two or more sysplex members. Because each of them have their own instance of the database manager, access to the data buffer must now be serialized. |

## Workload distribution

By *workload distribution*, we mean the piece of software able to distribute work among the various application servers. This capability is a key factor affecting scalability of a system. The database servers, as well as the transaction servers, usually have more than one application server serving the same set of transactions.

The better the workload distribution, the less demand for increasing hardware resources to achieve workload growth. There are different techniques used to achieve workload distribution. We list them here, from the simplest to the more sophisticated.

**Manually** The network is divided into subnetworks and each of them is served by an application server. This approach is a fixed one and can end in an overloaded server next to an underloaded one.

**Round robin** There are also workload distributors that are connected to more than one server and can send a request in a more balanced way. Even in this approach the balance is not always even, because the units of work, or transactions, can be of many different sizes.

These workload distributors can be external machines that are connected to different IP addresses, each corresponding to a server, or they can be internal workload distributors. These communicate with the application servers and receive all requests, sending them to the servers with a “one each” algorithm.

**Dynamic workload distribution**

The request reaches the workload distributor, which asks workload management for the relative load of each of the application servers. Depending on the answer, it drives the request to the application server best suited to it.

**Workload Management-driven application servers**

Application Environments allow WLM to start new servant address spaces in order to meet transactions performance goals. WLM will start a new servant region address space or stop a servant region address space as the workload varies.

WLM may start additional server regions to meet performance goals, based on the prioritization of its work compared to other work in the system, the availability of system resources needed to satisfy those objectives, and a determination by WLM whether starting more address spaces will help achieve the objectives.

## 3.3.4 Provisioning

As previously stated, provisioning has became a concept related to scalability. It refers to how easily a system can receive more resources when needed, without disruption.

One of the components in z/OS responsible for the management of system resources is the Workload Manager (WLM) component. WLM manages the processing of workloads in the system according to the company's business goals, such as response time. WLM also manages the use of system resources, such as processors and storage, to accomplish these business goals.

Provisioning often appears in the context of virtualization orchestration, and other *dynamic datacenter* concepts, and there are tools in the market to help with the creation of a dynamic datacenter. But each component of an IT infrastructure has its own capabilities. The capabilities related to provisioning in IBM System z are:

- Dynamic resource distribution.
- A IBM System z server can be partitioned into as many as 60 logical partitions.
- Each logical partition is completely isolated and protected from other LPARs.
- The processors can be shared among LPARs as a percentage of the total capacity. Each LPAR is given its minimum number of CPUs, as defined by the system administrator—although under specific conditions, an LPAR can acquire additional CPU resources if processors are not being used by other executing LPARs. Therefore, CPs can move among the executing LPARs as long as all Service Level Objectives (SLO) are being met by all workloads.
- The I/O bandwidth (channel cards and paths to externally attached devices) can be shared among the LPARs under WLM control (WLM needs to be defined prior to using it here).

## 3.3.5 Capacity on Demand

Capacity on Demand (CoD) encompasses the various capabilities available for you to dynamically activate one or more resources on your server as your business peaks dictate. You can activate inactive processors or memory units that are already installed on your server on a temporary or permanent basis.

Also called Nondisruptive hardware addition, it is a special hardware contract used to provide CPU resource flexibility. Hardware is delivered in full configurations with a microcode limit on the size that has been purchased. But the hardware is there, so it can be made active with a microcode update. CPU and memory can be added this way.

This feature is usually used to cope with unexpected workload growth for any reason (maintenance, failure, growth in demand, seasonal peaks, and so on).

There are different CoD options:

- Capacity Upgrade on Demand (CUoD)

  CUoD allows for the non-disruptive addition of Central Processors (CPs) and Speciality Engines. CUoD can add capacity up to the maximum number of available inactive Processing Units (PUs). For certain configurations, 8 GB memory increments can be added by CUoD, and also dynamic upgrade of all I/O cards in the I/O cage is possible.

- Customer Initiated Upgrade (CIU)

  CIU is designed to allow you to respond to sudden increased capacity requirements by downloading and applying a processor unit (PU) or memory upgrade via the Internet (using the IBM Resource Link™ and the Remote Support Facility options). With the Express option for CIU, an upgrade may be installed within a few hours after order submission.

- On/Off Capacity on Demand

  The On/Off CoD server offering allows you to pay a fee to enable and use temporary server capacity, such as a nondisruptive temporary addition of CPs, IFL, ICFs, zAAPs, and zIIPs. It also includes nondisruptive removal when the added capacity is no longer wanted.

### 3.3.6 Workload Manager (WLM)

Can your enterprise server handle the following?

- Manage resources, based on your business objectives?
- Run diverse mixed workloads without experiencing a loss in critical business throughput or response time?
- Help protect your most critical work from “killer” applications and “monster” queries?
- Dynamically alter work and capacity as business requirements change?
- Respond instantly to shifts in business priorities?

If your answers are yes, then it is very likely that your enterprise server is IBM System z. IBM System z and z/OS are able to achieve this and much more through its industry-unique, policy-driven Workload Manager (WLM). In short, WLM ensures that critical application response times are met.

One of IBM System z's greatest strengths is its ability to function under high server utilization, while efficiently balancing the use of system resources and the need to deliver quality service to your most important workloads.

The ability to dynamically manage very diverse workloads running concurrently is a unique attribute of zOS. You specify your business goals and WLM takes care of the rest. WLM is unique in offering system administrators the capability to specify goals for work in the system in business terms, rather than using low-level parameters.

The operative principle is that the system is responsible for implementing resource allocation algorithms that allow these goals to be met. WLM offers externals that capture business importance and goals and implements them on behalf of the system administrator.

Also, through the use of “periods” (time intervals for a given service consumption), WLM allows business objectives to change, based on the variation of the resource demands of the work requests. This addresses the fundamental problem that the resource demands of most work requests are unknown at the outset and can vary, depending on factors that may be known only at execution time.

The installation defines performance goals and assigns a business importance to each goal. These goals are specified in business terms and the z/OS WLM decides how much resource, such as CPU and memory, should be given to the task or work request to meet its business goal.

Thus, WLM is unique in the fact that it provides what can be called “just-in-time computing”, meaning that it adjusts dynamically the amount of CPU and memory available to work requests.

In simple terms, WLM has three objectives:

1. To achieve the business goals that are defined by the installation, by automatically assigning sysplex resources to workloads based on their importance and goals; this objective is known as *goal achievement*.
2. To achieve optimal use of the system resources from the system point of view; this objective is known as *throughput*.
3. To achieve optimal use of the system resources from the point of view of the individual address space; this objective is known as *response* and *turnaround time*.

Goal achievement is the first and most important task of WLM. Optimizing throughput and minimizing turnaround times of address spaces come after that.

Often, these latter two objectives are contradictory. Optimizing throughput means keeping resources busy. Optimizing response and turnaround time, however, requires resources to be available when they are needed. Achieving the goal of an important address space might result in worsening the turnaround time of a less important address space. Thus, WLM must make decisions that represent trade-offs between conflicting objectives.

To balance throughput with response and turnaround time, WLM does the following:

- Monitors the use of resources by the various address spaces.
- Monitors the system-wide use of resources to determine whether they are fully utilized.
- Determines which address spaces to swap out, and when.
- Inhibits the creation of new address spaces or steals pages when certain shortages of main storage exist.
- Changes the dispatching priority of address spaces, which controls the rate at which the address spaces are allowed to consume system resources.
- Selects the devices to be allocated, if a choice of devices exists, in order to balance the use of I/O devices
- Other z/OS components, transaction managers, and database managers can communicate to WLM a change in status for a particular address space (or for the system as a whole), or to invoke WLM's decision-making power.

For example, WLM is notified when:

- main storage is configured into or out of the system.
- An address space is to be created.
- An address space is deleted.
- A swap-out starts or completes.
- Allocation routines can choose the devices to be allocated to a request.
- The processor capacity of a IBM System z server has changed via the Capacity on Demand capability.

WLM is particularly well-suited to a sysplex environment. It keeps track of system utilization and workload goal achievement across all systems in the Parallel Sysplex and data sharing environments. For example, WLM can decide the z/OS system on which a batch job should run, based on the availability of resources to process the job quickly.

## Workload Manager components

All the business performance requirements of an installation are stored in a *Service Definition*. There is one Service Definition for the entire sysplex.

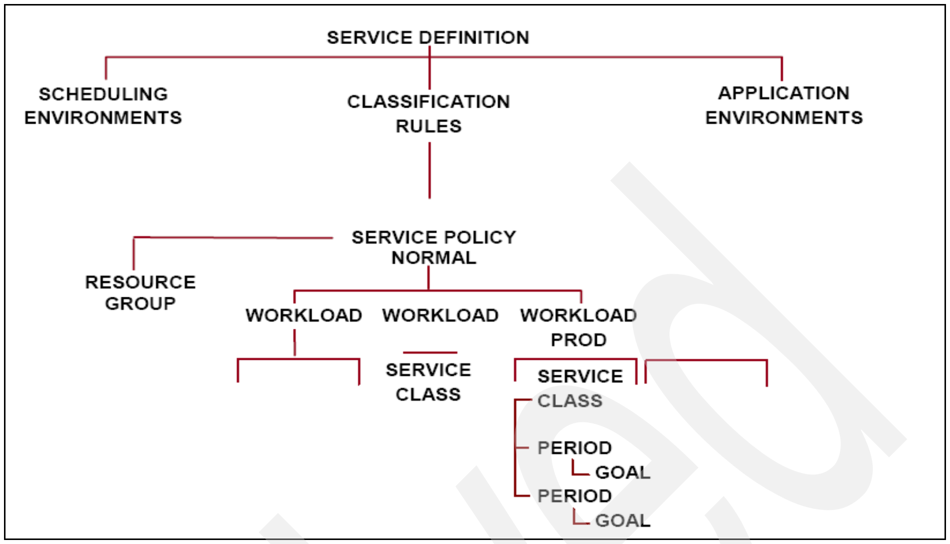


*Figure 3-7 WLM components*

The Service Definition contains the elements that WLM uses to manage the workloads, as shown in Figure 3-7 for this. These include:

- One or more service policies.
- Workloads: These are arbitrary names used to group various Service Classes together for reporting purposes. At least one Workload is required.
- Service Classes: There are where you define the goal for a specific type of work.
- Report classes: This is an aggregate set of work for reporting purposes.
- Performance goals: This is the desired level of service that WLM uses to determine the amount of resource to give to a unit of work.
- Classification rules and classification groups: These are used to assign the incoming work to a Service Class and, if needed, to a report class.
- Resource Groups: These are used to assign a minimum and a maximum amount of CPU SU/sec to one or more Service Classes.
- Application Environments: A set of application functions requested by a client and executed in a server address space. WLM can dynamically manage the number of server address spaces to meet the performance goals of the work making the requests.
- Scheduling Environments: A list of resource names and their required state. The Scheduling Environment allows you to manage the scheduling of work in

an asymmetric sysplex, where the systems differ in installed applications or installed hardware features.

## Work unit identification

The identification of work requests is supported by middleware and the operating system, which tells WLM when a new unit of work enters the system and when it leaves the system. WLM provides constructs to separate the work into distinct classes. This is called *work classification* and allows an installation to deal with different types of work.

One of the strengths of WLM is its ability to recognize and manage units of work being executed by important middleware applications running on z/OS.

Figure 3-8 depicts the general structure of transaction processing on z/OS. Depending on the middleware or subsystem, one or multiple address spaces provide a connection to external clients. These address spaces are usually called *regions*, from the early days when programs were allocated pieces (regions) of the available storage.

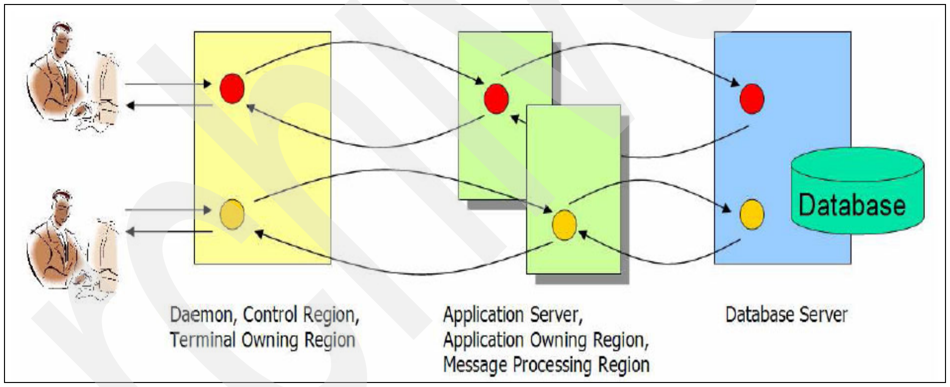

Database
Daemon, Control Region, Terminal Owning Region
Application Server, Application Owning Region, Message Processing Region
Database Server

*Figure 3-8 Transaction structure*

The region that receives the work request on z/OS checks its authority and resource requirements. If the application can process it, the region sends it to an application processing region.

Note that there may be more than one application processing region. For CICS®, these are named Application Owning Regions (AOR) and for IMS™, these are called Message Processing Regions. The application region starts a program which processes the request and the program invokes other middleware, operating system services or services from other subsystems to complete the request. A typical example is that the request accesses a database, in many cases DB2 on z/OS.

What we can see from this example is that a client work request spans across multiple address spaces on z/OS. In order to manage these requests, it is necessary to recognize the transaction flow and based on this, to manage the address spaces and requests to meet the end-user expectation.[12]

WLM developed a set of programming interfaces which allow middleware and subsystems to identify work requests to WLM, to help WLM to trace the work through the system and to identify the time the requests stay on z/OS.

Over time, WLM developed various techniques to recognize work on the system and to manage it independently from or together with the address space where the programs run to process them, as described here:

- The most basic variant is the address space. The start of an address space is known to WLM and if no more information is available about the work in the address space, WLM can manage it based on its resource consumption and demand. Address spaces are started and classified as started tasks and the only goals that are applicable for them are execution velocity goals or discretionary.
- Batch work re-uses already started address spaces. These are called *initiators* and they select work from the Job Entry Subsystems (JES). The beginning and end of each new batch job is signaled through a system interface to WLM. This allows WLM to recognize its duration, provide classification rules for them, and manage the initiators based on the currently active job.
- Time Sharing Option (TSO) uses a different technique. After a TSO user address space has been started, it waits on input from the TSO user terminal. When the user presses Enter, a new TSO transaction starts and its end is defined when the result is returned to the terminal.

  The begin and end of TSO transaction is signaled to WLM so that it is able to recognize the individual transactions and to manage the TSO address spaces based on them. UNIX System Services (OMVS) uses the same interface to inform WLM about the beginning and the end of end-user requests.
- CICS and IMS regions register as work managers to WLM and create control information (performance blocks) for all work which can be executed in parallel in the region.

  When a work request enters the TOR or IMS Control Region, it is classified and associated with a performance block. State conditions are recorded in the performance block during execution and when the request is passed from a TOR to an AOR, the continuation is maintained and information is provided to WLM to understand the relationship.

[12] For more information about this topic, refer to:
http://www.ibm.com/servers/eserver/zseries/zos/wlm/pdf/zWLM.pdf

## Defining the service level

In 2.6, “Service level agreement” on page 24, there is the definition of a Service Level Agreement (SLA). SLAs must be adhered to, and in z/OS the way to do so is by defining a service level for each workload. This is done in the service class definition.

A service level (a goal objective in the WLM scheme) can be measured in two different ways, depending basically on how long a request for work stays in the system and on how repeatable the request is. As an example, you cannot use the same measurement unit with a request for work that is to execute a transaction named ABC to consult the credit in a bank’s account with a request for work that represents a task that runs without end (for example, a database manager) or a batch job.

### *Importance of a goal*

While the goal type depends on the kind of work, its importance depends on how work is treated by the system. It indicates which among the “unhappy” service class should receive resources in order to achieve the target goal. The values can be 1 to 5, with 1 being the most important and 5 the least important.

### *Adjustment routine*

Every 10 seconds, the WLM policy adjustment routine gains control in one z/OS. It summarizes the system state information sampled (every 250 milliseconds) and combines it with measured data (response time, CPU usage, paging rate, and so on). At this point WLM learns the behavior of the workloads and the system.

Everything WLM does is based on measured and sampled data, which is combined with WLM’s learning of the relationship between workloads and resources in the system.

### *Workload Management controls*

WLM performs the following main management control activities:

| | |
|---|---|
| **CPU management** | Access to the CPU is controlled through dispatching priorities. The dispatcher maintains a queue of ready work units, which is sorted in priority order. This queue is managed by WLM, assigns dispatching priorities to service classes. |
| **I/O management** | WLM plays with I/O priority with the I/O requests generated by the units of work executing transaction code, as a procedure to honor the installation goals. It is also possible to dynamically |

vary the number of parallel accesses to, for example, a 3390 volume (PAV).

**Transaction management** This kind of work usually spans multiple address spaces on z/OS. In order to manage such requests it is necessary to recognize the transaction flow and, based on this, to manage the address spaces and requests to meet the end user expectation. An enclave is used to keep the priority and the account information for a unit of work independent of the address space‘s priority and account. As a consequence, this transaction can have priorities and account information only for itself, independent from others located in the address space.

We can summarize WLM with two statements:

- The work classification and workload management observation of how the work uses the resources provide the basis for managing the system.
- In the case where several service classes do not achieve their target goals, the importance will help in deciding which service class should be prioritized for getting resources.

### WLM extensions

WLM controls are developing in two directions: to better control workload distribution sysplex-wide and to be able to control server behavior in a multiplatform infrastructure.

### Intelligent Resource Director (IRD)

WLM has control over the resources on a single LPAR. There is an extension that can achieve better resource distribution among LPARs. It is called Intelligent Resource Director (IRD), and has three components:

- CPU LPAR management - WLM and LPAR scheduler can dynamically change the number of logical CPs and the weight of each logical partition.
- Dynamic channel path management, which allows dynamic connection of channels with I/O controllers when there is a surge in the demand. The connection is implemented via channels connected through switches only.
- I/O Priority, which allows the I/O requests to be ordered by the I/O priority decided by WLM.

## 3.4 Summary

In this chapter, we explained what scalability means. We also defined important terms and functions. We described how scalability is implemented on an IBM System z mainframe, and discussed hardware and software scalability levels. Finally, the relationship between Parallel Sysplex and scalability was covered, along with the concepts of Workload Manager were covered.

| **Key terms in this chapter** | | | | |
|---|---|---|---|---|
| Access time | Coupling Facility | ITR | Provision | Serialization |
| CF | CP | Locking | Scalability | SLA |
| CoD | Enqueuing | LPAR | Scale in | WLM |
| Communication | IRD | Parallel Sysplex | Scale out | Workload |

## 3.5 Questions for review

To help test your understanding of the material in this chapter, complete the following review questions:

1. Describe the two different directions of scalability.
2. Describe a “balanced system” approach in the z/Series architecture.
3. Describe a “book” in reference to a z/Series machine. State the built-in advantages of a book.
4. State your understanding of a Coupling Facility in a sysplex installation, and include a discussion of the three types of structures.
5. State your understanding of “scaling in” and “scaling out”.
6. Show your understanding of “serialization of resources”.
7. State your understanding of Workload Management.
8. Explain the Intelligent Resource Director.

4

# Integrity and security

**Objective:**

After completing this chapter, you will be able to:

- Describe the security and integrity needs of a large-scale operating environment
- Describe the methods for serialization in a multi-user environment
- Describe the built-in features that enable integrity and security
- Describe the two-phase commit process and why it is needed
- Describe the features needed from an add-on security package

## 4.1 Introduction to integrity

One of the ways the z/OS operating system provides data integrity is through Data Facility Storage Management Subsystem (DFSMS), which helps to automate and centralize the management of storage. To manage storage, DFSMS, also called SMS (for Storage Management Subsystem), provides the storage administrator with control over *data class, storage class, management class, storage group,* and *automatic class selection routine* definitions.

In this chapter we first discuss the ways in which z/OS contributes to the integrity of the data that it processes and the facilities it provides for frequent backups, auditing access to data, and logging changes to the data. Then we discuss security in 4.3, "Security" on page 79.

## 4.2 Integrity

Here we define two types of integrity:

| | |
|---|---|
| **Data integrity** | This is primarily concerned with the accidental damage and recovery of data.<br><br>One of the ways in which the operating system insures data integrity is by providing facilities for timely backup of the data. Data Facility Storage Management Subsystem (DFSMS) controls the creation and placement of data, provides a policy-based life-cycle management of data, and provides data backup facilities. |
| **System integrity** | This ensures there is no way for any unauthorized program to:<br><br>►Bypass store or fetch protection<br><br>►Bypass store or fetch protection<br><br>►Obtain control in an authorized state |

Auditing access to data is critical to ensuring the integrity of the data. The System Management Facilities (SMF) component tracks and creates audit records for every critical function of the operating system. SMF has many features, but here we focus on those related to data access.

Security products can prevent unauthorized read or write access to data, while creating logs that can be used to audit access to critical system resources.

## 4.2.1 Serialization

*Serialization* means make a resource to be used on a one-at-a-time basis. In a multi-user environment, it is possible for two or more tasks to attempt to update the same data at the same time. This would jeopardize the integrity of the data. z/OS has several methods for preventing tasks from concurrent updates: enqueue, reserve, lock, and global resource serialization services.

### Enqueue

Enqueues (ENQs) are designed as a protocol for sharing resources. Resources to be shared are given names that all users of the resource agree to use when they wish to access that resource. A request can be issued for *shared* or *exclusive* use, depending on whether or not the requester plans to make changes to the resource.

If someone has an exclusive ENQ, then no one else can have access to the resource and all other requesters will queue up and wait for the owner to free up the resource. If someone has a shared ENQ, other requesters who specify a shared ENQ can also access the resource. If someone requests an exclusive ENQ while there are one or more outstanding shared ENQs, then the requester, and any requesters that follow (shared and exclusive), must wait until all current shared ENQs have been released. In z/OS, ENQs are used to serialize on individual data sets (files).

ENQs can have a scope of step, system, or systems, as explained here:

- An ENQ with a scope of *step* prevents other tasks within the same unit of work from accessing the resource. Other users on the same system can access the resource.
- An ENQ with a scope of *system* takes effect for one system. Other systems can still access the resource even though a user has an exclusive ENQ for that resource.
- If a resource needs to be serialized across all systems that share it, then you can request an ENQ with a scope of *systems*. ENQs with a systems scope are propagated (passed) to all other systems that share the resource by Global Resource Serialization (GRS).

### Reserve

Reserves are used for serializing updates to DASDs that are shared between multiple z/OS images. A system that has a need to update control information about a DASD will issue a *reserve* against the device. If it is not reserved by another system, the requesting system is given a reserve against the device.

Reserve gives exclusive control of the device to the requesting system and locks all other systems out. More than one task on the owning system can access the device, and any updates to a data set on that system should be controlled by ENQing on the data set.

The use of reserve has performance implications, because it prevents all other systems that are sharing a device from accessing any file on the reserved device. Inappropriate use of reserve can cause systems to stop, waiting for access to a critical DASD unit. Some products, such as GRS, attempt to lessen the impact of reserves by changing reserves to global ENQs, which are less restrictive.

## Lock

A lock serializes the use of system resources by operating system routines and by processors. A *lock* is simply a named field in storage that indicates whether a resource is being used and by whom. Because locks use storage for serialization, they are much faster than ENQs, which is why they are used by system level tasks for resources that tend to be held for very short amounts of time.

Locks must be obtained in a certain order, also called the *lock hierarchy*. The purpose of the hierarchy is to try and prevent deadlocks. A deadlock occurs if one routine holds a lock and is looking for a lock that another routine is holding, while the second routine is looking for a lock that the first routine is holding. By requiring that locks be obtained in a specific order, the chances of deadlocks are greatly diminished, but not totally removed.

## Global Resource Serialization (GRS)

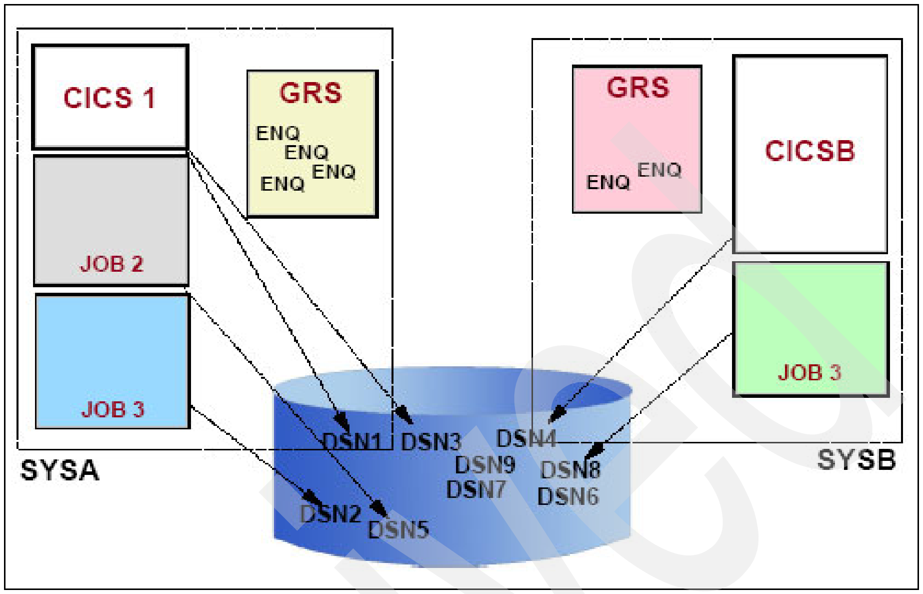


*Figure 4-1 Global Resource Serialization (GRS)*

In a multisystem sysplex, z/OS provides a set of services that applications and subsystems can exploit. Multisystem applications reside on more than one system and give a single system image to the users. This requires serialization services that extend across systems.

GRS provides the services to ensure the integrity of resources in a multisystem environment. Combining the systems that access shared resources into a global resource serialization complex enables serialization across multiple systems.

GRS expands the scope of ENQ serialization beyond a single system. Instead of ENQs serializing a resource on only one system, ENQs can be propagated by GRS to all of the systems that share that resource, as shown in Figure 4-1.

In particular, GRS can replace the need to use reserve to serialize on DASD volumes that are shared across systems. Instead of reserving the whole volume in order to update data contained in a data set on the volume, GRS can propagate the ENQ that was obtained for a data set. Because GRS enables users to serialize resources at the data set level, it can reduce contention for these resources and minimize the chance of an interlock occurring between systems.[1]

[1] Refer to IBM Redbook *ABCs of z/OS System Programming - Volume 5*, SG24-6985.

### 4.2.2 Data Facility Storage Management Subsystem (DFSMS)

DFSMS, or SMS for short, is the component of the operating system that controls all data set allocation and provides automatic backup for the data. Data Facility Product (DFSMSdfp™) is the heart of the storage management subsystem; it provides the logical and physical input and output for z/OS storage, keeps track of all data and programs managed within z/OS, and provides data access both for native z/OS applications and other platforms.

By controlling the devices on which data can be allocated, SMS can control the characteristics of the data. By moving critical data to devices with faster access, SMS can improve the performance characteristics of the data. By moving data to highly protected storage devices, SMS can provide extra security. By controlling data set management classes, SMS can provide automatic backup of the data.

SMS accomplishes these tasks through the use of Data classes, Storage classes, and Management classes, and by defining Automatic Class Selection (ACS) routines. Security products can provide different levels of access to different storage devices. Users requiring high levels of data security can use ACS routines, which are maintained by systems programmers, to define the storage devices on which a data set can be allocated, put critical data sets on highly protected volumes, and keep less critical data sets off these volumes.

To implement policies for managing storage, the storage administrator defines a set of SMS constructs that can apply different data placement and management policies based on the specific requirements of the data. Once defined, the storage administrator then can save those definitions in a central repository called the SMS Control Data Set.

The storage administrator can then write automatic class selection (ACS) routines that use naming conventions, or other criteria to automatically assign the classes that have been defined to data as it is created. The constructs that are assigned to the data at creation will then be used to manage that data throughout its life cycle.

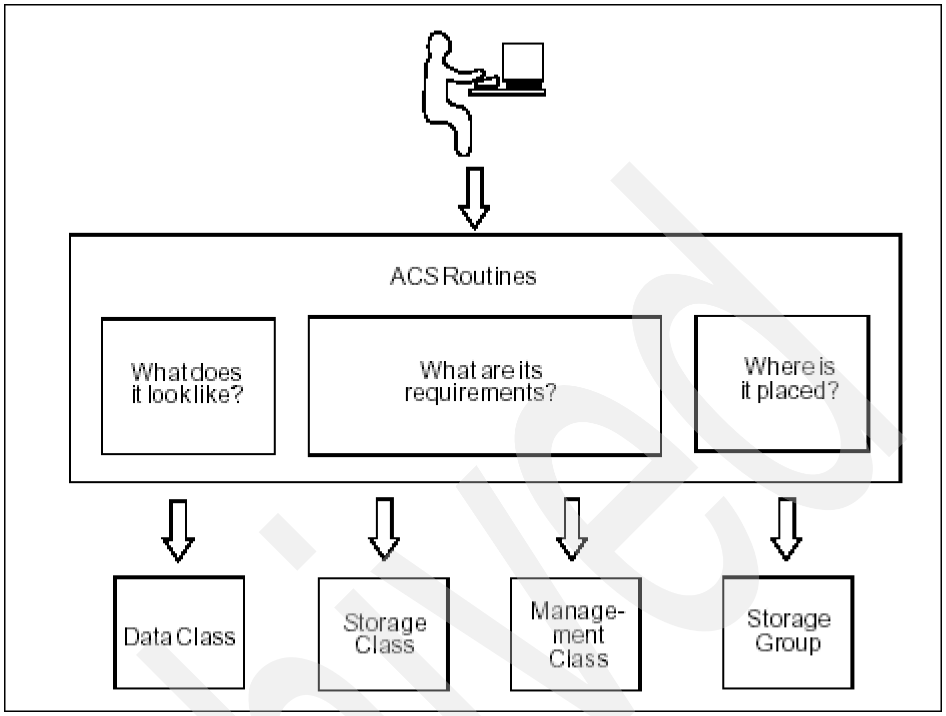


*Figure 4-2 SMS constructs*

The following is a brief description of each of the SMS constructs[2], shown in Figure 4-2.

## Data class

Data classes define allocation and space attributes that the system uses when creating a data set.

## Storage class

Storage classes define performance goals and device availability requirements that will be associated with the data set. Storage class is a required part of an SMS-managed environment. A storage class provides the criteria that SMS uses in determining an appropriate location to place a data set or object.

SMS uses the storage class to select a device according to the following criteria: device performance, the amount of space available on the volume, and how available a data set or object can be on that device.

[2] Refer to the IBM Redbook *DFSMShsm Fast Replication Technical Guide*, SG24-7069.

## Management class

Management classes define attributes that are used to manage the data set. These attributes are used to control retention, migration, backup, and release of allocated but unused space. Using the characteristics defined by the management classes, SMS can automatically back up data at whatever level of criticality is needed. Data can be backed up on any time schedule—nightly, weekly, monthly, and so on. In addition, copies of the backup tapes can be automatically shipped to offsite storage for disaster backup and recovery.

## Storage Group

A Storage group represents the physical storage that SMS manages. This storage can be collections of DASD volumes, tape volumes in tape libraries or optical volumes in optical libraries. The storage administrator groups the volumes to meet a specific service or business requirements.

## ACS routines

ACS routines pull it all together. They are used by system administrators to automatically associate SMS classes to data sets based on their space, performance, and availability needs. Numerous selection criteria are available for the ACS routine to determine how to associate particular SMS classes to a given data set.

There are two other specifications of SMS known as aggregate group and copy pool, which are described here:

## Aggregate group

The aggregate group defines a collection of data that will be handled as a single entity for disaster recovery, application transfer, application archiving, or data migration and defines a set of management characteristics to this data collection.

Often the aggregate group encompasses all the data belonging to a specific application, such as payroll. This allows all the data in the payroll application to be backed up for disaster recovery or be transferred from one location to another.

## Copy pool

A copy pool is a defined set of storage groups that contain data that can be backed up and recovered collectively using functions of the disk subsystem that allows a large amount of data to be copied very rapidly.

### 4.2.3 Auditing

*Auditing* is the process of ensuring that the information processing system (hardware, software, liveware, middleware, policies, and procedures) complies with the installation security policy. Auditing may be:

- A one-time project, such as a snap inspection, or
- An ongoing process, pursuant to policies

The two types of information security audits can be termed preemptive and reactive. As their names indicate, *preemptive* audits test security controls, and *reactive* audits respond to potential security breach events. Incident Response is an integral part of the security management plan.

Some companies resist implementing information security controls because they believe the costs are prohibitive. The cost in reactive audits, information compromise, lawsuits and fines for non-compliance and loss of business are such that it is becoming ever more painfully clear that an information security program is part of the normal cost of doing business.

#### System Management Facilities (SMF)

System Management Facilities (SMF) is a system component that writes audit records to DASD for every access or attempted access to a resource. These records can be offloaded to tape automatically when the SMF data set fills up. Detailed information about the access is recorded so that, in the event that it is needed, systems programmers can determine who accessed (or attempted to access) critical resources.

#### System logger

In order to be able to back out erroneous changes or to restore data up to the point of failure, it is necessary to log changes to the data. The operating system provides logging services through the system logger. It provides interfaces to define and update log files for applications and operating system components, and can synchronize the logging of updates to data sets across a sysplex. It is also necessary to log data for billing and reporting. Thus other user-written logs can now be recorded by the logger.

In the event that a data set must be restored from a backup tape, the log provides a means for reapplying all the updates to the data set since the last backup was taken.

### 4.2.4 Resource Recovery Services (RRS)

With the increasing number of resource managers (for example CICS, IMS, DB2, Websphere MQSeries®, and so on) now available on z/OS, there was clearly a need for a general sync point manager on z/OS that any resource manager could exploit. This is the role of Resource Recovery Services (RRS), a component of z/OS. RRS provides a global sync point manager that any resource manager on z/OS can exploit. It enables transactions to update protected resources managed by many resource managers.

Data does not reside in only one place. When data is updated, it generally is necessary to update more than one file at the same time. These files may reside on different systems and be managed by different applications. For instance, when you move money from your savings account to your checking account, the changes to the two files must be coordinated so that the balances are correct. If the update to the checking account fails for some reason, then the update to the savings account must be backed out. This coordination is accomplished by a protocol called "two-phase commit." This protocol assures that all updates are coordinated correctly and that data integrity is maintained.

RRS assists in the two-phase commit process by acting as the *sync point manager*, coordinating the commit or backout process for the various applications; see Figure 4-3. Note that the applications must still perform the commit or backout themselves—RRS only acts as a mediator between the applications.

## Two-phase commit

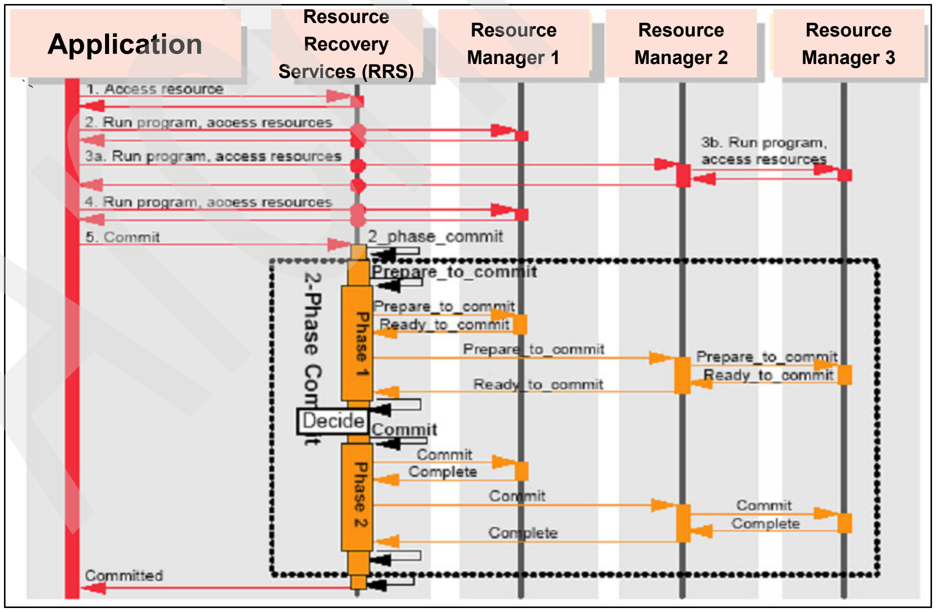


*Figure 4-3 Two-phase commit*

The two-phase commit protocol, as shown in Figure 4-3, is a set of actions used to make sure that an application program either makes *all* the changes to the resources represented by a single unit of recovery (UR), or makes *no* changes at all. This protocol verifies that either all changes or no changes are applied even if one of the elements (such as the application, the system, or the resource manager) fails. The protocol allows for restart and recovery processing to take place after system or subsystem failure.

The two-phase commit protocol is initiated when the application is ready to commit or back out its changes. The application program that initiates the commit or backout does not need to know who the sync point manager is or how two-phase commit works. This is hidden from the application; a program simply calls a commit or backout service and receives a return code indicating whether this has been successfully completed.

When the application issues the commit request, the *coordinating recovery manager*, also called the *sync point manager*, gives each resource manager participating in the unit of recovery an opportunity to vote on whether its part of the UR is in a consistent state and can be committed.

If all participants vote yes, the sync point manager instructs all the resource managers to commit the changes. If any participants vote no, the sync point manager instructs them to back out the changes.

## 4.2.5 Data backup and recovery

The z/OS operating system and its associated disk and tape subsystems provide a robust suite of tools that provide for data backup utilities to meet a wide range of data backup requirements, from disaster recovery to local recovery.

Installations need to choose the backup and recovery solution that best fits their needs. The choice that an installation makes will be based on factors such as:

1. Is the backup being performed in order to recover from local outages, or for disaster recovery?
2. Does the backup and recovery need to be device-independent?
3. What level of granularity is required: single data set, volume, application or site?
4. What is my *recovery time objective*? That is, how much time do I have to recover the data after an outage?
5. What is my *recovery point objective*? At what points in time do I make the backup, so that when recovery is performed, the processing of the data can resume where it left off at this chosen point in time.

6. How much money is the company willing to spend on a solution? This would be equivalent to you buying an insurance policy: the larger the negative economic impact to the company or individual, the more they are will to pay for the insurance policy. The same holds true for backup and recovery solutions.

This following sections introduce some of the backup and recovery tools that are available in the IBM System z hardware and z/OS operating systems. The toolbox contains a full complement of tools, so it is up to your installation to select the right tool to match its needs.[3]

## FlashCopy - (local to one subsystem)

FlashCopy® is a function available on some DASD subsystems. DFSMS uses this function to quickly copy data from source volumes to target volumes that are within the same physical subsystem. Flashcopy makes use of the internal structure of Redundant Array of Independent Disks (RAID) devices to increase the speed of the copy function.

The data is not actually copied; only the control information that points to the data in the RAID device is copied. If the data is changed using the source volume, the changed data is written to a different location on the RAID device, leaving the original version unchanged and still accessible using the target volume.

After the copy has been made, the copy can be accessed for creating tape backups without impacting any applications using the original copy of the data.

## Remote Copy

Remote copy in the z/OS operating system environment is a set of storage-based disaster recovery, business continuance and workload migration solutions that allows the copying of data to a remote location in real time. Having a copy of the data in a remote location protects business data from both local and regional disasters that could render the original processing location inoperable.

Remote copy refers to the four main copy services functions provided in z/OS that are called Advanced Copy Services functions. They are:

1. Metro Mirror (also known as synchronous Peer-to-Peer Remote Copy or PPRC)
2. Global Mirror (also known as asynchronous PPRC)
3. Global Copy (also known as PPRC-Extend Distance or PPRC-XD)
4. Global Mirror for IBM System z (also known as Extended Remote Copy or XRC)

[3] Refer to the IBM Redbook *The IBM TotalStorage DS6000 Series: Copy Services with IBM eServer zSeries*, SG24-6782.

There are other remote copy functions which combine elements of those listed above, including:

1. Metro/Global Copy (Metro Mirror and Global Copy)
2. Metro/Global Mirror (Metro Mirror and Global Mirror)

## Metro Mirror

Metro Mirror is a hardware solution that provides rapid and accurate disaster recovery, as well as a solution for workload movement and device migration. DASD volumes residing on different physical subsystems are paired. Updates made to the primary DASD volumes are synchronously duplicated on the secondary DASD volumes.

Note that, because the operating system has to wait while the updates are synchronized, there is a performance penalty for using Metro Mirror over distances greater than 10 km. The local storage subsystem and the remote storage subsystem are connected through a communications link.

By providing a duplicate set of volumes in a remote location, Metro Mirror can be used for disaster recovery. In the event of a disaster at your primary site, you can switch to your remote site without losing any data.

## Global Copy

Global Copy is a hardware solution similar to Metro Mirror, except that the primary and secondary sites can be separated by greater distances through the use of channel extenders and telecommunications lines. In this solution, the primary site’s disk subsystem updates the secondary site’s disk subsystem incrementally, thus reducing the bandwidth requirements.

## Global Mirror

Global Mirror combines Global Copy with Flashcopy to provide backup at a remote secondary site. The key to this solution is that the update of the secondary site is asynchronous from the updates at the primary site, and there is minimal impact to the application performance, even at greater distances.

## Global Mirror for IBM System z

Global Mirror for IBM System z (also known as Extended Remote Copy or XRC) is similar to Metro Mirror in that it maintains a copy of DASD data at a remote location. However, since it is asynchronous, it can be used over greater distances. With Global Mirror the data is not kept quite as up-to-date as with Metro Mirror.

Global Mirror for IBM System z is a hardware and software solution that relies on a function called the *system data mover*, which is part of DFSMS. It is a high-speed data movement program that efficiently and reliably moves large amounts of data between storage devices. Global Mirror for IBM System z is a continuous copy operation, capable of operating over long distances. It runs unattended, without involvement by the application users. If an unrecoverable error occurs at your primary site, the only data that is lost is data that is in transit between the time when the primary system fails and the recovery at the recovery site.

### Other backup and recovery solutions

Rather than using one of the hardware or hardware or software-based solutions mentioned in the previous sections, your installation might decide that a software-only solution would best suit their backup and recovery needs. Software-only solutions tend to be less costly than some of the solutions mentioned, but again the cost of the solution needs to be measured against the financial impact of the data loss. The solutions discussed here can create backups and allow recovery at various levels of data aggregation.

- **Data set level backup and recovery** - This allows for backing up and recovering a single data set.

  In some cases the data backup can be a physical process or it can be a logical process. The advantage of a logical process is that it can be device-independent; that is, the data could be backed up from one type of disk or tape subsystem and recovered to a disk or tape subsystem with totally different device characteristics and device geometry.

- **Volume level backup and recovery** - This allows for backing up and recovering data at a volume level.

  All the data on a disk or tape volume gets backed up in a single process. In some instances only a single data set or group of data sets can be restored from the volume that was backed up, or the entire contents of the volume can be restored in a single process.

- Application level backup and recovery - This provides backup and recovery at an aggregate level.

  Data for an entire application or group of applications could be backed up and restored as a single entity. This allows an enterprise to identify only those applications that are critical to the continuance of their business if an outage occurs, and only back up and recover those applications. This can be a very cost-effective strategy since only a subset of the data in the installation is backed up, rather than all the data that exists are their location.

## 4.2.6 Performance

DFSMS works with the operating system to monitor performance characteristics of each DASD volume on the system. Users, in conjunction with system programmers, can specify the performance and availability needs for individual data sets. These needs are defined via Storage classes. SMS uses Storage classes, along with system performance statistics, to determine if data accesses are meeting the goals defined for the storage class. If they are not meeting goals, then SMS can automatically move the data set to a DASD that will provide the necessary performance for the data.

## 4.3 Security

This leads us into a discussion on security. The first step in providing for data integrity is to adequately secure access to the operating system. If unauthorized users cannot log into the operating system, they cannot access the data. By providing a robust set of security features, the operating system insures that data is accessed and updated only by authorized users.

Access to data is tracked by several components of the operating system: System Management Facilities (SMF), System Authorization Facility (SAF), and Resource Access Control Facility (RACF®). These components provide an audit trail that can be used to detect attempts at unauthorized access to data, and can detect patterns of access by authorized users that are beyond the scope of their valid use of the data.

### 4.3.1 Introduction

z/OS is among the most secure operating systems in the world. Since its inception, z/OS has put security at the forefront, often sacrificing performance and ease of use in order to assure the integrity of the system. Security methods are buried deep within the components of the operating system, providing the highest level of security possible. Operating system components work together with security products to accomplish this goal:

- The U.S. Government Common Criteria Program certifies IBM Software at the highest rankings
  - z/OS and RACF are at the highest level of certification (EAL4+).
- Enforced Isolation
  - Each user has its own unique address space.
  - Each LPAR provides total isolation executing as a separate server.
  - Each address space within an LPAR is isolated from the others.
- The Access Control Environment Element
  - The z/OS Security Control Block is protected by the operating system.

- The Authorized Program Facility (APF) assures that system-level services can only be accessed by programs that are authorized to use those services.
- Program States assure that user programs cannot access resources that belong to the operating system.
- Storage protect keys assure that programs cannot access storage that does not belong to them.
- The z/OS supervisor call (SVC) instruction provides a facility for user-written programs to get access to system-level functions, without being able to perform those functions themselves.
- System Authorization Facility (SAF) provides the framework for security in the operating system. It works in conjunction with external security software packages (RACF, TOP SECRET, ACF2, and so on) to provide access services for all resources available in the complex.

### 4.3.2 U.S. Government certification

The Evaluation Assurance Level of a computer product or computer system is a numerical grade assigned following the completion of a Common Criteria security evaluation, which is an international standard in effect since 1999. The computer system must meet specific assurance requirements involving design documentation, design analysis, functional testing or penetration testing. The higher the EAL, the more the stringent qualifications are imposed.

The United States Government has rated the mainframe secured, certifying it with the Evaluation Assurance Level (EAL4+). That is, the mainframe as a platform offers EAL4 certification, while the logical partition under PR/SM contains a rating of EAL5 certification.

EAL4 permits a developer to gain maximum assurance from positive security engineering based on disciplined commercial development practices which, though rigorous, do not require substantial specialist knowledge, skills and other resources. EAL4 is the highest level at which it is likely to be economically feasible to retrofit to an existing product line.

EAL5 permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practices supported by judicious application of special security engineering techniques. EAL5 is therefore applicable in those circumstances where developers or users require a high level of independently-assured security in a rigorous development approach, without incurring unreasonable cost attributable to extraordinary security engineering methods.

**Important:** Only the IBM mainframe partitions have attained an EAL5 rating.

### 4.3.3 Enforced isolation

#### Unique address space

The range of virtual addresses that the operating system uniquely assigns to users or programs is the address space. This is an area of contiguous virtual addresses available for executing instructions and storing data. Using address spaces provides the means for distinction between the programs and data executing within a logical partition. This is the method z/OS uses to establish a private container for processing each end-user request, and therefore ensures runtime protection by the operating system's architecture.

#### LPAR as a separate server

Among the system control functions is the capability to divide the system into logical partitions (LPARs). An LPAR is a subset of processor hardware that is defined to support an operating system. An LPAR contains resources (processor, memory and input/output devices) and operates as an independent system. Multiple logical partitions can exist within a mainframe hardware system.

LPARs provide the means for high availability employing cloning techniques by which an LPAR, during failure or due to maintenance schedules, can switch to another executing partition without end-user disruption.

LPARs are, in practice, equivalent to separate mainframes. Each LPAR runs a copy of the operating system and is IPLed separately from other active LPARs. This can be any operating system and at different version and release levels.

### 4.3.4 The accessor environment element

In z/OS there is a credential control feature known as the accessor environment element (ACEE), which accompanies a logical process or user request within the operating system. It is available to identify the authenticated user access control authorization checking to z/OS Resource, and is used by various auditing methods.

The user process is assigned an ACEE credential, which embodies the "identity" that has previously been assigned to the user by a security administrator. Note that this is *not* a programming interface and is used by the operating system on behalf of the user.

### 4.3.5 Authorized program facility (APF)

In z/OS, APF is a facility that permits the identification of programs that are authorized to use restricted functions. APF was designed to restrict access to sensitive, or *privileged*, system instructions in order to avoid integrity exposures that might occur as a result of the use of these instructions. Privileged instructions allow a high level of access to powerful system functions and can, therefore, cause security or integrity problems if they are used inappropriately.

In order for a program to run *authorized*:

- It must have the APF attribute assigned to it.
- It must also reside in a library that has been defined by the system programmer as being authorized.

If a program fulfills both of these requirements, then it can access privileged instructions.

APF libraries are always given a high level of protection by whichever security software is being used on the system, in order to prevent their misuse.

### 4.3.6 Program states

Every program running in the system runs in a particular program state. System resources are protected by only allowing programs running in *supervisor* state to have access to them. This provides an extra layer of protection for system resources. In order to run in supervisor state, a program must be APF-authorized and loaded from an APF-authorized load library.

Operating system programs run in supervisor state, which gives them access to all of the resources on the system. This is somewhat analogous to running as a superuser in UNIX or Linux.

Programs that have been loaded from non-APF-authorized libraries can only run in *problem program* state, which gives them very little access to system resources. If they need to access a protected resource, they must request that an operating system program that is running in supervisor state access the resource on their behalf. The problem state program never gets access to the resource. All access is by the operating system program.

### 4.3.7 Storage protection keys

Mainframe (real) storage is divided into *frames*, which are 4096-byte chunks of storage. Every frame of real storage has a field that defines the storage protection key for that frame. (They are often called “storage protect keys.”)

Protection keys are numeric, 0 through 15. Keys 0 through 7 are *system keys* and can only be accessed by operating system programs running in supervisor state. Keys 8 through 15 are *user keys* and can be accessed by programs running in problem state. Most user-written programs use Key 8.

If a program attempts to access storage in a key that is different than the one it has been designated to use, the operating system will abend (Abnormal END) the program. The storage protection key for a program is defined in the program status word (PSW), which is an operating system element that controls program execution. See “The program status word” on page 162 for more information about this topic.

## 4.3.8 Supervisor call

A supervisor call (SVC) is a special privileged instruction that interrupts the program being run and passes control to the supervisor so that it can perform the specific service indicated by the instruction. The supervisor can be called by other programs, such as problem state programs, to perform specific system-level functions that the calling program is not authorized to perform.

For instance, if a problem state program wants to access a data set, it must issue an SVC, since problem state programs are not allowed to access data sets directly. SVCs are highly protected and must be loaded from a specific set of program libraries that are defined by the systems programmer.

Use of SVCs can be restricted by requiring that the calling program be APF-authorized. By isolating system functions from problem state programs, SVCs provide another layer of protection for operating system resources.

## 4.3.9 System Authorization Facility

System Authorization Facility (SAF) is the function of the operating system whose sole purpose is to provide security for resources. The operating system calls SAF whenever a user is going to access a protected resource, such as a data set, a program, or the system itself. SAF, in turn, will call an external security product, if one has been installed, to either authorize or deny the access. If the user is not authorized to access the resource, the request is denied.

All users of the operating system must be defined to the security product and granted access to the operating system before they can attempt to access any resource.

External security products such as RACF and TOP SECRET are used to define users of resources and their levels of access, or non-access, to the resources.

### Packaged security - Resource Access Control Facility (RACF)

As previously mentioned, some security functions are provided in the base operating system and various add-on z/OS products. Others are packaged in an optional Security Server feature. Chief among the functions packaged within the Security Server is the Resource Access Control Facility (RACF).

RACF incorporates various elements of security, such as user identification and authentication along with access control. You use RACF to protect resources and information by controlling access to those resources.

RACF identifies you when you log on to the operating system. It does so by requiring a user identification, the user ID that is a unique identification string. RACF then verifies that you are the user you say you are by requesting and checking passwords for personal accountability.

RACF also enables an installation to define what authorities you have, or what authorities a group to which you belong has. It controls what you can do on the system. Some individuals have a great degree of authority, while other have little authority. The degree of authority you are given is based on what you need to perform your job role.

Besides defining user and group authorities, RACF protects resources. A *resource* is considered your organization's information stored in its computer system, such as datasets or databases.

RACF stores all this information about users, groups and resources in profiles. A *profile* is a record of RACF information that has been defined by the security administrator. There are user, group, and resource profiles.

Using information in its profiles, RACF authorizes access to certain resources, therefore RACF applies user attributes, group authorities and resource authorities to control use of the system.

## 4.4 Summary

System z mainframes are considered the most secure systems because their architecture has integrated data integrity and security throughout its hardware and software components.

We have seen how System z manages the concept of serialization. It can lock a single piece of data (rather than an entire file), mainly for update cases when one user must have sole access to that specific data.

Through mechanisms like ENQs, locking, reserve, and global resource serialization, for example, the information is protected providing security and data integrity.

We described how SMS works through the use of data classes, storage classes, and management classes. We have seen that SMS uses automatic class selection (ACS) routines to control and improve performance characteristics of the data, giving extra security and automatic backup of the data.

System resources are protected by allowing programs to run in supervisor state only if they are loaded from an APF-authorized library. We have seen that an SVC can be called to ask the system to perform actions that are not allowed by a general user.

Finally, we have seen how system storage is protected through the internal use of storage protection keys in the PSW that controls program execution.[4]

| **Key terms in this chapter** | | | | |
|---|---|---|---|---|
| ACEE | EAL | GRS | Remote copy | SMF |
| APF | ENQ | Integrity | RRS | Storage protect key |
| Audit | Remote Copy | Lock | SAF | SVC |
| Backup | FlashCopy | Program states | Security | System logger |
| DFSMS | Global Mirror | RACF | Serialization | Two-phase commit |

## 4.5 Questions for review

To help test your understanding of the material in this chapter, complete the following review questions:

1. What is the difference between integrity and security?
2. What facilities can be used to back up data in a mainframe environment?
3. What are some of the questions that facilities must address in order to select the correct backup and recovery solution to meet their needs?
4. How do ENQs, reserves, and locks differ? What is each used for?
5. How do the various SMS constructs such as Data classes, Management classes, and Storage classes work together to define data set attributes?
6. Explain how two-phase commit works to keep data in sync across different applications.

[4] For more information about integrity and security, refer to IBM Redbook *Introduction to the New Mainframe: Security*, SG24-6776.

7. How does APF work to protect the operating system?
8. What is an SVC? How do SVCs aid security?
9. What is the purpose of SAF?

5

# Availability

**Objective:**

After completing this chapter, you will be able to:

- Understand what availability means to a commercial enterprise
- Describe the inhibitors to availability
- Describe operating system facilities that improve availability
- Describe the major components of Parallel Sysplex

## 5.1 Introduction to availability

In simplest terms, *availability* means how much the computer system is available to its users, and the ideal is 100% of the time. This chapter discusses the availability needs of commercial data processing and the evolution of the hardware and operating system to fulfill these needs.

The transition of businesses to an online 24/7 environment is briefly discussed here, along with the forces that are driving this requirement: the World Wide Web, a worldwide client base, and regulatory requirements. However, our primary focus is on the inhibitors to system availability and the changes to the operating environment that enable it to overcome these problems. We discuss the evolution from single CP systems to Parallel Sysplex, and the changes to hardware and software that made this evolution possible.

Commercial data processing has evolved from an environment where updates were gathered during normal daytime business hours and run overnight in a mass update mode (batch), to an online, real-time, worldwide, 24-hour a day environment where updates need to be made immediately. Clients want their transactions to be handled as soon as they are entered, and government regulations in certain industries, such as banking, mandate high levels of availability.

These changes have forced the operating system to evolve as well. Businesses can no longer tolerate operating system downtime due to software and hardware errors or upgrades.

## 5.2 What is availability?

Webster's Dictionary defines availability as "being accessible for use"[1]. In a computer processing environment, however, availability is not always so clear-cut. If the operating system is running, is that availability? What if the network is unavailable and the end user cannot connect? What if all of the hardware components are running up but the application is down? Availability is dependent on who the end user is. If a user connected via the intranet can access a particular application, then it is available to that user. The same application may not be accessible via the Internet due to a bad connection, so to an Internet user the application is not available.

Availability is the state of an *application* being accessible to the *end user*. In the mainframe world there are additional definitions for different level of availability, as described here.

[1] http://www.websters-online-dictionary.org/definition/AVAILABILITY

### *Continuous Availability (CA)*

This refers to the attribute of a system to deliver nondisruptive service to the end user 7 days a week, 24 hours a day (there are no planned or unplanned outages).

### *High Availability (HA)*

This refers to the attribute of a system to provide service during defined periods, at acceptable or agreed-upon levels, and mask unplanned outages from end users. It employs Fault Tolerance, Automated Failure Detection, Recovery, Bypass Reconfiguration, Testing, Problem and Change Management.

### *Disaster Recovery (DR)*

This refers to recovery after a disaster, such as a fire, that destroys or otherwise disables a system. Disaster Recovery techniques typically involve restoring data to a second (recovery) system, then using the recovery system in place of the destroyed or disabled application system.

## 5.2.1 Outages - planned and unplanned

An outage (unavailability or "downtime") is the amount of time a system is not available to an end user. Outages may be planned or unexpected (unplanned). Planned outages include causes like database reorganization, release changes, and network reconfiguration. Unplanned outages are caused by some kind of a hardware, software or data problem. While planned outages can be scheduled, they still are disruptive. The modern trend is to try to avoid planned outages altogether. This requires extensive hardware and software facilities.

### *Cost of an outage*

In today's computer environment an outage, whether planned or unplanned, is not only unwelcome—it is also costly. Downtime statistics are staggering and range from tens of thousands of dollars to multiple millions of dollars per hour of downtime; see Table 5-1.

*Table 5-1 Financial Impact of Downtime Per Hour (by various Industries)*

| Industry | Cost of Downtime $ per hour |
|---|---:|
| Brokerage Retail | 6.5 million |
| Credit Card Sales Authorization | 2.6 million |
| Airline Reservation Centers | 90,000 |
| Package Shipping Services | 28,250 |
| Manufacturing Industry | 26,761 |

| Industry | Cost of Downtime $ per hour |
|---|---|
| Banking Industry | 17,093 |
| Transportation Industry | 9,435 |

Source: ©Eagle Rock Alliance, Ltd. All Rights Reserved. Reprinted by Permission.

And although the financial impact is significant, the damage can extend well beyond the financial realm into key areas of client loyalty, market competitiveness, and regulatory compliance.

## 5.3 Inhibitors to availability

Factors that can cause loss of availability include hardware failure, software failure, and environmental problems. Computers, even the most reliable, may occasionally fail. Programs may have errors that cause them to fail. Cables may accidentally become disconnected or cut.

Availability comes in many forms on the mainframe. The mainframe has the distinction of the five 9s (see *Myth of the nines*[2]) of availability (99.999%) for up time of annual service for *both* software and hardware. This translates to 5.3 minutes of downtime yearly, which can likely be shortened further due to the means of applying maintenance and upgrades concurrently while the system is running.

Although known for its resilience in availability, application inhibitors can affect the mainframe's inherent capability for uptime. The mainframe is architected with redundant components, but application designers may overlook this form of planning and instead rely on other methods such as rebooting a server instance.

In order to ensure infrastructure and application availability, the mainframe using z/OS provides for cloned LPAR images to work as a single instance for an application environment. This is referred to as a Parallel Sysplex, where two or more LPARs are copies of one another running the same applications. This is also known as a *cluster-type solution*.

Parallel Sysplex provides the means to operationally continue the workload in the executing partition in the event the other image fails. The key to a transparent failover is to ensure application inhibitors are not employed in the design.

Note the considerations for the following areas:

- Affinities

2 http://en.wikipedia.org/wiki/Myth_of_the_nines

- Either inter-transaction or system-image affinity may have a consequence for failover integrity, such as using a component (physical) or name (logical) unique to a specific system or runtime environment.

- Using a memory counter.
- Time stamps between server environments (timers and expiration events).
- Parallel execution may have a consequence in that you cannot assume the next transaction will always run on the same system:
  - Keeping working storage or temporary areas specific to a system (pinned memory) may cause unpredictable results in the event another image needs to run the workload.
  - Do not assume the execution of work will always be the same as the order of arrival.
  - Do not design an application associated to a particular version or software level.
  - Design for data sharing, where a single data instance can be used by multiple applications.

There is no way to totally avoid failures in any single component, but there are ways to lessen the impact of these failures. Redundancy, error detection and recovery, and continuous monitoring of critical resources are some of the ways to achieve high availability.

## 5.4 Redundancy

The IBM mainframe product line offers layer upon layer of *fault tolerance* and *error checking*. If a failure occurs, the *built-in redundancy* of the IBM System z shifts work from failing components to components that work in order to prevent the end-user service from being interrupted. The failed components may be removed and replaced while the processor is still active, so service may continue.

Hardware failures have been reduced by engineering redundancy into each of the major components to avoid any single point of failure. Modern direct access storage devices (DASD) have been architected with hardware redundancy and built-in error correction microcode.

The IBM mainframe strategy employing reliability, availability, and serviceability (RAS) is focused on a recovery design that is necessary to mask errors and make them "transparent" to client operations. An extensive hardware recovery design has been implemented to detect and correct array faults. In cases where total transparency cannot be achieved, the capability exists for the client to restart the server with the maximum possible capacity.

IBM System z computers have backups for all of critical components, and new functions regarding redundancy continue to evolve.

**Power supplies** The IBM System z has dual power supplies with separate power cords. Should one power supply fail, the other can handle the power requirements for the whole computer. The server can continue operation after a failure of one of the two power units. The power units are designed so that the server will continue to operate after a failure of one phase of a three-phase feed.

**Internal battery** In the event of an interruption to the input power, the Internal Battery Feature (IBF) will provide sustained server operation for a short period of time. The duration of the battery support is highly dependent upon the server model and I/O cages installed. The IBF is optional.

**Note:** Many power interruptions are due to the temporary loss of a single phase on the three-phase input line. The IBM System z can tolerate loss of a single phase even with no battery backup. This capability, coupled with the Internal Battery Feature and full operation from either of the dual independent line cords, gives a very high degree of resilience to transient power dropouts. When coupled with an Uninterruptible Power Supply (UPS) capability, the internal battery ensures that there is no loss of power during startup of the emergency supply.

**Cooling** The current IBM System z is an air-cooled system assisted by two Modular Refrigeration Units (MRU). If one of the MRUs fails, backup Motor Scroll Assemblies (MSAs) are switched in to compensate for the lost refrigeration capability with additional air cooling. At the same time, the oscillator card is set to a slower cycle time, slowing the system down by up to 10% of its maximum capacity, to allow the degraded cooling capacity to maintain the proper temperature range. Running at a slower cycle time, the MCMs produce less heat. The slowdown process is done in steps, based on the temperature in the books.

**Oscillator** IBM System z has two oscillator cards: a primary and a backup. In the event of a failure of the primary oscillator card, the backup is designed to detect the failure, switch over dynamically, and provide the clock signal to the server transparently.

**Processor**

Early mainframe computers were uniprocessors with only one central processor (CP). If the processor failed, all of the work being run on the system was lost. Users needed more reliability and more processing power than a uniprocessor could offer, so multiprocessors (MP) were introduced.

Having more than one processor gave the hardware designers the ability to write code that could self-monitor the health of the mainframe. CPs signaled each other to make sure that they were still in operation. If one CP failed, the others could take action to recover or isolate the failing CP. This greatly enhanced the reliability of the hardware, but since work was confined to an individual mainframe, there was still the possibility of a system failure causing downtime.

Today, mainframe computers have spare CPs built into the framework of the Multiple Chip Module (MCM). This is known as *CP sparing*. Should one CP fail, others can automatically be brought online to take over the workload. This concept will also work for the speciality engines.

**Memory**

There are *spare memory chips* on every memory card. The memory cards are continuously being checked to clean correctable errors and detect uncorrectable ones.

If an error cannot be corrected, the chip is made unavailable and a spare is used instead. This results in four spares per processor memory array (PMA) with Chipkill™. (Chipkill memory is a design feature that packs memory in such a way that in the event of a chip failure only one bit in an Error Correction Code (ECC) chain will be affected.)

On machines having more than one book sharing their L2 memory, another source of error can be the access path. To prevent this, there are two parallel paths.

**Books**

In newer mainframe computers, processors and memory have been componentized into modules referred to as *books*. Each book is manufactured with several CPs (including spares), memory, high-speed cache, and input/output (I/O) interfaces.

| | |
|---|---|
| | While books communicate with other books in the mainframe, they are separate entities and, for the most part, can be replaced without needing to bring down the whole system. |
| **Support Elements** | Two ThinkPad Support Elements (SEs) are mounted behind the CPCs cover. One ThinkPad acts as the primary (active) SE and the other acts as an alternate (spare). |
| | The primary SE is used for all interactions with the System z server. The alternate SE has a special workplace with limited tasks available. |
| | Both SEs are connected to the CPC, via the Power Service and Control Network (PSCN). For certain SE errors, an automatic switch-over is made to assign the formerly alternate SE as the primary. Automatic mirroring copies critical configuration and log files from the primary SE to the alternate SE two times a day. |

To learn more technical details about IBM System z redundancy, refer to the IBM Redbook *IBM System z9 Enterprise Class Technical Guide*, SG24-7124.

### 5.4.1 Concurrent maintenance and upgrades

In addition to the hardware redundancy aspects, the IBM System z server has the ability to permit a variety of concurrent maintenance and upgrades:

| | |
|---|---|
| **Duplex units** | Duplexed units, such as power supplies, allow one unit to be replaced while the other is operational. |
| **Microcode update** | In many cases, new levels of driver code may be installed concurrently with server operation. |
| **I/O cards** | When slots are available on the installed I/O cages, additional I/O cards can be added. For some special I/O cards, there are spare ports on the card which can be activated. |
| **Memory** | Concurrent memory upgrade or replacement is allowed. Memory can be upgraded concurrently using LIC-CC if physical memory is available on the books. |
| **PU Conversion** | Concurrent conversion between different PU types (CPs, ICFs, IFLs, zAAPs, zIIPs) is allowed, providing flexibility to meet changing business environments. |
| **Capacity upgrade** | The IBM System z provides concurrent, on-demand upgrades for the server hardware. With operating system |

support and appropriate planning, concurrent upgrades can also be nondisruptive to the operating system. Such upgrades provide additional capacity (memory, processors) without a server outage.

Capacity upgrades can be either permanent or temporary:

Permanent
- Capacity Upgrade on Demand (CUoD)
- Customer Initiated Upgrade (CIU)

Temporary
- On/Off Capacity on Demand (On/Off CoD)
- Capacity BackUp (CBU)

Refer to 3.3.6, “Workload Manager (WLM)” on page 56 to learn more about CUoD, CIU, and OOCoD. Concepts such as dynamic, nondisruptive addition of processor resources like processors, memory and I/O for permanent use reduce the need for planned outages.

#### Capacity Backup (CBU)

Capacity Backup (CBU) is the temporary activation of CPs, IFLs, ICFs, zAAPs and zIIPs for robust disaster recovery. The CBU features provide the ability to concurrently increment the CP or specialty engine capacity of your System z server, using Licensed Internal Code, Control Code (LIC-CC), in the event of an unforeseen loss of substantial System z computing capacity at one or more sites. The CBU features contain additional resources and alter the target server to an agreed-upon configuration for up to a 90-day period.

### 5.4.2 Accessing peripheral devices

In a continuous availability environment, keep the following points in mind when configuring the peripherals.

- Create a configuration that will survive the loss of any single component within the subsystem.
- Configure your storage subsystem such that you can survive, or at least recover from, a complete loss of the whole subsystem.
- Mirror your data such you can survive even if you lost a storage subsystem, or even a complete data center.

## *Create a redundant I/O configuration*

Chapter 6, “Accessing large amounts of data” on page 109, explains the main concept of accessing data through the channel subsystem. Figure 5-1 on page 96 illustrates how to create an I/O configuration to meet availability goals.

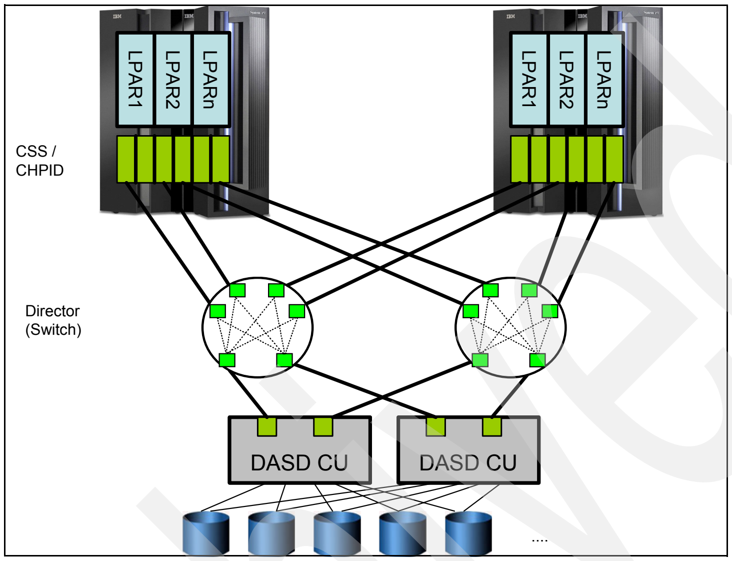


*Figure 5-1   Example redundant I/O configuration*

## *Configure the storage subsystem*

Features in today’s storage subsystems that are used by mainframe servers support this. Note, however, that although some are standard features, others are optional, and some are not available on all storage subsystem types. They are:

- Independent dual power feeds
- N+1 power supply technology/hot swappable power supplies, fans
- N+1 cooling
- Battery backup
- Non-Volatile Subsystem cache, to protect writes that have not been hardened to DASD yet
- Non disruptive maintenance
- Concurrent LIC activation
- Concurrent repair and replace actions
- RAID architecture
- Redundant microprocessors and data paths
- Concurrent upgrade support (that is, ability to add disks while subsystem is online)
- Redundant shared memory

- Spare disk drives
- Remote Copy to a second storage subsystem
  - Synchronous - Peer-to-Peer Remote Copy (PPRC)
  - Asynchronous - Extended Remote Copy (XRC)

Explaining the details of storage subsystems is beyond the scope of this document. However, an example is the current high-end IBM System Storage™ DS8000™; for details about this system, visit:

`http://www.ibm.com/servers/storage/disk/ds8000/`

### *Mirroring data*

For backup, recovery and disaster recovery scenarios, you need to mirror your data, not only by using the internal RAID functions of the storage subsystem, but also a second storage subsystem. Refer to 4.2.5, "Data backup and recovery" on page 75, for more details about this topic.

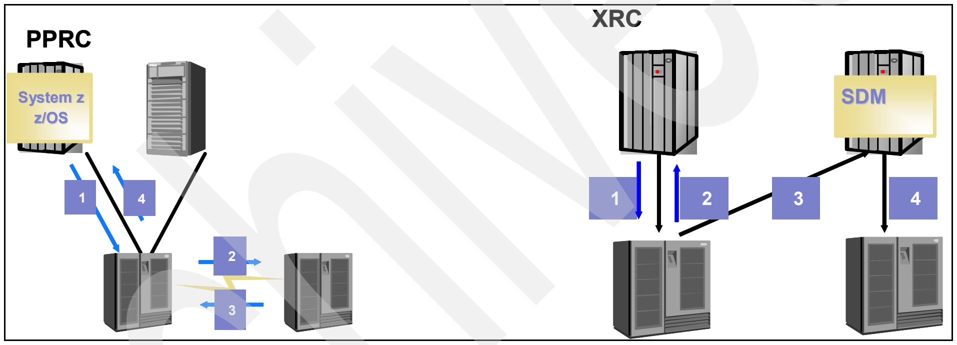


*Figure 5-2 Disk mirroring using PPRC and XRC*

Figure 5-2 illustrates disk mirroring PPRC and XRC. Peer-to-Peer Remote Copy (PPRC) is a *synchronous* copy technology. As soon as data is written to the Primary disk subsystem, the control unit forwards it to the secondary subsystem. The secondary writes it and sends an acknowledgement back to the primary. At this point, the primary lets the application know that the I/O completed. Because the control unit does not care where the I/O request came from, PPRC supports any operating system.

Extended Remote Copy (XRC) is an *asynchronous* copy technology. The System Data Mover (SDM) is a component of z/OS. It reads data from the primary control units, and coordinates applying them to the secondary volumes. **Note:** XRC only works with z/OS data.

## 5.4.3 Continuous availability of mainframes using clustering

Even with the improvement in reliability of newer mainframe systems, there remained the possibility of an outage due to hardware failure or the need to upgrade the hardware. To handle situations where hardware outages were unavoidable, and to provide for vertical expansion of applications, the sysplex (SYStems comPLEX) was developed.

“A sysplex is a collection of z/OS systems that cooperate, using certain hardware and software and microcode, to process workloads, provide higher availability, easier systems management, and improved growth potential over a conventional computer system of comparable processing power.” [3]

Parallel Sysplex technology is an enabling technology, allowing highly reliable, redundant, and robust mainframe technologies to achieve near-continuous availability. A properly configured Parallel Sysplex cluster is designed to remain available to its users and applications with minimal downtime.

Parallel Sysplex is a way of managing such a multi-system environment, providing such benefits as:

- No single points of failure
- Capacity and scaling
- Dynamic workload balancing

These benefits are detailed in the following sections.

### *No single points of failure*

In a Parallel Sysplex cluster, it is possible to construct a parallel processing environment with no single points of failure. Because all of the systems in the Parallel Sysplex can have concurrent access to all critical applications and data, the loss of a system due to either hardware or software failure does not necessitate loss of application availability to the end user.

### *Capacity and scaling*

The Parallel Sysplex environment can scale nearly linearly from 2 to 32 systems. This scaling can be a mix of any servers that support the Parallel Sysplex environment.

### *Dynamic workload balancing*

The entire Parallel Sysplex cluster can be viewed as a single logical resource to end users and business applications. And just as work can be dynamically distributed across the individual processors within a single SMP server, so too

[3] Refer to the IBM Redbook *ABCs of z/OS System Programming, Volume 5*, SG24-6985.

can work be directed to any node in a Parallel Sysplex cluster having available capacity.

## Base sysplex

A *base sysplex* is a group of LPARs or Central Processor Complexes (CPCs) that communicate via a high-speed connection, namely XCF (cross-system Coupling Facility), and are managed by a single z/OS system console. Applications are limited to running on a single LPAR, but they can communicate via XCF with applications on other systems. Workload is managed on a system-by-system basis.

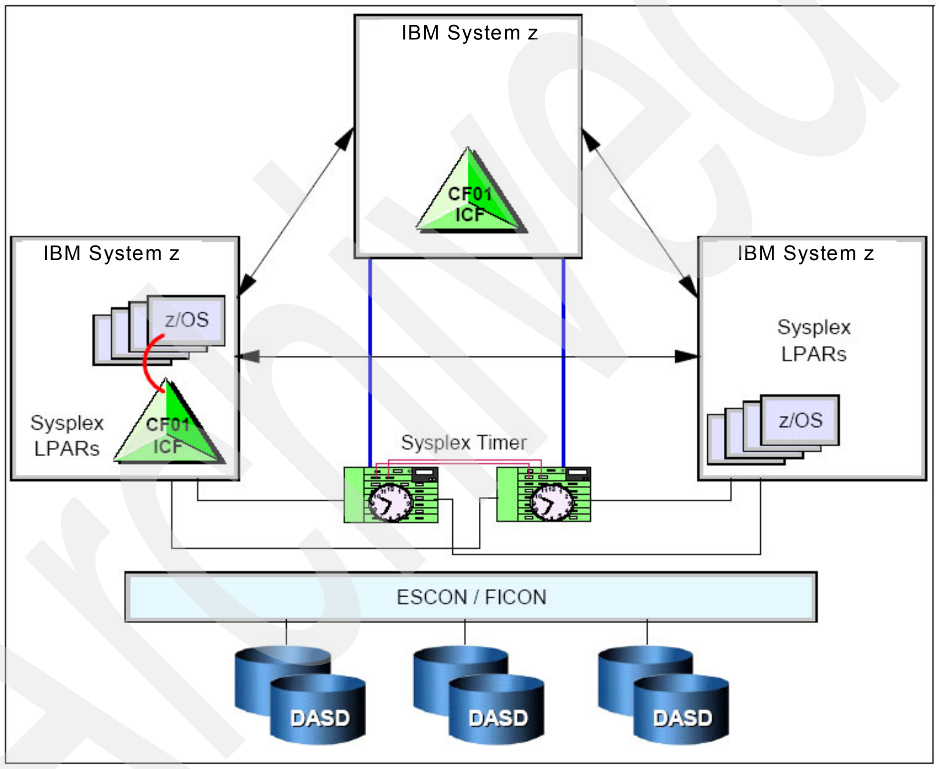


*Figure 5-3 Parallel Sysplex configuration*

## Parallel Sysplex

The evolution of Parallel Sysplex has been a quantum leap in operating system availability. A Parallel Sysplex, as illustrated in Figure 5-3, is a group of LPARs or CPCs that share a workload, share data over a Coupling Facility (CF), have a single time source (Sysplex Timer®), and are managed as one entity.

The Coupling Facility is central to the operation of Parallel Sysplex. It is an LPAR or standalone CPC with a specialized operating system, called the Coupling Facility Control Code (CFCC), which is a Licensed Internal Code (LIC). The CF is connected via high-speed links to all systems, and it acts as a shared data repository for all members of the sysplex.

The Sysplex Timer is a hardware device that is connected to all members of the sysplex. It provides a common time source for synchronizing time among the members of a sysplex.

### *How Parallel Sysplex works*

If an LPAR fails in a Parallel Sysplex, the other systems can take over the workload of the failing system using a component called WorkLoad Manager (WLM), which manages work running on all the systems in a sysplex. WLM acts as a load balancer, routing work to whichever system in the sysplex can best handle the new workload. WLM allows the systems in a sysplex to work together as though they were a single system.

Using CBU, the hardware on the running systems in a sysplex can be dynamically and nondisruptively upgraded with more processor power.

The System Logger is a facility that merges the information from logs on individual systems in the sysplex, and from individual subsystems operating on different systems in the sysplex, into coherent structures that represent a sysplex-wide view of the contents of each log.

For more information about Parallel Sysplex, refer to 3.3.3, “Parallel Sysplex” on page 48.

## 5.5 z/OS elements for availability

With IBM mainframes, the availability design-point focuses on the *application*. Why? Because applications do not simply rely on server hardware. Instead, they require an integrated environment in which hardware, firmware, operating systems and middleware work together to provide application and data availability.

Approximately one-third of the z/OS code base provides rich RAS functionality delivering reliability, availability and serviceability–often resulting in outage events being completely masked from applications—and in severe cases, resulting in graceful degradation rather than complete failure. And concurrent maintenance capabilities, supported by both the hardware as well as operating systems, help to mask planned outage events from the application as well.

## 5.5.1 z/OS components

### *Workload Manager (WLM)*

With symmetry and dynamic workload balancing, your applications can remain continuously available across changes, and your sysplex remains resilient across failures. Adding a system, changing a system, or losing a system should have little or no impact on overall availability. With symmetry and data sharing, using the Parallel Sysplex Coupling Facility, you also have enhanced database availability.

### *Automation*

Automation plays a key role in availability. Typically, automation routines are responsible for bringing up applications, and if something goes wrong, automation handles the application's restart. The need for automation is quite important in the sysplex, both for availability and other reasons.

### *Automatic Restart Manager (ARM)*

A facility of z/OS called Automatic Restart Manager (ARM) provides a fast restart and automatic capability for failed subsystems, components, and applications. Automatic Restart Manager plays an important part in the availability of key z/OS components and subsystems, which in turn affects the availability of data.

For example, when a subsystem such as CICS, IMS DB, or DB2 fails, it might be holding resources, such as locks, that prevent other applications from accessing the data they need. Automatic Restart Manager quickly restarts the failed subsystem so the subsystem can then resume processing and release the resources, making data available once again to other applications.

### *Health Checker*

The IBM Health Checker for z/OS component can be used by installations to gather information about their system environment and system parameters to help identify potential configuration problems before the problems impact availability or cause outages. Individual products, z/OS components, or Independent Software Vendors (ISV) software can provide checks that take advantage of the IBM Health Checker for z/OS framework.

### *Resource Recovery Services (RRS)*

Resource Recovery Services (RRS) assists in the two-phase commit process by acting as the *sync point manager*, coordinating the commit or backout process for various applications. The applications must still perform the commit or backout themselves, however, as RRS only acts as a mediator between the applications. Refer to 4.2.4, “Resource Recovery Services (RRS)” on page 73, for more information about this topic.

### *System Modification Program Extended (SMP/E)*

System Modification Program Extended (SMP/E) is the name of the software that is used in z/OS to manage system software configuration and to add and remove system modifications like Program Temporary Fixes (PTFs). Refer to 7.3.1, “System software configuration management” on page 132, for more information about this topic.

### *Alternate CPU recovery*

If a processor receives a machine check that indicates it needs to be repaired before it is used again, the machine check handler can reassign the task it was running to an alternate processor.

### *Alternate consoles*

The definition of each system console can indicate an alternate console. In the event of an recurring error on a console, multiple console support (MCS) will automatically reroute messages from the failing console to its alternate.

## 5.5.2 Error recording

If you do not know you have a problem, then how can you fix it? z/OS provides robust facilities for detecting and reporting on errors.

### System log (SYSLOG)

The system log (SYSLOG) provides a repository for all messages generated by the operating system and software running on the operating system. Messages produced by z/OS generally contain a great deal of information and can be used by the operations staff to monitor the system, and by automation software to respond to problems without the need for human intervention.

### LOGREC

Whenever there is a system-level abend (ABnormal END of program), a record is written to the LOGREC data set with environmental information concerning the abend, such as registers and the program status word (PSW) at the time of the abend. Symptom information such as the abending load module and the name of the CSECT (program section) is also included.

When hardware errors occur, most hardware writes symptom records to LOGREC containing the details of the error. Often the information contained in LOGREC records is enough to allow the systems programmer to open a problem record with the software or hardware vendor.

### Traces

#### *System trace*

For every interrupt that happens in the operating system, for every I/O event, and for certain program branch instructions, the operating system writes an entry in the system trace. The system trace provides a useful overview of all the events that have happened on the system, and can be invaluable in detecting program loops. The system trace can be started and stopped, but it generally runs all the time.

#### *Generalized Trace Facility (GTF)*

GTF traces the same type of information as the system trace, but can provide much more detail. In combination with Serviceability Level Indication Processing (SLIP), it can be invaluable for debugging system-level problems.

#### *Component trace*

A more recent development is the component trace. Most operating system components now write trace records to their own trace data set. These component-specific records can be used for debugging or performance purposes. They offer detailed information to help vendor support personnel resolve problems.

## 5.5.3 Recovery

Because availability is so important in a large-scale business environment, a large portion of the operating system code pertains to recovery. When the operating system is running thousands of different tasks for hundreds of users, it is not feasible to restart the operating system in order to fix a problem with one user's tasks or with one component of the system.

For this reason, z/OS and its predecessors have placed the highest priority on keeping the system running by incorporating very robust recovery routines into the fabric of the operating system. Facilities such as RTM, ESTAE, and FRR provide the highest level of system availability possible, as described here:

- Recovery Termination Manager (RTM)

  RTM monitors all terminating tasks and passes control to the appropriate recovery routines. RTM determines if the terminating program has an associated recovery routine, and whether to pass control to that routine.

- Extended Specify Task Abnormal Exit (ESTAE)

  An ESTAE is a recovery routine that provides recovery for programs running enabled, unlocked, and in task mode. Programmers can code ESTAE routines to provide cleanup and retry for tasks that have abended.

Typically, ESTAE routines free system resources and provide problem determination information for debugging purposes.

- Functional Recovery Routine (FRR)

  A Functional Recovery Routine performs many of the same functions as ESTAEs. It is written for a system-level program that is running disabled, locked, or in service request block (SRB) mode.

- Associated Recovery Routine (ARR)

  An Associated Recovery Routine is a more structured alternative to an ESTAE and FRR.

## 5.6 Disaster recovery (DR)

Disaster recovery (DR) is a discipline that has existed in the mainframe arena for many years. In addition to the existing ability to recover from a disaster, many businesses now look for a greater level of availability, covering a wider range of events and scenarios. This larger requirement is also called *IT Resilience*.

A set of tiers exists for disaster recovery readiness, as listed in Table 5-2 on page 105. The tiers range from the least expensive to the most expensive. They start at Tier 0 (No Disaster Recovery plan), where all data is lost and recovery is not possible. At the upper position is Tier 7, which provides total IT business recovery through the management of processors, systems, and storage resources across multiple sites.

The only known Tier 7 system is the IBM mainframe Geographically Dispersed Parallel Sysplex™ (GDPS®) system. GDPS manages not just the physical resources, but also the application environment and the consistency of the data, providing full data integrity (across volumes, subsystems, operating system platforms, and sites), while providing the ability to perform a normal restart in the event of a site switch. Thus, it helps to keep the duration of the recovery window to a minimum.[4]

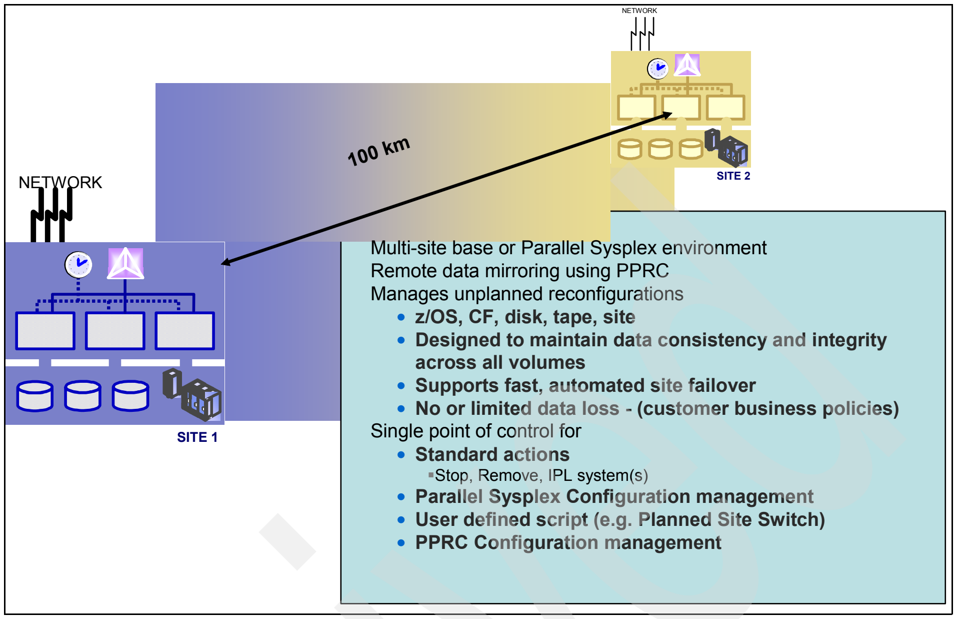
[4] For more information, refer to the IBM Redbook *GDPS Family - An Introduction to Concepts and Capabilities,* SG24-6374, and to the following site:
`http://www.ibm.com/servers/eserver/zseries/gdps`

*Table 5-2 Disaster recovery tiers*

| **Tier** | **Description** | **Data loss (hours)** | **Recovery time (hours)** |
|---|---|---|---|
| 0 | No DR plan | All | n/a |
| 1 | PTAM (Pickup Truck Access Method) | 24-48 | > 48 |
| 2 | PTAM and hot site | 24-48 | 24-48 |
| 3 | Electronic vaulting | < 24 | < 24 |
| 4 | Active second site | Minutes to hours | <24 (<2) |
| 5 | Second site 2-phase commit | Seconds | <2 |
| 6 | Zero data loss | None/seconds | <2 |
| 7 | GDPS | None/seconds | 1-2 |

## Examples for disaster recovery

The following examples present the general concept of disaster recovery using the high-end technology of IBM mainframe, Parallel Sysplex, Mirroring Data using PPRC and the automate GDPS procedures.



Multi-site base or Parallel Sysplex environment
Remote data mirroring using PPRC
Manages unplanned reconfigurations

- **z/OS, CF, disk, tape, site**
- **Designed to maintain data consistency and integrity across all volumes**
- **Supports fast, automated site failover**
- **No or limited data loss - (customer business policies)**

Single point of control for

- **Standard actions**
  - Stop, Remove, IPL system(s)
- **Parallel Sysplex Configuration management**
- **User defined script (e.g. Planned Site Switch)**
- **PPRC Configuration management**

*Figure 5-4 Example for a multi-site datacenter*

Figure 5-4 illustrates a high-level view of GDPS/PPRC topology. Based on the concept of “Mirroring data” on page 97, the physical topology of a GDPS/PPRC consists of a Parallel Sysplex cluster spread across two sites (known as site 1 and site 2) with one or more z/OS systems at each site, separated by up to 62 miles or 100 kilometers (km).

The multi-site sysplex must be configured with redundant hardware (for example, a Coupling Facility and a Sysplex Timer in each site), and the cross-site connections must also be redundant. All critical data is mirrored from the primary site (site 1, in Figure 5-4) to the secondary site (site 2).

GDPS manages unplanned reconfigurations as well as planned. It supports automated site failover and manages planned actions required on a day-to-day basis with standard actions.

With a multi-site Parallel Sysplex, this provides a continuous availability (CA) and disaster recovery (DR) solution. In addition, GDPS provides a set of panels for standard actions, and you can customize your own scripts as well.

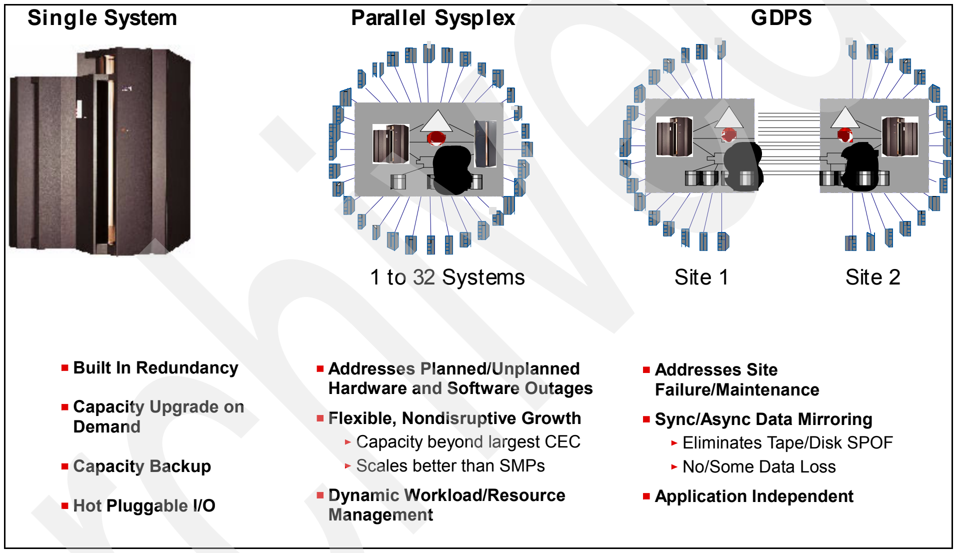
Refer to the following site for more information about GDPS:

```
http://www.ibm.com/servers/eserver/zseries/gdps
```

## 5.7 Summary

Being “available” is the state of an application being accessible to the end user. The ideal for availability is 100% of the time. Businesses can no longer tolerate downtime from software and hardware errors, or even from system upgrades.

There is no way to totally avoid failures in any single component, but there are ways to lessen the impact of these failures. Redundancy, error detection and recovery, as well as continuous monitoring of critical resources, are some of the ways to achieve high availability. To handle situations where hardware outages are unavoidable, and to provide for vertical expansion of applications, the sysplex (SYStems comPLEX) was developed.



- Built In Redundancy
- Capacity Upgrade on Demand
- Capacity Backup
- Hot Pluggable I/O

- Addresses Planned/Unplanned Hardware and Software Outages
- Flexible, Nondisruptive Growth
  - Capacity beyond largest CEC
  - Scales better than SMPs
- Dynamic Workload/Resource Management

- Addresses Site Failure/Maintenance
- Sync/Async Data Mirroring
  - Eliminates Tape/Disk SPOF
  - No/Some Data Loss
- Application Independent

*Figure 5-5 Summary - achieving high availability*

z/OS provides robust facilities for detecting and reporting on errors. To complement the hardware availability, a large portion of the operating system code pertains to recovery. z/OS includes facilities such as RTM, ESTAE, FRR, RRS, and GDPS to provide the highest level of system availability possible.

| Key terms in this chapter | | | | |
|---|---|---|---|---|
| ARM | Data sharing | HA | Recover | Sysplex Timer |
| Automate | Disaster | LPAR | SMP/E | System log |
| Availability | Disk mirroring | MTBF | SPOF | Trace |
| CA | GDPS | N+1 | Sysplex | |

## 5.8 Questions for review

To help test your understanding of the material in this chapter, complete the following review questions:

1. Define availability.
2. Give examples of redundancy built into modern computer hardware.
3. How do multiprocessors aid in system availability?
4. What are the main differences between a base sysplex and a Parallel Sysplex?
5. What error detection facilities are available on the mainframe?
6. What is the purpose of RTM?
7. What is the difference between an ESTAE and a FRR?

6

# Accessing large amounts of data

**Objective:**

The objective of this chapter is to discuss the characteristics of data storage and management in a large-scale computing environment.

After completing this chapter, you should be able to describe the concepts of:

- Channel subsystems
- Control Units
- DASD
- RAID
- Multiple allegiance/PAV
- Random access to data
- Databases
- Data sharing
- DFSMS

## 6.1 Introduction

The core function of a large-scale computing environment is to process corporate data. In order for this function to be performed in an efficient manner, access to the data is the most important consideration. This chapter discusses the design of disk input/output (I/O) devices and how they are accessed from an IBM System z server.

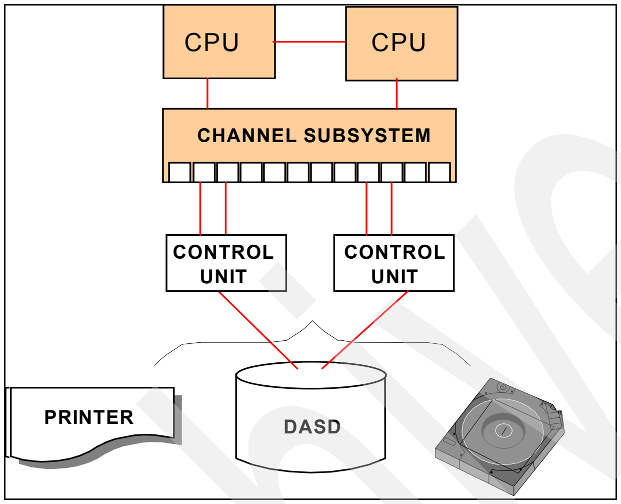


*Figure 6-1 I/O connectivity*

For any data processing, the program acting on it must reside in the processor's main storage. As shown in Figure 6-1, I/O operations result in the transfer of information between main storage and an I/O device. Data is normally stored separately from the program on an external storage device such as Direct Access Storage Device (DASD), tape or printer.

Between main storage and the devices (where data resides or is intended to be), there are other hardware components. These components, such as the channel subsystem, potentially ESCON/FICON® directors (switches), and control units and channel paths (ESCON® or FICON), are needed to connect the external storage device to the processor.

## 6.2 Channel subsystem

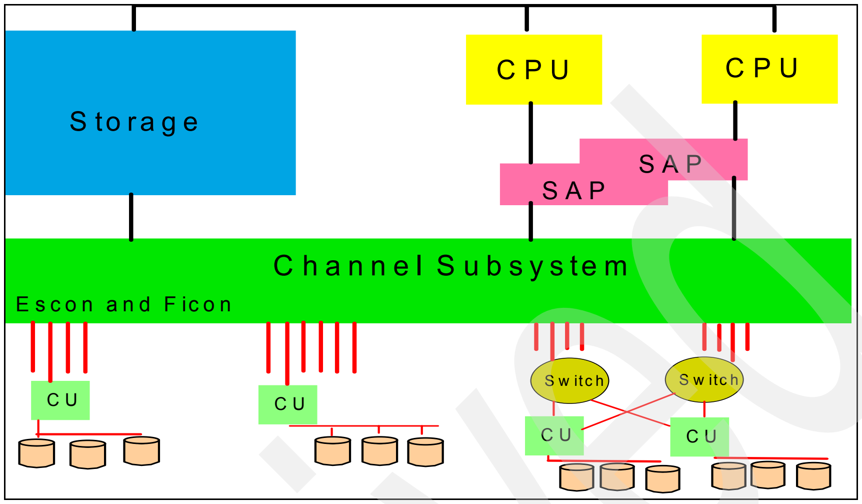


*Figure 6-2 Channel subsystem*

It is a common hardware technique to increase performance by using specialized microprocessors, such as for graphics in a personal computer. An IBM System z server has the potential for thousands of I/O operations a second. To manage them, one or more dedicated microprocessors, known as Channel Subsystems (CSSs), are used. At the time of writing the largest IBM System z could have 1024 channels spread across four CSSs.

The CSS provides the server communications to external devices using channel connections. The channels allow transfer of data between main storage and I/O devices or other servers under the control of a “special” channel program.

A CSS accepts requests, generated by the programs executing on a CP, to start I/O operations to a specific device address. Once the request is accepted, the CP can continue running programs. So the speciality processor System Assist Processors (SAPs) provide the internal I/O assistance processing between the CPU and the channel subsystem, as shown schematically in Figure 6-2.

A SAP is a processing unit (PU) that runs the channel Subsystem Licensed Internal Code to control I/O operations. All SAPs perform I/O operations for all logical partitions and all attached I/Os in an IBM System z.

## 6.3 Control units

The channel design requires devices to be connected to a *control unit*.

The function from a control unit is to execute channel commands. The channel protocol consists of channel command words (CCWs), which represent actions that the device has to perform. An operation can consist of more than one CCW, referred to as a *chain*. The control unit is responsible for making sure that an operation is complete before requesting the next CCW in the chain from the channel.

For example, adding data to the end of an existing file on tape requires the following operations:

1. Position the tape at the end of the existing data
2. Write the new data
3. Indicate that the writing of the new data has ended
4. Rewind the tape

These actions are carried out without the intervention of the CPU or the operating system. The inclusion of a control unit in the architecture allowed the designers to create functionality that would have been beyond the power of the early mainframe processors. In fact, this illustrates an early use of *dedicated microprocessors*.

The early design of disk devices allowed a device to be connected to only one control unit. For performance and resilience reasons, later designs allowed a device to be connected to more than one control unit.

## 6.4 DASD CKD architecture

A direct access storage device (DASD) is a term derived from the early mainframe environments referring to disks and drums. Early disks were manufactured in a similar way to the modern PC disks with recording surfaces and read/write (R/W) heads.

The R/W heads were mounted on a single assembly and therefore had to move together, which led to the concept of tracks and cylinders, as explained here:

- A *track* is the surface area that *one* head can read from or write to while stationary.
- A *cylinder* is the area that *all* heads in the assembly can read from or write to while stationary.

The movement of the heads, being a mechanical action, was time-consuming and was best avoided. Therefore, writing the data to an entire cylinder was the most effective thing to do. When the cylinder was full, the head assembly was moved to the next cylinder.

A personal computer disk uses a *fixed block architecture* (FBA) in which the data is written in a standard way and interpreted by software when it is read from the disk. The IBM System z uses a *count key and data* (CKD) architecture, which allows for a degree of processing by the control unit.

The term *record*, somewhat confusingly, is used differently in software and hardware contexts—although both see it as a discrete piece of information.

A *software record* is a minimum piece of information. However, for more efficient I/O operations, records may be combined to form a block for writing to disk. From a hardware point of view, the block is a record. The CKD format on a disk track is shown in Figure 6-3.

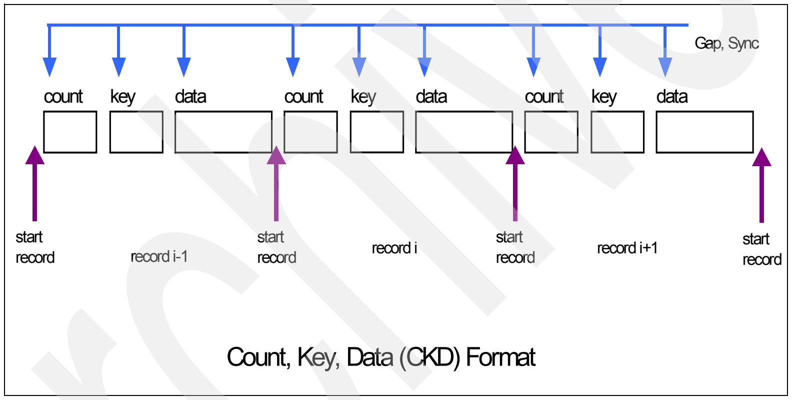


*Figure 6-3 Count key and data architecture (CKD)*

The count field and the data are always present, but the key field is only present in some circumstances. The count field identifies the position of the record on the disk and includes the cylinder and track number. It also contains the length of the key, if present, and the length of the data.

The key field can be used to identify which records a block contains. There are specialized CCWs that search the key fields. When a match is found, they read the associated data. This reduced the need to read all records to find a specific record, before relational databases came into common use.

## 6.5 Mapping for access to devices

This document does not cover the internal operation of the z/OS platform. However, it simplifies the following topics to briefly mention three elements, referred to as *control blocks*. These are the Unit Control Block (UCB), the Unit Control Word (UCW) and the Control Unit Control Block (CUCB), as explained in more detail here:

| | |
|---|---|
| **UCB** | The UCB is used by z/OS to represent every device that may be used for an I/O operation. It contains state information and queues of requests to access the device. |
| **UCW** | The UCW is equivalent to the UCB in representing a device, but it is used by the hardware. It contains state indicators and points to one or more CUCBs. |
| **CUCB** | As mentioned, a device may be connected to several control units, which in turn may be connected to several channels. A CUCB is present to represent each control unit defined to the system. It contains the pointers to the channels it is connected to. |

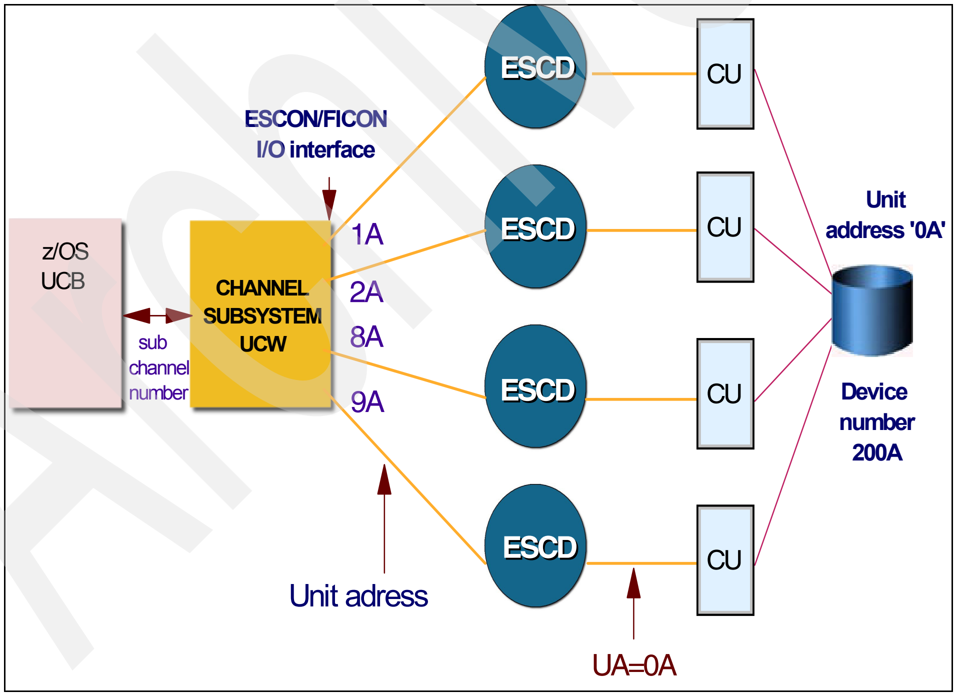


*Figure 6-4 Mapping for access to devices*

As illustrated in Figure 6-4, the CSS uses the UCW to find the CUCBs, and decides which is the best channel/control unit pair to use for the operation. After this decision is made, the relevant channel microprocessor is signalled to start the I/O. When the I/O operation is complete, the channel notifies the CSS, which in turn notifies a CP.

## 6.6 DASD subsystem

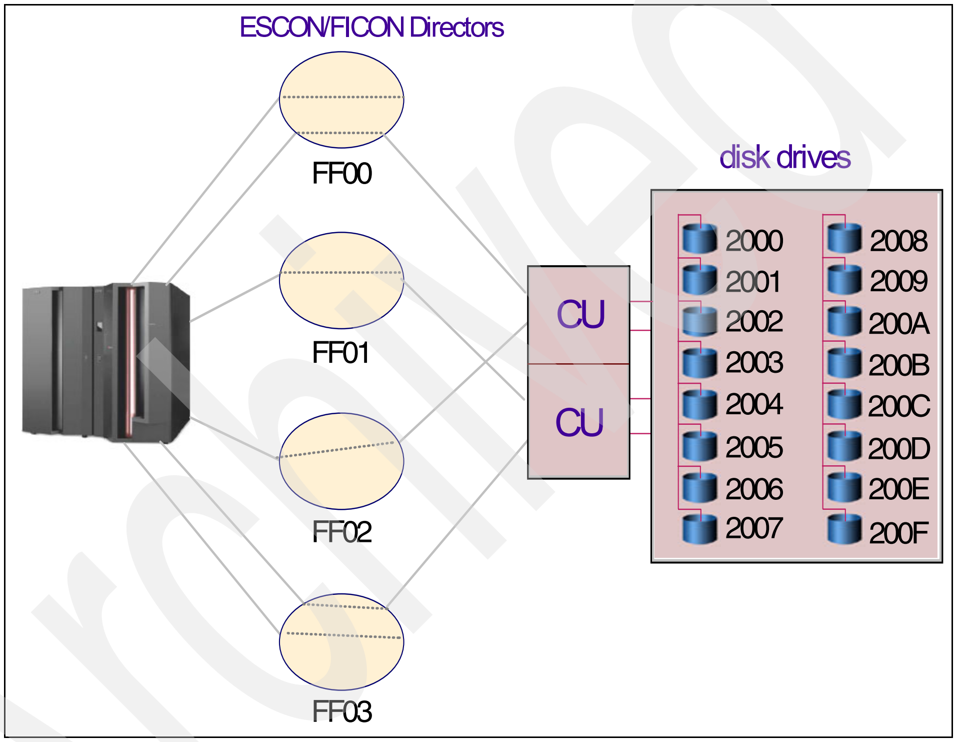


*Figure 6-5 DASD subsystem*

The combination of a control unit and a number of disk drives is referred to as a DASD subsystem, shown in Figure 6-5. The control unit allows the use of devices that may not have been designed for a channel architecture.

For example, a disk could be designed to be attached to a SCSI interface and the control unit will therefore convert to and from the channel architecture. The control unit also provides a caching facility, which allows data to be stored prior to being written and prestaged for sequential reading.

A control unit may be connected to several channels, and a device may be connected to several control units. This capability contributes to operational availability and resilience; that is, the I/O operation can have alternate paths to reach the device. Additionally, an ESCON or FICON director can be connected in between the control unit and the IBM System z, thereby increasing system stability and scalability.

Each device has a unique address that is used by z/OS to start an I/O operation, no matter how many channels and control units the device can be accessed through.

This original DASD design makes the surface or R/W heads a single point of failure. Moreover, only one read or write operation can take place at any time, which causes performance bottlenecks when multiple LPARs are sharing data. The solution to the single point of failure was the development of Redundant Array of Independent Disks (RAID). This technology also allowed the “single operation at a time” bottleneck to be addressed, as discussed in the following sections.

## 6.7 Redundant Array of Independent Disks (RAID)

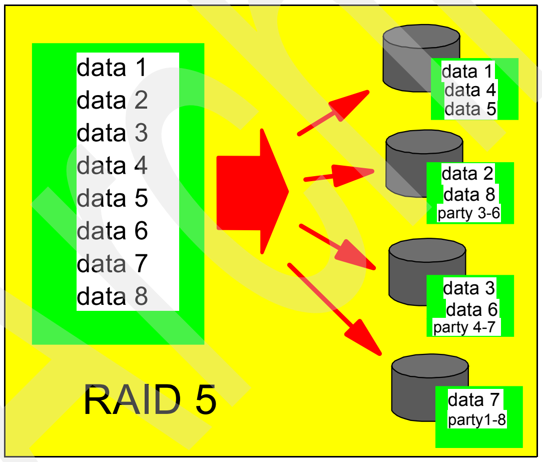


*Figure 6-6 Example RAID-5*

Redundant Array of Independent (earlier, “Inexpensive”) Disks (RAID) is a direct access storage architecture where data is recorded across multiple physical disks with parity separately recorded, so that no loss of access to data results from the loss of any one disk in the array. In the event of a failure of one of the disks, it can be replaced and the data rebuilt from the remaining disks. From the point of view of the server the data resides on a single physical disk, referred to in z/OS as a *volume*.

There is a variety of common and insignificant RAID-levels. RAID-5 has a high I/O rate and a medium data rate. In Figure 6-6 you see that RAID-5 (just like RAID-10) is used by the IBM DS8000 controller in the majority of configurations.[1]

To help performance, the control unit is equipped with a large storage cache of non-volatile storage, which is used for both read and write operations. A write operation is seen by the server to be completed when the data is in the cache (*not* when it is written to the internal disks). For read operations, the data is assembled in the cache and kept there in case of future reference, providing that the cache capacity is sufficient.

## 6.8 Reducing the number of logical volumes

The disks internal to a RAID subsystem are industry-standard FBA format, and therefore the control unit has to emulate CKD. Because the CKD architecture is emulated by the RAID subsystem, a logical volume could in theory be of any size—al though in practice the DFSMS software limited the sizes to the DASD being emulated (refer to 6.14, “Data placement and management” on page 122, and 4.2.2, “Data Facility Storage Management Subsystem (DFSMS)” on page 70 for more information about this topic).

With the steadily increasing amount of data to be stored and processed, the number of logical volumes to manage became a problem. This problem was addressed by DFSMS, allowing substantially larger logical volumes to be defined and therefore reducing the total number of volumes.

However, larger volumes meant more data per volume, and therefore more I/O requests were made to each volume. Access to volumes in z/OS was originally designed with the assumption that only one operation is permitted at a time. The UCB representing the logical volume only permits one active I/O operation; any others have to be queued, which delays processing.

The RAID design means that the operation perceived by the server as being on a single volume is, in reality, either taking place in cache or across several internal disks. Because the physical and logical operations are not directly related, a RAID-based DASD subsystem permits multiple operations to a logical volume.

The features that allow multiple access are described in the following section.

[1] Refer to the IBM Redbook *ABCs of z/OS System Programming Volume 3*, SG24-6983, for more information about this topic.

## 6.9 Multiple Allegiance/Parallel Access Volumes

Two enhancements are available with the RAID-based DASD subsystem. These are Multiple Allegiance (MA) and Parallel Access Volumes (PAV), which have similar functionality. Let's look at these enhancements in more detail.

### Multiple Allegiance (MA)

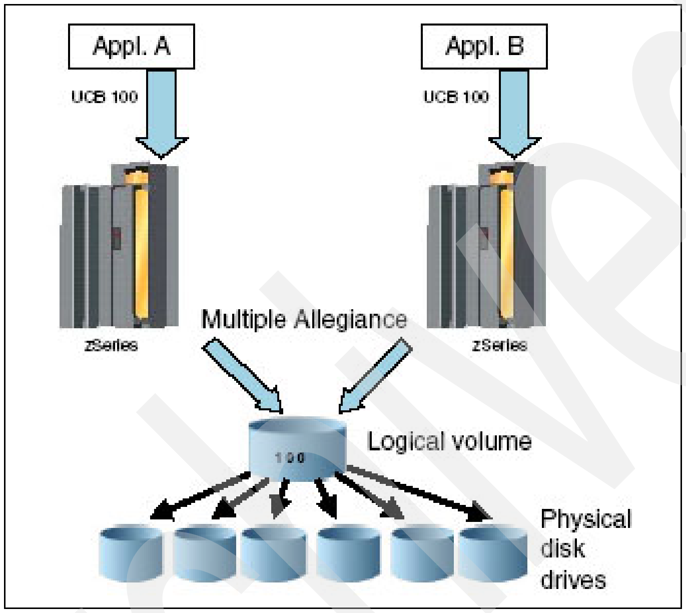


*Figure 6-7 Multiple Allegiance*

MA is a hardware feature that allows LPARs on one or more physical machines to have concurrent access to a logical volume, as shown in Figure 6-7. Each I/O operation indicates whether the intent is to read or write data, and the control unit serializes any operations that are in conflict.

## Parallel Access Volume (PAV)

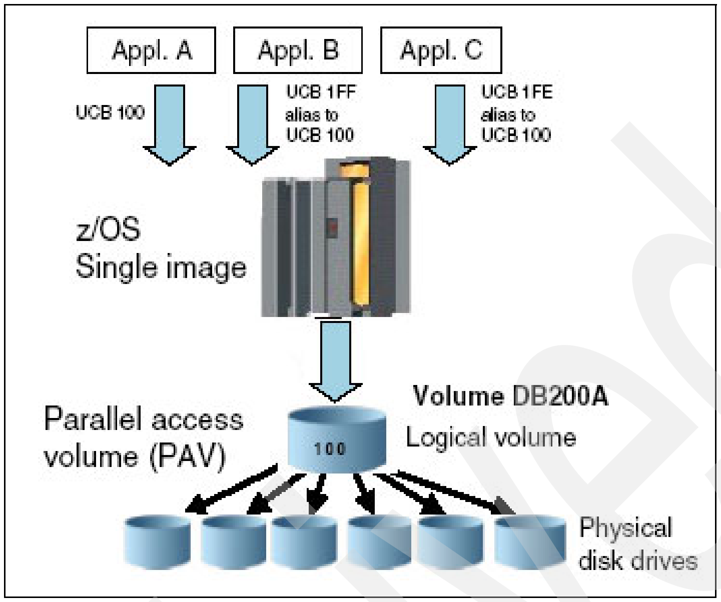


*Figure 6-8 Parallel Access Volume (PAV)*

Parallel Access Volume is a methodology for allowing multiple I/O operations to a logical volume from a single z/OS system, as shown in Figure 6-8. The function is split between the hardware and the software.

The UCB architecture still said that only one I/O was allowed to a disk, and PAV was introduced to address this limitation. The concept behind PAV is that every disk has its normal *base* UCB and also a number of *alias* UCBs, all of which connect to the logical disk. This means that it is possible to schedule concurrent I/Os to a disk: one through the base UCB, and the rest through the alias UCBs.

If z/OS detects that delays are occurring because requests are being queued on a base UCB, checks are made to see if concurrent I/O operations would reduce the queue. This may not always be the case, because the delays could be due to overloading of the control unit or channels.

If concurrent I/O *is* appropriate, however, then an alias address is selected to be used as another route to the logical volume. This association is communicated to the DASD subsystem. It maps I/O operations made to the alias addresses onto the base logical volume. If this is not sufficient to reduce the delays, then more aliases can be assigned to the base.

## 6.10 Random access to data

The RAID design provides a much greater capability for processing data than the previous generation of DASD. However, the first request to read a data record requires the component parts to be retrieved from the internal disks, and this is a relatively long operation. After the first reference is complete, the data will be in cache and therefore subject to high-speed retrieval. The problem with this, however, can be illustrated with an example from the banking industry.

The key piece of information relating to a client account is the account number, and a large bank will potentially have millions of these numbers. An increasing number of clients are using Internet access to their accounts, and this requires the reading of the account record. However, this type of access is likely to be only once or twice a day and no advantage is gained from the RAID technology because the data is not in cache. Therefore, the technique of predictive loading can be considered in such cases, as described in the following section.

### Predictive loading

If data is known to be infrequently referenced but is important for the performance of a function, then preloading it can be considered. In the case of the bank account numbers, a program could be used which reads, without processing, all of the numbers. This has the effect of loading the data into the control unit cache for fast retrieval later. A variation of this technique is to load the data into virtual storage, which gives significantly faster access.

## 6.11 Databases

Databases could be the subject of a complete course in their own right. They are mentioned here because they are one of the reasons for holding data in one place.

In a large commercial environment, data is likely to be held on relational databases. In theory, each database could be held on a separate server, but this can lead to problems in synchronizing updates.

An alternative solution is to have all of the related databases on a single large server that only manages the reading and updating requests from clients. This arrangement gives good data integrity but usually poor processor utilization.

The z/OS solution of having the clients and database manager on the same server gives high processor utilization. It also reduces the complexity of communication and therefore the exposure to failure between the client and server.

## 6.12 Data sharing

The ability to share data for both reading and updating, from multiple programs that can be running on different physical machines, is a key requirement for capacity, scalability, and availability. The problem to be solved in any environment is how to indicate an interest in a piece of data and lock it for updating, without impacting performance.

To achieve data sharing in z/OS, Parallel Sysplex is required, with structures in the Coupling Facility (CF) providing the necessary mechanisms. Parallel Sysplex is described in more detail in 5.4.3, “Continuous availability of mainframes using clustering” on page 98 and 3.3.3, “Parallel Sysplex” on page 48.

## 6.13 Data Facility Storage Management System

In addition to internal storage, external storage devices (for example, tape volumes) are still used as well. The use of tape subsystems is generally applied for sequential, archiving, or backup processing (see 4.2.5, “Data backup and recovery” on page 75).

One component of the DFSMS, the base of the storage management system (SMS) of DFSMS, is the Object Access Method (OAM). OAM uses the concepts of system-managed storage, introduced by SMS, to manage, maintain, and verify tape volumes and tape libraries within a tape storage environment.

In general, a *tape library* is a set of tape volumes and the set of tape drives where those volumes may be mounted. The relationship between tape drives and tape volumes is exclusive; a tape volume residing in a library (library-resident tape volume) can only be mounted on a tape drive contained in that library (library-resident tape drive), and a library-resident tape drive can only be used to mount a tape volume which resides in the same library.

A tape library can consist of one or more tape systems. When a volume is entered into a tape library, it is assigned to a tape storage group. A tape library can contain volumes from multiple storage groups, and a storage group can reside in up to eight libraries.

As new tape data sets are created, the installation allocates data sets to tape volumes in an SMS-managed tape library by associating one or more tape storage group names (using the SMS storage group ACS routine) with the allocation request. DFSMS ensures that only tape devices within the tape libraries associated with the tape storage groups are allocated to the request. Existing tape data sets on library-resident volumes are allocated to tape drives within the library where the volume resides.

SMS is used to define a storage hierarchy for objects and the parameters for managing those objects. OAM uses this hierarchy definition and management parameters to place user-accessible objects anywhere in the SMS storage hierarchy. The object storage hierarchy can consist of:

- Direct access storage device (DASD)
- Tape volumes associated with a tape library device (SMS-managed, library-resident tape volumes), and tape volumes outside of a library device (non-SMS-managed, shelf-resident tape volumes)
- Optical volumes inside a library device (SMS-managed, library-resident optical volumes), and optical volumes outside of a library device (SMS-managed, shelf-resident optical volumes)

Information related to tape volumes is kept in the tape configuration database (TCDB), which is an Integrated Catalog Facility user catalog that contains tape volume and tape library records. The TCDB can be used to maintain information about an IBM tape library and the volumes that reside there.

Tape drives, depending on the type and model, can use the channel subsystem, potentially ESCON/FICON Directors (switches), control unit and channel path (ESCON or FICON) to connect to the processor. [2]

## 6.14 Data placement and management

In the type of environment discussed here, there can be terabytes of data to be managed. The DASD subsystems may have different performance characteristics, such as having more cache. Regulatory requirements might determine how many copies of the data need to exist and for how long. To manage these types of requirements, automation tools are necessary.

In z/OS, the DFSMS component provides the tools to perform these actions. The name given to the tool will indicate its characteristics, which can refer to:

- Performance
- Space
- Backup

### Performance

Data can be placed on disk subsystems based on the performance requirements for that class of data. If the data has a high reference rate, then it may be placed on a disk subsystem that has plenty of available cache. The data could also be

[2] Refer to the IBM publication *z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries*, SC35-0427.

striped across multiple disk assemblies so that different parts of the data can be read concurrently. Data used for batch processing tends to be processed sequentially, and the preloading of data to the RAID control unit cache could provide the needed performance.

Online processing tends to process data in a random fashion and demands a higher level of performance. In this case the data would most likely be placed on disk subsystems with greater performance characteristics or facilities, such as Parallel Access Volumes, that allow multiple readers to the same volume concurrently.

### Space

z/OS does not allow files to be of unlimited size. A primary and secondary space scheme is used. The primary space is made available on a volume, and then a secondary space, up to a limit, is allowed if the former is used up. Rules can be established to make sure that sufficient space is available under most circumstances so that the user does not need to take any action.

### Backup

As any personal computer user is aware, disk failure or accidental damage to data can cause major disruption. In a large-scale environment, where data can be interrelated, recovery can be a lengthy process unless a strict regime of backing up data is in force.

DFSMS provides a full complement of backup and recovery solutions as previously discussed in 4.2.5 “Data Backup”. The solution could be for local recovery or disaster recovery, it could be a software, hardware or a hybrid solution and it could be at the data set, volume, application and site level.

### Migration

All media for storing data has a cost. DFSMS provides the capability to move infrequently used files from the more expensive high performance disk subsystems to a less expensive disk or tape subsystems. This data movement is called *migration*.

During migration, DFSMS can compress the data to reduce its size and can release space in those data sets that were initially over-allocated and not all the allocated space has been used. Migration is usually an automatic process that is invoked when a file has not been referenced for a set time, usually at least a few days or data can be migrated by command. When data is in its migrated state, it is not readable by an end user or application and must be recalled back to the original disk subsystem in order to be accessed.

In DFSMS, there can be multiple migration levels. These are known as migration level one (ML1) and migration level two (ML2). In most cases, data that has gone unreferenced for a period of time is migrated to ML1. If the data remains unreferenced on ML1 for a preset period of time, then migration to ML2 occurs and the data is moved to an even lower cost per byte medium such as tape.

If a particularly aggressive view is taken, then a file can go straight to ML2 status, bypassing ML1. If the data is referenced while in either ML1 or ML2 status, then it is automatically returned to the user disk volume (recalled) where it can be accessed by a user or application. Recall can invoked automatically, such as when a data set is referenced by a user or application, or it can be recalled by command.

## 6.15 Summary

Accessing large amounts of data can be done on almost any commercial computing system. In a large-scale commercial environment, access to data can be required for a variety of applications, which might involve updating of the data. Having all of the data in a single environment simplifies its management and allows high processor utilization.

However, extremely high levels of reliability are necessary in the hardware and software. When running multiple applications, restarting the system to recover one of them is a serious event. The combination of z/OS software, an IBM System z, and well-tested application software should be able to run for months and only be restarted for maintenance activities.

Tape media has a high capacity and relatively low cost for storing data, but it is only practical for data archiving or data that will be processed sequentially. Data that will be accessed for online applications, where performance is an important consideration, needs to be stored on disk.

Data transfer rates are usually expressed in terms of bits per second. The IBM System z channel data rates are expressed in bytes per second. At the time of writing, the fastest channel has a transfer rate of 4 Gbit per second.

| Key terms in this chapter | | | | |
|---|---|---|---|---|
| Channel subsystem | Control unit | DFSMS | Migration | Tape library |
| CKD | DASD | FBA | PAV | |
| Control block | Data sharing | Logical address | RAID | |

## 6.16 Questions for review

To help test your understanding of the material in this chapter, complete the following review questions:

1. What are Control units?
2. Can you explain RAID?
3. What is “random access to data”?
4. Why is data sharing needed?
5. What is the difference between MA and PAV?
6. Why don’t we place all our data on tape? It is less costly. What do we use?

## 6.17 Topics for further discussion

1. By today’s standards, what constitutes a large amount of data?

7

# Systems management

**Objective:**

After completing this chapter, you will be able to:

- ► Understand systems management disciplines
- ► Understand the different data types that /OS uses
- ► Understand how errors are handled

## 7.1 Introduction to systems management

*Systems management* is a general term that is widely used, and whose meaning is dependent on the context. In some cases systems management means the operator interface, while in others, it means provisioning capacity.

In a large-scale commercial system, systems management usually is considered to be "A collection of disciplines aimed to monitor and control a system's behavior." The disciplines usually included in systems management are:

| | |
|---|---|
| **Configuration management** | The techniques, resources, and tools used to maintain an accurate inventory of the hardware and software (system and application). |
| **Workload management** | Techniques, resources, and tools used to define, monitor, and maintain the levels of service the unit of work needs. |
| **Operations management** | Techniques, resources, and tools used to have more effective and error-free operator communication. |
| **Network management** | Techniques, resources, and tools used to monitor the network status, availability, and performance, and taking corrective action if needed. |
| **Storage management** | Techniques and tools used to manage external storage (disk and cartridge storage). |
| **Security management** | Techniques, resources, and tools used to define, monitor, and maintain the security standards. |
| **Performance management** | Techniques, resources, and tools used to define, monitor and, eventually, act upon the overall system performance. |
| **Problem management** | Techniques, resources, and tools used to open, follow, resolve, and close any problem (hardware, software or application). |
| **Change management** | Techniques, resources, and tools used to track and manage changes, and their dependencies and implications. Changes can be hardware, software, configuration, and so on. |

Some of these functions are performed by the operating system or appropriate subsystems, while others are provided in specialized tools marketed by various software companies.

Many businesses concentrate more on running their day-by-day operations than on planning for or anticipating critical situations. Thus, the implementation of these systems management disciplines is frequently driven by critical situations such as unacceptable response time, client complaints, problems that are not being solved in a timely manner, changes in software or hardware that create problems, changes in the business environment that require more computational resources, and so on.

However, most of the information about how a system is performing, which is needed for systems management, already exists. z/OS monitors, collects, logs and registers a great deal of valuable system information. Reporter systems can then order it in a way that is readable and significant to people, so that immediate decisions can be made or the appropriate planning can be done to accommodate or provide new resources.

## 7.2 System data

In a z/OS environment, the operating system is responsible for the collection and maintenance of the data, even if the data is generated by different components (the system itself, the batch manager, transaction manager, database manager, security manager, workload manager, and so on).

Data collection for accounting, reporting, and managing a system is described here:

- Accounting

  Large-scale commercial operating systems are usually run on behalf of many different users, departments, and even companies. They need an accounting system to be able to know the costs (at least in resource usage) of each piece of work. But as a prerequisite for an accounting system, there must be data collected.
- Reporting

  As on any other platform, the two main tasks of the operations staff is to be able to run the system with the best performance possible, and to be able to resolve any error that might prevent achieving the level of service stipulated in the SLA.
  - Performance

    The performance manager monitors the system's performance and keeps data about it. This data is the base for producing performance reports.

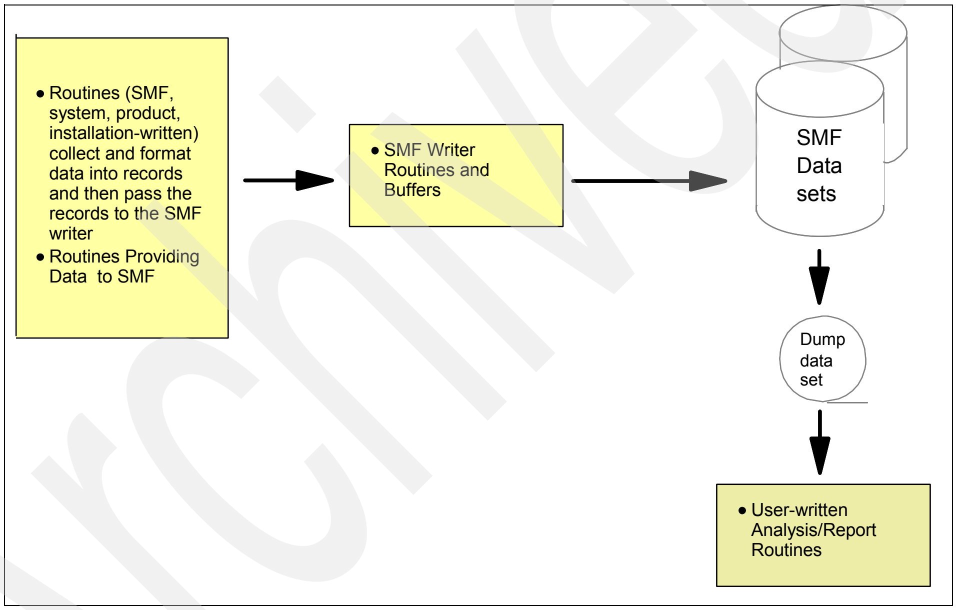
– Errors

  System and subsystems must keep an error log to help in problem determination.

The usual flow of information is that the system or subsystem gets the data and puts it in a common repository. This data is usually raw data that must be "cooked." Usually, every manager that writes data into the central repository has some software to read it and report about it. There are also tools that allow this data to be loaded into a standard data base, in order to be able to use a standard database interface for querying and reporting.

## Data collection in z/OS

The implementation of data collection in z/OS is summarized in Figure 7-1.



• Routines (SMF, system, product, installation-written) collect and format data into records and then pass the records to the SMF writer
• Routines Providing Data to SMF
• SMF Writer Routines and Buffers
SMF Data sets
Dump data set
• User-written Analysis/Report Routines

*Figure 7-1 System data repository*

The z/OS component System Management Facilities (SMF) has the function of being a repository. It uses a set of files in an alternating fashion; when one file fills, it overflows onto the second, and then the third, and so on. Usually there are three or four of these files.

This data is retained and subsequently used to provide a range of user-specified reports. In many installations, the data is copied to a daily file, then periodically the SMF data is copied to a weekly or monthly file for long-term storage and subsequent retrieval.

The operating system provides the facility to automatically switch the SMF capture onto the next data set. Automation operational tools are also employed to delete the data sets at various times, or under certain conditions.

The z/OS component System Management Facilities (SMF) collects and records system and job-related information that the installation can use in:

- Billing users
- Reporting reliability
- Analyzing the configuration
- Scheduling jobs
- Summarizing direct access volume activity
- Evaluating data set activity
- Profiling system resource use
- Maintaining system security

SMF formats the information that it gathers into system-related records (or job-related records). System-related SMF records include information about the configuration, paging activity, and workload. Job-related records include information about the CPU time, SYSOUT activity, and data set activity of each job step, job, APPC/MVS transaction program, and TSO/E session.

As illustrated in Figure 7-1 in a simplified fashion, several routines of SMF (system, program-product and installation-written) collect and format data into records and then pass the records to the SMF writer. SMF routines copy records to SMF buffers and transfer records from the SMF buffers to the SMF data sets. SMF has a dump program that copies data from SMF data sets to dump data sets for permanent storage.

Analysis and report routines, either user-written or program products, process information records. Analysis routines read the SMF data set, list the dumped SMF data set, use a sort/merge program to order the SMF-recorded information, or perform a detailed investigation of one particular SMF data item, such as job CPU time under TCBs. Report routines usually format and print the statistics and results of the analysis routines.[1]

## 7.3 Configuration management

In large-scale commercial systems, there are always a large number of hardware devices, system software items (such as compiled modules, source modules, data items and so on), and application software items to manage.

[1] More information can be found in the IBM publication *z/OS: MVS System Management Facilities (SMF)*, SA22-7630.

The Worldwide Project Management Method (WWPMM) defines configuration management as the process of defining, monitoring, and controlling the status and versions of the components of a system, which together constitute a consistent whole for a period of time.[2]

Capability Maturity Model (CMM) defines configuration management as a discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specific requirements.[3]

The first definition is more general, and the second definition is more application-oriented. Following is a brief description of how these three areas (hardware, system software, and application software) are managed in z/OS. Hardware and system software tools are integrated in the operating system, and application software configuration can be controlled with various vendor tools.

### 7.3.1 System software configuration management

System Modification Program Extended (SMP/E) is the name of the software that is used in z/OS to manage system software configuration. It is used for the installation of software products on the z/OS system, and for tracking the modifications made to those products.

Usually, it is the responsibility of the systems programmers to ensure that all software products and their modifications are properly installed on the system. Systems programmers must also ensure that all products are installed at the proper level so that all elements of the system can work together.

System software development in this environment has characteristics that have determined the development direction of the configuration management tools. These characteristics are:

- Large number of components.
- Great packing flexibility, meaning that Installation X may need Module A but not Module B, or vice versa. This flexibility has always been important in allowing installations to build customized environments.
- Backward compatibility requirement. The investment of large-scale commercial operating systems in software must be preserved, so there is a need for backward compatibility (that is, every new software version must support everything the previous version did).

[2] Standard IBM Worldwide Project Management Method (WWPMM)
[3] IBM Capability Maturity Model Integration Model

- Different software developers and vendors.
- Long-supported versions.

SMP/E supports an inventory of each single software piece and, more importantly, the relationship between the pieces. Each modification set (including one or more piece of software) is called a SYSMOD and is the actual package containing information that SMP/E needs to install and track system modifications.

There are four types of SYSMODs[4] as described here:

- Program Temporary Fix (PTF)

  These SYSMODs prevent or fix problems with an element, or introduce new elements.
- Authorized Program Analysis Report (APAR)

  These SYSMODs correct a defect in a current release of an IBM-supplied program.
- FUNCTION

  These SYSMODs introduce the elements for a product.
- User Modifications (USERMOD)

  These SYSMODs customize an element.

It is useful to understand the following terminology when dealing with SMP/E:

- Prerequisites

  The minimum maintenance level necessary to be able to safely install the modification set. This is expressed with the names of the sysmods (usually PTFs) that must be present in the system.
- Corequisites

  The modifications to the maintenance level that are dependent on the installed software. This is expressed as: If PTF abc is present, then PTF xyz must also be present.
- IF requisite

  This a modification control statement (MCS) used to specify requisites that must be installed for a given SYSMOD if a specified function is also installed.
- HOLDDATA

  This is an MCS used to indicate that a given SYSMOD contains errors or requires special processing before it can be installed. Examples include holds for errors, documentation updates, workstation downloads, IPLs, and specific

[4] Refer to the IBM publication *IBM SMP/E for z/OS User's Guide*, SA22-7773, for more information.

actions that must be taken before or after APPLY processing, ACCEPT processing, or both.

- Installation instructions and restrictions

  Hardware prerequisites, special instruction, operation needs, new or changed messages, and so on.

### 7.3.2 Hardware configuration management

An *I/O configuration* consists of the hardware resources available to the operating system, and the connections between these resources. When you define a configuration, you need to provide both physical and logical information about these resources. For example, when defining a device you provide physical information (such as its type and model), as well as logical information (such as the identifier you will assign in the configuration definition).

The resources to be managed are channels, switches, control units, and devices that are connected and can be accessed from the LPARs that are configured in the sysplex. These LPARs can reside in one or more separated IBM System z machines.

It is necessary to define an I/O configuration to the operating system (software) and the channel subsystem (hardware). The Hardware Configuration Definition (HCD) element of z/OS consolidates the hardware and software I/O configuration processes under a single interactive end-user interface. The validation checking that HCD does when data is being entered helps to eliminate errors before you attempt to use the I/O configuration.

The output of HCD is an I/O definition file (IODF) that contains I/O configuration data. An IODF is used to define multiple hardware and software configurations to the z/OS operating system.

These actions must be completed in order for a hardware configuration to become usable:

1. It has to be written in a system file.
2. It has to be split between the hardware and the software. As stated in Chapter 6, “Accessing large amounts of data” on page 109, an I/O operation involves both operating system routines as well as channel subsystem work.
3. It has to be tested to ensure that this configuration will not cause a problem when activated.
4. It has to be activated sysplex-wide, to ensure that there are no conflicts in accessing the devices.

This scheme is meant to provide security, integrity, and flexibility to a large-scale operating system environment, as explained here:

- Security

  The definition of the hardware is centralized. Only users that need to use it must be authorized, and the activation action can be protected.

- Integrity

  Only LPARs that need to be able to share the same device are given access to it. LPARs in the same production sysplex usually see the same devices in order to be able to access the same data—but there may be other LPARs on the same channel subsystem that must be allowed to see only their own devices. This is the case of a testing or development system. There also may be LPARS running different operating systems, for example, the case of many LPARs running Linux.

- Flexibility

  The granularity is at a single-device level. Besides that, LPARs are only affected by the modifications on the I/O devices they see.

### 7.3.3 Application configuration management

Application configuration management is a discipline that is common to any platform. On a IBM System z platform, these tools must be able to manage Java™ (WebSphere Application Server) as well as COBOL applications. The IBM tool for application configuration management is know as Software Configuration and Library Manager (SCLM).

## 7.4 Workload management

The Workload Manager (WLM), whose functions are described in Chapter 3, “Scalability” on page 37, is responsible for the distribution of resources among the units of work that are executing over a given period of time.

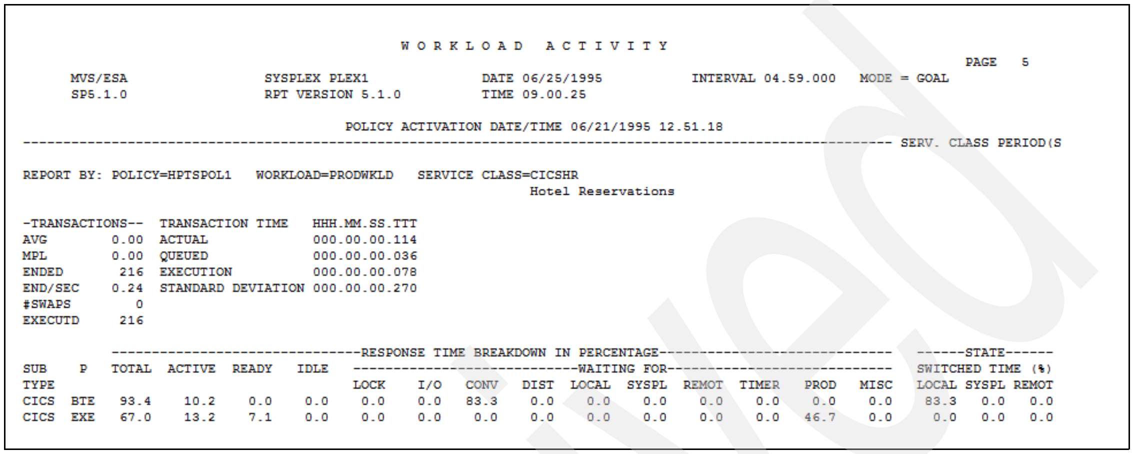
As previously mentioned, the units of work are classified in service classes. There are report classes, too, which are intended to group work not on a performance basis, but on a reporting basis.

WLM writes summarized data over a period of time (usually half an hour) for both classes. WLM retains the following data, which can subsequently be requested in a report:

- CPU time used
- Memory used and pages/sec from this class
- I/Os done and I/O rate

- Transaction rate (if it is a transactional class)
- How the goal was achieved in the last period

Figure 7-2 is an example of an RMF 5.1.0 Workload Activity Report. It shows a CICS DBCTL workload with CICS/ESA® 4.1 and IMS/ESA® 4.1.



```
                                              W O R K L O A D   A C T I V I T Y
                                                                                                                    PAGE   5
      MVS/ESA                 SYSPLEX PLEX1              DATE 06/25/1995          INTERVAL 04.59.000   MODE = GOAL
      SP5.1.0                 RPT VERSION 5.1.0          TIME 09.00.25

                                        POLICY ACTIVATION DATE/TIME 06/21/1995 12.51.18
----------------------------------------------------------------------------------------------------------- SERV. CLASS PERIOD(S

REPORT BY: POLICY=HPTSPOL1   WORKLOAD=PRODWKLD   SERVICE CLASS=CICSHR
                                                               Hotel Reservations

-TRANSACTIONS--  TRANSACTION TIME    HHH.MM.SS.TTT
AVG        0.00  ACTUAL              000.00.00.114
MPL        0.00  QUEUED              000.00.00.036
ENDED       216  EXECUTION           000.00.00.078
END/SEC    0.24  STANDARD DEVIATION  000.00.00.270
#SWAPS        0
EXECUTD     216

           ------------------------------RESPONSE TIME BREAKDOWN IN PERCENTAGE----------------------------   ------STATE------
SUB    P   TOTAL  ACTIVE  READY   IDLE   ----------------------------WAITING FOR---------------------------   SWITCHED TIME (%)
TYPE                                     LOCK    I/O   CONV   DIST  LOCAL  SYSPL  REMOT  TIMER   PROD   MISC   LOCAL SYSPL REMOT
CICS  BTE   93.4    10.2    0.0    0.0    0.0    0.0   83.3    0.0    0.0    0.0    0.0    0.0    0.0    0.0    83.3   0.0   0.0
CICS  EXE   67.0    13.2    7.1    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   46.7    0.0     0.0   0.0   0.0
```

*Figure 7-2 A WLM RMF report example*

Note that waiting on a another product is the largest component of the CICS execution phase of processing. In this example, the other product is IMS/ESA 4.1.

## 7.5 Operations management

Operations in a large-scale commercial operating system is crucial for performance and availability. The term *operations* usually refers to two related activities:

- Batch scheduling

  This is the operator's control over the *flow* of batch jobs, which usually must be run in a dependent way. This dependency tree is what is managed by the batch scheduling tools.

- Console operations

  This is the operator's control over the various *tasks* that are running in the system. These functions include starting, stopping, cancelling, and managing priorities of every task active in the system.

Now let's take a closer look at these activities.

## 7.5.1 Batch scheduling

A major task facing large-scale commercial operating systems is dealing with a huge number of batch jobs. Having thousands of batch jobs a day is quite normal for a medium to large sysplex system, and it usually represents between 50% and 70% of the total work. The heavy contribution of batch jobs to the overall system workload is what drove the development of tools to automate the scheduling and execution control of batch jobs.

The z/OS implementation of such tools range from a relatively simple solution based on direct relations and deadlines implemented by the Job Entry Subsystem (JES2 or JES3), to more sophisticated tools that specialized vendors (including IBM) have developed.

A batch scheduling tool must have most of these capabilities:

- The capacity to run multiplatform with a single administrative console. Centralized operations are useful to maintain single scheduling for that site's work, and to decrease administration activity.
- Obtain information about job status and statistics.
- Rules-based and event-based processing. Any tool must be able to:
  - Schedule batch on a specific time basis (for example, run batch job BBB at 3 a.m.).
  - Schedule on an if-then-else basis (if batch job ABC runs successfully, then run batch job XXX, else run batch job ZZZ).
  - Schedule on an event-driven basis (when file A is created, execute batch job EEE).
- Provide automated failure alerting for the operations staff.
- Perform automatic recovery and restart for failed jobs.

## 7.5.2 Console operations

A *console* is one of the communication channels between the operating system and subsystems and the outside world. Consoles in z/OS must be present for the system to run. Today's technology allows the console to be:

- A keyboard/display unit, attached to a controller or communications link. This can be monochrome ("green screen") or color (four colors or seven colors).
- A simulated keyboard/display unit attached to a controller or communications link; in particular, a TN3270 client on a TCP/IP network.
- A programmable console interface:

  - Subsystem consoles, which are console interfaces used by automated operations tools and components of the operating system.
  - EMCS console interfaces, which are tools available to any program if the user has suitable authorization from SAF.
- A hardware management console (HMC), which is a special function that is optimized for control of the entire processor complex. In emergencies, the HMC is available for use as a system console.

Large-scale commercial operating systems must offer enough information through messages to adequately inform the operator about what is happening in the system. As the messages appear on the screen and then scroll off, they are retained in a logfile called the system log (SYSLOG).

Let’s look at some of the characteristics of messages:

- Every component has its own identifier, which is associated with the first three letters of the message. For instance, IGCxxxxxT is from the data manager, and IRRxxxxT comes from the security manager.
- The last letter of the message (T) indicates whether it is:
  - An information message (I)
  - A decision message (D) - An operator has to choose an action
  - An action message (A) - An operator is instructed to do something
  - A warning error message (W)
  - A critical error message (E)
- The components of the system that control consoles are known as multiple console support (MCS) and device independent operator console support (DIDOCS). The definition of each console controls the messages that go to it, by type of message and by the system originating the message.
- Every system component sends messages to the consoles. Therefore, in medium to large installations, the consoles can be flooded with them. This flood can be reduced by using a filter. What is filtered is up to the user. Generally all error messages are kept, but the filter screens informational messages. However, messages are not lost because they are all recorded in the system log.
- The consoles can be defined as sysplex-wide, so that the sysplex has a *single point of control*. Sysplex-wide means that messages from all systems appear on the same console, and the scope of the messages can be controlled. A system operator can issue commands that apply to only one system, commands that apply to some of the systems, or commands that apply to all systems in the sysplex.

**Note**: Most decision messages, some error messages, and even some information message must be followed by an operator command.

There are automation tools to automate console operations. System Automation (SA) for z/OS is an IBM product that can be tailored to capture all messages requiring an action, as well as to automate repetitive tasks. SA captures the event, and if coded to do so, performs an action automatically.

An example of this automation may be to attempt to restart a network node that has become inactive. It may be coded to do this an additional three times (or three passes) within the next three minutes. If the node does not become active, SA may be coded to generate an alert to Operations for immediate action.

Another example is the automated restart of an application subsystem, upon certain abend conditions. These captured actions can also be delivered immediately to a system or batch operator for follow-up. SA can also interface with a scheduling package to initiate an automated procedure within the scheduling package. For example, a job may be triggered to run when a certain event occurs.

## 7.6 Network management

z/OS participates like any other node in the network and complies to the standards in this area (SNMP communication, for instance). On such networks, the z/OS network management subsystem usually acts as the focal point.

## 7.7 Storage management

In z/OS, the software that manages the external storage devices is known as DFSMS. It is discussed in 4.2.2, “Data Facility Storage Management Subsystem (DFSMS)” on page 70.

## 7.8 Security management

Security and integrity are discussed in Chapter 4, “Integrity and security” on page 65.

In addition, z/OS can use a firewall as one of its address spaces, joining the general security scheme. An example of such a scheme is shown in Figure 7-3.

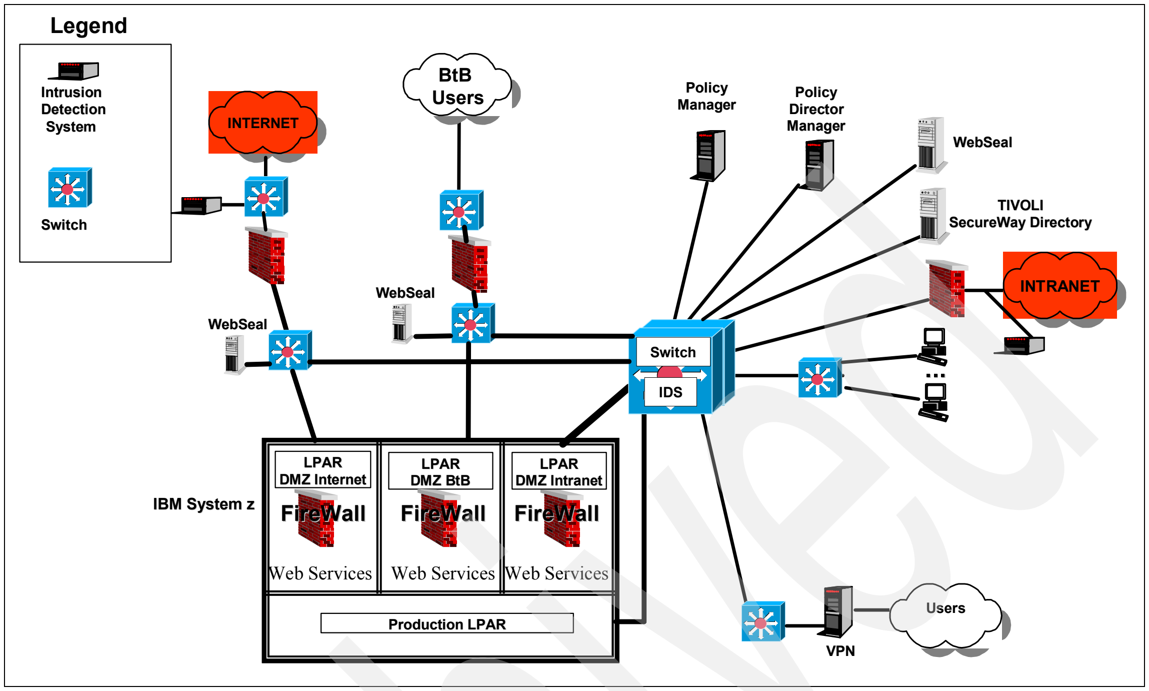


*Figure 7-3 A sample enterprise security layout*

## 7.9 Performance management

Performance management is a key discipline in the systems management area. It includes measuring, analyzing, reporting, and tuning the performance of IT resources. Performance management falls into two categories:

- Real-time monitoring, alerting, problem identification, and problem resolution
- Bench marking, modeling, rerunning problem scenarios, and trending performance metrics feeding capacity planning

IT resources include:

- Hardware
- Software
- Applications

Capacity planning is another discipline very tightly coupled to performance management. It can be considered as a long-term performance management activity, or as another discipline where business requirements are modelled according to hardware/software requirements.

Different methods can be used for capacity planning. The method and tool selected will depend on the time, cost, skill, and level of detail and accuracy that are available.

LPAR Capacity Estimator (LPAR/CE), CP2000 Quick Sizer, as well as Processor Capacity Reference (PCR) and zProcessor Capacity Reference (both based on Large Systems Performance Reference (LSPR) data) are examples of tools that can help you estimate various configurations and workloads using a spreadsheet-like approach.

These tools are based on previous IBM comparisons among models of mainframes, resulting in an Internal Throughput Ratio (ITR) for each machine and an Internal Throughput Rate Ratio (ITRR) when comparing to a base model in terms of performance capacity.[5]

Performance management objectives usually include the following:

- Optimize response time and throughput of IT resources
- Take corrective actions to alerts and problem requests

To realize these objectives, the performance management discipline includes some or all of the following activities:

- Define and maintain performance alerts
- Define performance report formats
- Balance and tune IT resources and workloads
- Analyze change requests from a performance management perspective
- Collect performance statistics
- Respond to alerts (real-time by operations management - trended by performance management)
- Manage the resolution of performance problems
- Automate performance tuning and alerts

## 7.9.1 z/OS implementation

There are many tools in the system performance arena that deal with performance management issues. In general, they monitor performance in one or more of the “flavors” that include the monitoring flavor used by the Resource Measurement Facility (RMF) or Omegamon (IBM tools) tools.

[5] Refer to the IBM Redbook *ABCs of z/OS System Programming Volume 11*, SG24-6327, for more information about this topic.

## Resource Measurement Facility

Resource Measurement Facility (RMF) monitors performance data in three flavors:

- Batch monitoring

  There is a collector address space that collects performance data and writes it onto a general repository, to be processed afterwards. The collection is done on a seconds basis, while the writing in the repository is done on a minutes basis (usually from 5 to 30 minutes). There is always a compromise between the granularity (and therefore, the accurancy) of the measurements and the overhead required to collect more detailed data.

  This kind of monitoring is often used to gain a historical view of the behavior of the performance indicators. The kind of data that can be found there is as follows:

  – CPU Statistics - PR/SM, CF(general and detail), cryptographic hardware activity

  – Storage - Paging activity, page data set activity, virtual storage activity

  – Workload activity - Service given to the classes, transactions ended, goal attainment

  – I/O Activity - Channel path activity, device activity, switch analysis, disk statistics

  – Other - Enqueue activity

- Online monitoring

  RMF can take a snapshot of system behavior at a point in time. RMF is usually started when needed to allow the Operations staff to determine how a system (or systems, if in a sysplex) is behaving at a given moment.

  RMF deals with the same kind of data as batch monitoring, but in addition to general values, online monitoring allows for more granularity and values can be seen in a unit of work basis. In this area, there are many tools from different vendors that allow this kind of performance monitoring.

- Delay monitoring

  “Performance” deals with how many resources are allocated to a given user, as well as with how long the user is delayed. This delay is analyzed by sampling every second and averaging over a minute or two to see how the users are delayed waiting to use the resources. It is an online tool, so a resource bottleneck can be detected and corrected at that point in time.

## 7.10 Problem management

Problem management is a discipline that can be viewed from various angles:

- How problems are solved
- How problems are reported
- How problems are tracked

Chapter 4, "Integrity and security" on page 65, describes the first case: how the operating system tries to solve the problems it finds. Problem tracking is a general issue in IT, and large-systems commercial operating systems usually integrate on the general problem tracking circuit. There are many tools that can be used to track problems from the moment they appear to the moment they are solved.

In the following section we explain what aspect of problem management is system-managed. With system management, a problem symptom is sent to the system console for an immediate action. Problems related to hardware or system software are also recorded in an error logging file (LOGREC), as described in "LOGREC" on page 102.

### 7.10.1 Trend reporting

z/OS has a utility program known as EREP that is used to print LOGREC records. LOGREC data is usually gathered on a daily or weekly basis. This data can then be analyzed to help in problem determination, or to highlight trends in order to apply preventive maintenance (hardware errors).

### 7.10.2 Operator console

A system operator console is another place where error information is displayed. The consoles are usually sysplex-wide in scope, and problems originating from any sysplex members come to a single console, thus allowing a single point of control. A hardware management console (HMC) is available for use in case a system enters a disabled wait state and loses contact with the system console.

## 7.11 Change management

Change management is a discipline that is meant for the whole infrastructure. z/OS helps to manage changes applied to software through SMP/E, asking for prerequisites and logging changes. A similar task is performed for hardware-related changes by the hardware configuration definition (HCD) product.

## 7.12 Summary

In this chapter, we discussed nine different disciplines of systems management: configuration management, workload management, operations management, network management, storage management, security management, performance management, problem management, and change management. Keep in mind that a given computing environment could have fewer disciplines, or even more systems management disciplines, depending on its requirements. For example, security management has permanent technical and business requirements in many environments.

It is important to realize that ultimately, the systems management disciplines described here implicitly relate to service level management; that is, to meeting the committed service level. For example, efforts to improve system performance in order to meet SLA commitments may involve isolating and resolving problems in the network, storage, workload balancing, device or channel configuration or operations.

We have seen how z/OS provides and uses tools to obtain and process the information required for better system management. The generated system information is collected and formatted in a way that is understandable to humans so it can be analyzed and used to improve overall system behavior.

Effective systems management allows you to monitor the system components, be aware of circumstances that may lead to unexpected results, and apply appropriate correct action, solutions, and improvements.

| **Key terms in this chapter** | | | | |
|---|---|---|---|---|
| APAR | HCD | Problem management | SMP/E | Workload management |
| Change management | Network management | PTF | Storage management | |
| Configuration management | Operations management | RMF | System data | |
| FUNCTION | Performance management | Security management | USERMOD | |

# Autonomic computing

**Objective:**

After completing this chapter, you will be able to:

- ► Understand autonomic computing capabilities
- ► Understand the autonomic computing framework
- ► Know the autonomic capabilities of large-scale commercial systems
- ► Know the autonomic computing tool

## 8.1 Introduction

Technology has permeated so many aspects of our lives today that it is almost expected. Because technology appears in even the simplest tasks, it seems routine. However, along with the evolution of a larger technical infrastructure comes the task of maintaining and managing it.

The very technology that makes our life easier—it automates a set of business processes and integrates applications and data across an enterprise—should be "smart" enough to manage itself. And this principle is the basis upon which autonomic computing is built.

The term "autonomic" comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations automatically without any external help. We walk up a flight of stairs and our heart rate increases. When it is hot, we perspire. When it is cold, we shiver. We do not tell ourselves to do these things; they just happen.

### 8.1.1 Autonomic computing principles

Similarly, the way to handle the complexity problem is to create computer systems and software that can respond to changes in the digital environment, so the systems can adapt, heal, and protect themselves. Only then will the need be reduced for constant human maintenance, fixing, and debugging of computer systems.

In a self-managing autonomic environment, system components—from hardware (such as storage units, desktop computers, and servers) to software (such as operating systems, middleware, and business applications)—can include embedded control loop functionality. Although these control loops consist of the same fundamental parts, their functions can be divided into four broad embedded control loop categories.

Autonomic computing systems must follow four principles (have four categories of attributes). The systems must be:

**Self-configuring** This means that they can adapt dynamically to changing environments. Self-configuring components adapt dynamically to changes in the environment, using policies provided by the IT professional. Such changes could include the deployment of new components, the removal of existing ones, or dramatic changes in the system characteristics. Dynamic adaptation helps ensure continuous strength and productivity of the IT infrastructure, resulting in business growth and flexibility.

| | |
|---|---|
| **Self-healing** | This means that they can discover, diagnose and react to disruptions. Self-healing components can detect system malfunctions and initiate policy-based corrective action without disrupting the IT environment. Corrective action could involve a product altering its own state or effecting changes in other components in the environment. The IT system as a whole becomes more resilient because day-to-day operations are less likely to fail. |
| **Self-optimizing** | This means that they can monitor and tune resources automatically. Self-optimizing components can tune themselves to meet end-user or business needs. The tuning actions could mean reallocating resources—such as in response to dynamically changing workloads—to improve overall utilization, or ensuring that particular business transactions can be completed in a timely fashion.<br><br>Self-optimization helps provide a high standard of service for both the system's end users and a business's clients. Without self-optimizing functions, there is no easy way to divert excess server capacity to lower priority work when an application does not fully use its assigned computing resources. In such cases, clients must buy and maintain a separate infrastructure for each application to meet that application's most demanding computing needs. |
| **Self-protecting** | This means they can anticipate, detect, identify, and protect against threats from anywhere. Self-protecting components can detect hostile behaviors as they occur and take corrective actions to make themselves less vulnerable. The hostile behaviors can include unauthorized access and use, virus infection and proliferation, and denial-of-service attacks. Self-protecting capabilities allow businesses to consistently enforce security and privacy policies. |

By enabling computers to take care of themselves, autonomic computing is expected to have many benefits for business systems, such as reduced operating costs, lower failure rates, and better security.

This definition is based on a paper titled *Autonomic Computing: an architectural blueprint for autonomic computing* that can be found on an Autonomic Computing site at:

http://www.ibm.com/autonomic

## 8.1.2 Autonomic computing concepts

The architectural concepts presented here define a common approach and terminology for describing self-managing autonomic computing systems. The autonomic computing architecture concepts provide a mechanism for discussing, comparing, and contrasting the approaches that different vendors use to deliver self-managing capabilities in an IT system.

### Autonomic computing system

An *autonomic computing* system can be defined as a computing system that senses its operating environment, models its behavior in that environment, and takes action to change the environment or its behavior.[1] It has, to various extents depending on the implementation, properties of self-configuration, self-healing, self-optimization and self-protection. The base loop is the so-called control loop shown in Figure 8-1.

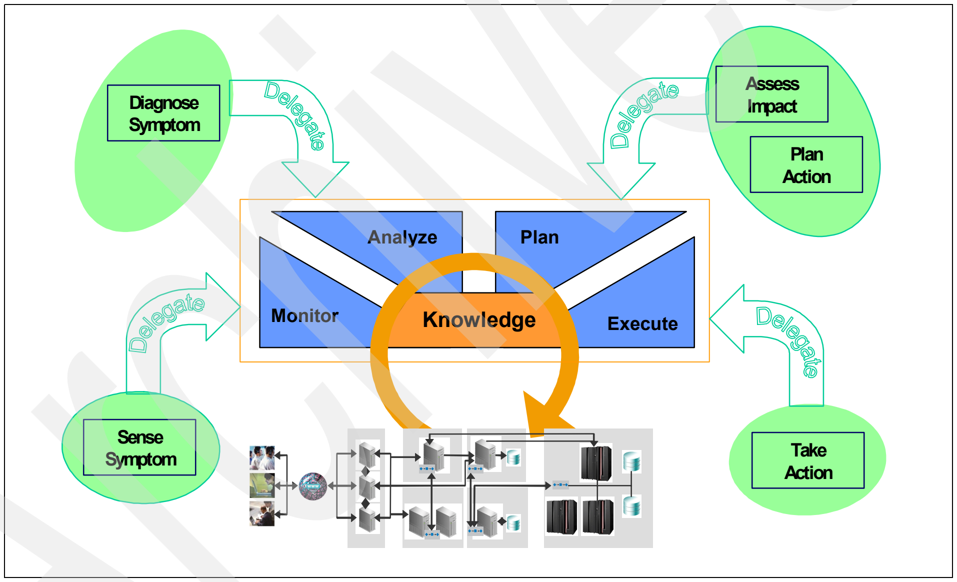


*Figure 8-1 Standard control loop*

The autonomic computing manager and control loop is a never-ending process. It is derived from control theory and manufacturing, which is quite advanced, yet simple.

[1] See the IBM Redbook *A Practical Guide to the IBM Autonomic Computing Toolkit*, SG24-6635, for more information about this topic.

### *Autonomic manager*

The autonomic manager is a component that implements the control loop going through four parts that share knowledge: monitor, analyze, plan, and execute[2], as described here.

1. The *monitor* part provides the mechanisms that collect, aggregate, filter, manage, and report details collected from an element.
2. The *analyze* part provides the mechanisms that correlate and model complex situations.
3. The *plan* part provides the mechanisms that structure the action needed to achieve goals and objectives. The planning mechanism uses policy information to guide its work.
4. The *execute* part provides the mechanisms that control the execution of a plan.

Each iteration captures relevant information to allow for smarter decisions the next time around and continually keep looking at it—that is really the central paradigm of autonomic computing.

These control loops are spread over system components. The autonomic computing framework in Figure 8-2 illustrates the various layers and components of a “model” autonomic system.

[2] Refer to IBM Redbook *Problem Determination Using Self-Managing Autonomic Technology*, SG24-6665, for more information about this topic.

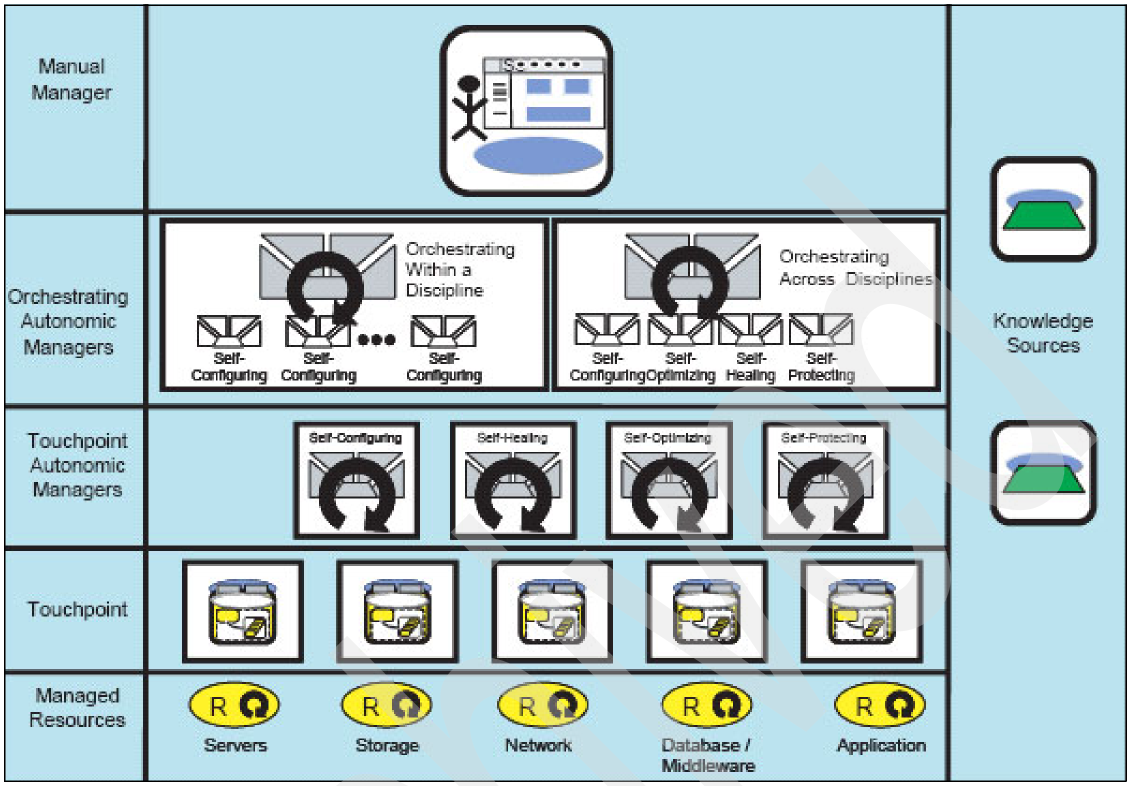


*Figure 8-2 Autonomic computing framework*

These parts are connected using patterns that allow the components to collaborate using standard mechanisms such as Web services.

### *Managed resources*

A *managed resource* is a hardware or software component that can be managed. It could be a server, storage unit, database, application server, service, application, or some other entity. A managed resource might contain its own embedded self-management control loop. These embedded control loops are one way to offer self-managing autonomic capability. The control loop might be deeply embedded in a resource so that it is not visible through the manageability interface.

The managed resource can be a single resource, or a collection of resources. The managed resource is controlled through its sensors and effectors, as described here:

- The *sensors* provide mechanisms to collect information about the state and state transitions of an element.

- The *effectors* are mechanisms that change the state (configuration) of an element. In other words, the effectors are a collection of set commands or application programming interfaces (APIs) that change the configuration of the managed resource in some way.

As shown in Figure 8-2, the combination of sensors and effectors form the manageability interface, referred to as the *touchpoint*, which is available to an autonomic manager.

### Touchpoints

A touchpoint is an autonomic computing system building block that implements *sensor* and *effector* behavior for one or more of a managed resource's manageability mechanisms.

It also provides a standard manageability interface. Deployed managed resources are accessed and controlled through these manageability interfaces. Manageability interfaces employ mechanisms such as log files, events, commands, application programming interfaces (APIs), and configuration files. These mechanisms are meant to gather details about the resource (sensor) or to change the behavior of the managed resources (effector).

### Touchpoint autonomic managers

Touchpoint autonomic managers work directly with the managed resources through their touchpoints. They can perform various self-management tasks, so they embody different intelligent control loops. Examples of such control loops, using the four self-managing categories introduced in 8.1.1, "Autonomic computing principles" on page 146, include:

- Performing a self-configuring task such as installing software when it detects that some prerequisite software is missing
- Performing a self-healing task such as correcting a configured path so installed software can be correctly located
- Performing a self-optimizing task such as adjusting the workload control settings when it observes an increase or decrease in capacity
- Performing a self-protecting task such as taking resources offline if it detects an intrusion attempt

Most autonomic managers use policies (goals or objectives) to govern the behavior of intelligent control loops. Touchpoint autonomic managers use these policies to determine what actions should be taken for the managed resources that they manage.

A touchpoint autonomic manager can manage one or more managed resources directly, using the managed resource's touchpoint or touchpoints. These managers are meant to provide control on a superior level. There are four typical scopes of resources controlled by a single touchpoint autonomic manager:

- A *single resource scope* is the base.

  It implements a control loop that accesses and controls a single managed resource, such as a network router, a server, a storage device, an application, a middleware platform, or a personal computer.

- A *homogeneous group scope* aggregates resources of the same type.

  An example of a homogeneous group is a pool of servers that an autonomic manager can dynamically optimize to meet certain performance and availability thresholds.

- A *heterogeneous group scope* organizes resources of different types.

  An example of a heterogeneous group is a combination of heterogeneous devices and servers, such as databases, Web servers, and storage subsystems that work together to achieve common performance and availability targets. Another example of this kind of group would be WLM in z/OS. It deals with a combination of heterogeneous resources (CPU, storage resources, and I/O resources).

- A *business system scope* organizes a collection of heterogeneous resources so an autonomic manager can apply its intelligent control loop to the service that is delivered to the business.

  Some examples are a client care system or an electronic auction system. The business system scope requires autonomic managers that can comprehend the optimal state of business processes—based on policies, schedules, and service levels—and drive the consequences of process optimization back down to the resource groups (both homogeneous and heterogeneous) and even to individual resources.

These resource scopes define a set of decision-making contexts that are used to classify the purpose and role of a control loop within the autonomic computing architecture.

### *Orchestrating autonomic managers*

A single touchpoint autonomic manager acting in isolation can achieve autonomic behavior only for the resources that it manages. The self-managing autonomic capabilities delivered by touchpoint autonomic managers need to be coordinated to deliver system-wide autonomic computing behavior. Orchestrating autonomic managers provide this coordination function.

There are two common configurations:

- Orchestrating within a discipline

An orchestrating autonomic manager coordinates multiple touchpoint autonomic managers of the same type (one of self-configuring, self-healing, self-optimizing, or self-protecting).

- Orchestrating across disciplines

  An orchestrating autonomic manager coordinates touchpoint autonomic managers that are a mixture of self-configuring, self-healing, self-optimizing, and self-protecting.

An example of an orchestrating autonomic manager is a workload manager. An autonomic management system for workload might include self-optimizing touchpoint autonomic managers for particular resources, as well as orchestrating autonomic managers that manage pools of resources. A touchpoint autonomic manager can optimize the utilization of a particular resource based on application priorities. Orchestrating autonomic managers can optimize resource utilization across a pool of resources, based on transaction measurements and policies. The philosophy behind workload management is one of policy-based, goal-oriented management.

Tuning servers individually using only touchpoint autonomic managers cannot ensure the overall performance of applications that span a mix of platforms. Systems that appear to be working well on their own may not, in fact, be contributing to optimal system-wide end-to-end processing.

### Manual managers

A *manual manager* provides a common system management interface for the IT professional using an integrated solutions console. Self-managing autonomic systems can use common console technology to create a consistent human-facing interface for the autonomic managers of IT infrastructure components.

The primary goal of a common console is to provide a single platform that can host all the administrative console functions in server, software, and storage products to allow users to manage solutions rather than managing individual components or products. Administrative console functions range from setup and configuration to solution run-time monitoring and control.

The common console architecture is based on standards (such as standard Java APIs and extensions including JSR168, JSR127, and others), so that it can be extended to offer new management functions or to enable the development of new components for products in an autonomic system.

## 8.2 z/OS implementation of autonomic computing

z/OS systems have many autonomic features built in. There are many control loops embedded on the base components (hardware and software), and there are autonomic manager implementations as well (Workload Manager or Storage Manager).

Most second-level orchestration autonomic managers (AC managers) are provided by tools developed by various vendors. One of these vendors is the IBM Tivoli® family of products.

## 8.3 Self-healing

The characteristics of z/OS that can be considered as self-healing address many different aspects of the system's health.

- Some of them are related to the capability of the hardware to detect a failure and solve it. The solution can be:
  - Repair - as in the *Error Correction Code* (ECC) scheme.

    ECC is an arrangement for correction of failures of one order and for detection of failures of a higher order. Commonly, ECC allows correction of single-bit failures and detection of double-bit failures; it is used in every place some data is read (Level 1 Cache has Parity controls, Level 2 cache has ECC).
  - IBM System z Hardware has spare units on many components. This is called N+1 design.

    This design provides fault-tolerant capabilities, allowing:
    - Use of spare PUs

      As stated before, IBM System z has at least two spare PUs on each system. There can be more if the book is not used at its full capacity. Hardware microcode is able to detect a permanent CPU failure and substitute for it. z/OS uses logical CPs, so the use of the spare CP, or even the loss of one of them, is transparent.
    - Use of spare memory chips

      There are spare chips (for example, eight chips on IBM zSeries z990) on every memory card. The memory cards are continuously being checked to clean correctable errors and detect uncorrectable ones. When an error cannot be corrected, the chip is made unavailable and a spare is used instead. There are also error thresholds that trigger a "call home" procedure.

On machines having more than one book sharing their Level 2 memory, another source of error can be the access path. To prevent this, there are two parallel paths.

- I/O has multipath access

  As seen in Chapter 6, “Accessing large amounts of data” on page 109, multiple I/O paths help prevent an I/O operation from failing on the IBM System z side. Even though I/O operations can be recovered, a threshold is established. When the threshold is reached, a “call home” occurs.

For more information about hardware availability, refer to Chapter 5.4, “Redundancy” on page 91.

- There are also design points that allow these schemes to work, for instance the Chipkill memory design.
- Electronic Service Agent™ - this offers “call home” support. Every IBM System z machine can, and usually does, have a phone connection with the nearest plant. This link is used to report errors that will lead, or will probably lead, to a replacement of a failing component.
- Concurrent updates
  - Most maintenance can be done while the systems are up and running. Nevertheless, updates are usually done during periods of low activity using System Modification Program/E, which manages the system software configuration and checks for pre-requisites and co-requisites.
  - As stated in 3.3.4, “Provisioning” on page 54, the way hardware is packed allows users to have more capacity (CPs and memory) than needed. When the resources are to be brought online, a concurrent microcode update is all that is needed. In the case where a new book must be added, the time required will be longer, but in most cases the update can still be concurrent.
- System software

  There are self-healing capacities in system software, as well. These capacities fall into three main areas, dealing with data replication, automation tools, and virtualization techniques:

  - Data replication
    - Coupling Facility (CF) structure duplexing

      As described, the Coupling Facility is the heart of data-sharing technology. Data in CF can be replicated to allow recovery in case of a failure. To recover, there is usually a procedure to follow that should be automated.

    - Synchronous copy

The system could make a synchronous copy of data before returning control to the user or application program. This ensures that there are two identical copies of the data prior to the user or application moving to the next step in processing. The synchronous copy of the data could be local or might even be at a remote site.

  – Automation engines

    - System automation for z/OS
    - Geographically Dispersed Parallel Sysplex (see 5.6, "Disaster recovery (DR)" on page 104)

  – Virtualization

    - Dynamic Virtual IP takeover and takeback
    - Dynamic disk balancing

      DFSMS spreads data over the disks on the same storage group depending on the load. This is a sort of virtualization. If a disk fails, systems eliminate it from the pool, switching new allocations to the rest.

- Health checking

  The system may be able to check the status of its own health by having automated health checks. It may provide such checks as ensuring that critical control data has valid backups or that system parameters are not set such that they contradict each other. These checks can be performed automatically at a particular point in processing, or each time the system is started.

## 8.4 Self-configuring

Software tools, disciplines, and automatic with or without manual intervention can be applied to help for a self-configuring process:

- msys for setup simplifies the management tasks for z/OS software setup.
- z/OS Wizards are Web-based dialogs that assist in z/OS customization.
- Capacity upgrade CPU provides instant access to additional processors or servers, memory, I/O.
- Customer-initiated upgrade.
- Automatic hardware detection/configuration.
- Automatic communication configuration.

## 8.5 Self-protecting

Self-protecting features include:

- LPAR
- Intrusion detection IDS, PKI
- Hardware cryptographic (coprocessors, accelerators and CP assist) adapters
- Digital certificates providing identity authentication
- SSL and TLS (manages Internet transmission security), Kerberos (authenticates requests for service in a network), VPN, encryption
- Tivoli Policy Director
- LDAP (aids in the location of network resources)

HiperSockets™ Intrusion Detection Services (IDS) enables the detection of attacks and the application of defensive mechanisms on the z/OS server.

Public Key Infrastructure (PKI) is embedded in z/OS. PKI consists of a certificate authority (CA) that provides digital credentials to participants and a public key cryptographic system that uses these digital credentials to help ensure overall message integrity, data privacy, signature verification, and user authentication.

In IBM System z systems, there are hardware and software components that are certified by security standards like FIPS 140-2 Level 4 (IBM CEX2 Cryptographic coprocessor and accelerator), Common Criteria EAL4 (IBM Tivoli Directory Server, IBM WebSphere Application Server, between others), ICSA Labs (the crypto algorithms in S/390® Virtual Private Network (VPN) support), and ZKA, the security standard for the financial industries in Germany (IBM 4758 PCI Cryptographic Coprocessor).[3]

## 8.6 Self-optimizing

Self-optimizing features include:

- Intelligent Resource Director (IRD) extensions (non-z/OS partitions - CPU (Linux), I/O, server-wide) - allows dynamic resource allocation on IBM System z servers
- Dynamic LPAR, WLM LPAR extensions for Linux
- Parallel Sysplex Extensions - Sysplex Distributor, CP Coupling
- BIND9 DNS-DNS BIND Version 9.0 on z/OS

[3] For more details on security standards for IBM products, refer to the following site: http://www.ibm.com/security/standards/st_evaluations.shtml

- z/OS Workload Manager (CPU, memory, I/O, TCP/IP QOS, Web request management, and batch initiator balancing)

The unique workload and self-management capabilities provided by z/OS Workload Manager (WLM) and the IBM System z IRD allow z/OS to handle unpredictable workloads and meet response goals through effective use of CPU and I/O resources with minimal human intervention for setup and operation, making it the most advanced self-managing system. IBM System z workload management also allows workload balancing of non-z/OS partitions, in particular Linux images.

## 8.7 Summary

Ideally, we want computers to behave somewhat like a person in reacting to and correcting malfunctions and taking preventive and anticipative actions in avoid undesirable results. This behavior is known as autonomic computing.

In this chapter we described the principles from which autonomic computing is derived, namely self-configuring, self-healing, self-optimizing and self-protecting. These principles are based on the implementation of the concept of a control loop. The control loop is a never-ending process that is always going through four stages or steps (monitoring, analyzing, planning, and execution). These stages constitute the knowledge base needed to be continuously aware of what is happening, and to take the corrective actions needed to have the process perform optimally.

The principles and concepts of autonomic computing have led to the creation of the Autonomic Computing framework. Within this framework we put the management and control elements needed to observe and act directly on the computing resources, or in the combination of the computing resources to enhance their performance. Thus, the framework goes from the resource itself (which can be a server, storage, network, database or application), up to human intervention to manage the resource.

We saw what is in the middle of the framework: the touchpoints while collecting information; the touchpoint autonomic managers going through the control loop according to the principles; and the orchestrating part, which applies the same principles for a particular discipline and to the whole system.

Finally, we mentioned the z/OS products that support these principles, concepts, and the framework. This hardware and software support of autonomic computing has contributed to making IBM System z an extremely reliable mainframe.

| Key terms in this chapter | | | | |
|---|---|---|---|---|
| Autonomic | Control loop | Monitoring | Self-healing | Sensor |
| Autonomic computing control loop | ECC | msys for Setup | Self-optimizing | SMP/E |
| Autonomic computing framework | Effector | Self-configuring | Self-protecting | Touchpoint |

A

# Architecture summary

**Objective:**

This appendix consolidates the information given about interrupt handling and the role of the program status word (PSW).

## Interrupt processing

An *interrupt* is an event that alters the sequence in which the processor executes instructions. An interrupt might be *planned* (specifically requested by the currently running program) or *unplanned* (caused by an event that might or might not be related to the currently running program). z/OS uses six types of interrupts, as explained here:

- **Supervisor call** or **SVC** interrupts - occur when the program issues an SVC instruction. An SVC is a request for a particular system service.
- **I/O interrupts** - occur when the channel subsystem signals a change of status. For example, an I/O operation completes, an error occurs, or an I/O device (such as a printer) becomes ready.
- **External interrupts** - indicate any of several events, such as a time interval expires or the CP receives a signal from another CP.
- **Restart interrupts** - occur when the operator selects the restart function at the console or when a restart signal processor (SIGP) instruction is received from another CP.
- **Program interrupts** - occur when an event occurs that requires additional processing. This could be when the program attempts to perform an invalid operation and might have to be ended. It might be a legal operation that references a virtual address that is not in real storage and z/OS intervention is required.
- **Machine check interrupts** - caused by machine malfunctions.

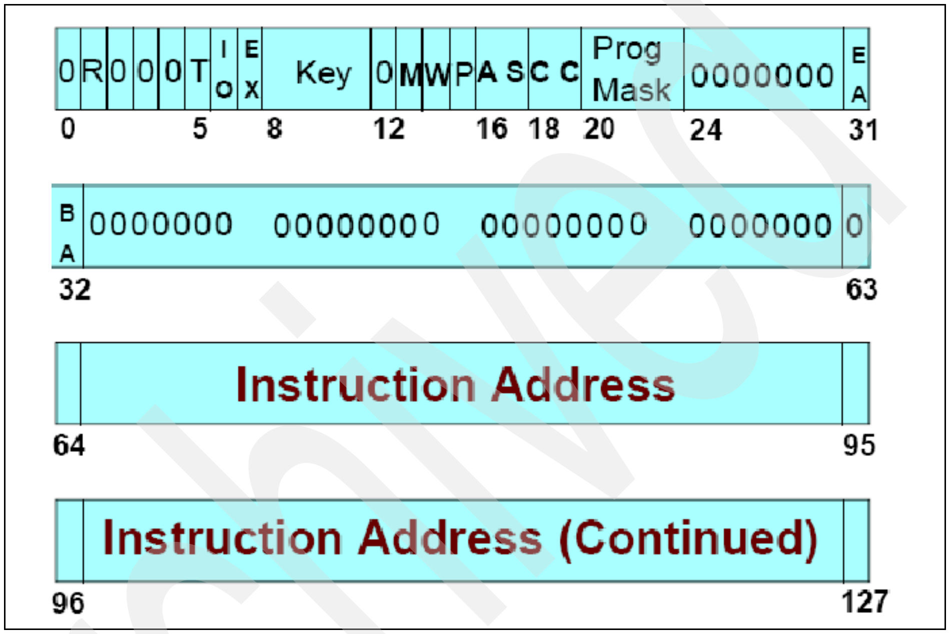
When an interrupt occurs, the CP saves pertinent information about the program that was interrupted and then routes control to the appropriate interrupt handler routine. The program status word, or PSW, is a key resource in this process.

## The program status word

The program status word (PSW) is a 128-bit data area in the processor that provides details crucial to both the hardware and the software. Although each processor has only one PSW, it is useful to think of three types of PSWs in order to understand interrupt processing. The three PSWs are the current PSW, the new PSW, and the old PSW.

The unit of work running on a CP is represented by the current PSW, which includes the address of the next program instruction and control information about the program that is running (every application running on z/OS has its own control information). The PSW controls the order in which instructions are fed to the processor, and indicates the status of the system in relation to the currently running program. The current PSW indicates the next instruction to be executed. It also indicates whether the processor is enabled or disabled for I/O interrupts, external interrupts, machine check interrupts, and certain program interrupts. When the processor is enabled, these interrupts can occur. When the processor is disabled, these interrupts are ignored or remain pending.

The format of the PSW is shown in Figure A-1.



0 R 0 0 0 T I O E X Key 0 M W P A S C C Prog Mask 0000000 E A
0 5 8 12 16 18 20 24 31
B A 0000000 00000000 00000000 0000000 0
32 63
Instruction Address
64 95
Instruction Address (Continued)
96 127

*Figure A-1 Overview of fields in PSW*

The bit positions indicated with a zero (0) do not have a function, at present. The following list describes the function of each non-zero bit, working from left to right:

- Bit 1 indicates that tracing is taking place, which will cause a program interrupt to occur when certain conditions are met.
- Bit 5 indicates that virtual-to-real address translation will take place (this is rarely set off).
- Bit 6 indicates that I/O interrupts can occur on this CP.
- Bit 7 indicates that external interrupts, such as a timed event, can occur on this CP.
- Bits 8 to 11 inclusive contain a storage key. A program running in a key other than zero is only allowed to alter storage with a matching key.

- Bit 13 indicates that machine check interrupts can occur on this CP. This is only off when a machine check is being handled.
- Bit 14 indicates that there is no more work to run on the CP and it should wait for an interrupt.
- Bit 15 indicates the problem state. When this is off, the CP is running in supervisor state and restricted instructions are allowed to be executed.
- Bits 16 and 17 indicate the translation mode for cross-memory functions.
- Bits 18 and 19 are a condition code. They indicate the outcome of an instruction. For example, if two values are compared, the condition code indicates high, low, or equal.
- Bits 20 to 23, when set off, prevent certain program checks from occurring. An example of this is fixed point overflow, which may be of no significance to a program processing certain types of scientific data.
- Bits 31 and 32 control the addressing mode of the CP. Both off means 24-bit addressing, both on indicates 64-bit addressing. Bit 31 off and bit 32 on indicates 31-bit addressing, and the fourth combination is invalid.
- Bits 64 to 95 contain the address of the next instruction to be executed by the CP.

## New and old PSWs

There is a new PSW and an old PSW associated with each of the six types of interrupts. The new PSW contains the address of the routine that can process its associated interrupt. The old and new PSWs are held in storage locations that are defined by the architecture. There is an old/new pair for each interrupt type.

If the processor is enabled for interrupts when an interrupt occurs, PSWs are switched through the following technique:

1. Storing the current PSW in the old PSW location associated with the type of interrupt that occurred
2. Loading the contents of the new PSW for the type of interrupt that occurred into the current PSW

The current PSW, which indicates the next instruction to be executed, now contains the address of the appropriate routine to handle the interrupt.

The new PSW always has the wait and program bits set off, and processing on the CP starts from the address in the PSW. This means that when an interrupt occurs, a z/OS routine gets control in supervisor state.

## Security

The PSW is the focal point for architected security. The program state bit restricts the instructions that can be issued and the protect key restricts the storage that may be altered. To request restricted services, a program in problem state issues an SVC. The SVC interrupt loads a new PSW, which switches to supervisor state.

Appendix B.

# Glossary

**A**

**abend.** abnormal end.

**abnormal end.** End of a task, a job, or a subsystem because of an error condition that cannot be resolved by recovery facilities while the task is performed. See also *abnormal termination*.

**abnormal termination.** (1) The end of processing prior to scheduled termination. (2) A system failure or operator action that causes a job to end unsuccessfully. Synonymous with *abend*, *abnormal end*.

**ACB.** access control block.

**ACCEPT.** The SMP/E command used to install SYSMODs in the distribution libraries.

**accept.** In SMP/E, to install SYSMODs in the distribution libraries. This is done with the ACCEPT command.

**accepted SYSMOD.** A SYSMOD that has been successfully installed by the SMP/E ACCEPT command. Accepted SYSMODs do not have the ERROR flag set and are found as SYSMOD entries in the distribution zone.

**access authority.** An authority that relates to a request for a type of access to protected resources. In RACF, the access authorities are NONE, READ, UPDATE, ALTER, and EXECUTE.

**access list.** A list within a profile of all authorized users and their access authorities.

**access method.** A technique for moving data between main storage and I/O devices.

**ACID properties.** The properties of a transaction: atomicity, consistency, isolation, and durability. In CICS, the ACID properties apply to a unit of work (UoW).

**address.** The unique code assigned to each device, workstation or system connected to a network.

**address space.** The complete range of addresses available to a program. In z/OS, an address space can range up to 16 exabytes of contiguous virtual storage addresses that the system creates for the user. An address space contains user data and programs, as well as system data and programs, some of which are common to all address spaces. See also *virtual address space*.

**addressing mode (AMODE).** A program attribute that refers to the address length that is expected to be in effect when the program is entered. In z/OS, addresses can be 24, 31, or 64 bits in length.

**administrator.** A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

**allocate.** To assign a resource for use in performing a specific task.

**ALLOCATE command.** In z/OS, the TSO/E command that serves as the connection between a file's logical name (the ddname) and the file's physical name (the data set name).

**alphanumeric character.** A letter or a number.

**AMODE.** addressing mode.

**ANSI.** American National Standards Institute.

**AOR.** application-owning region.

**APAR.** authorized program analysis report.

**APAR fix.** A temporary correction of a defect in an IBM system control program or licensed program that affects a specific user. An APAR fix is usually replaced later by a permanent correction called a PTF. APAR fixes are identified to SMP/E by the ++APAR statement.

**APF.** authorized program facility.

**API.** application programming interface.

**APPC.** Advanced Program-to-Program Communications.

**application.** A program or set of programs that performs a task; some examples are payroll, inventory management, and word processing applications.

**application-owning region (AOR).** In a CICSPlex® configuration, a CICS region devoted to running applications.

**application program.** A collection of software components used to perform specific types of work on a computer, such as a program that does inventory control or payroll.

**APPLY.** The SMP/E command used to install SYSMODs in the target libraries.

**apply.** In SMP/E, to install SYSMODs in the target libraries. This is done with the APPLY command.

**APPN.** Advanced Peer-to-Peer Network.

**ARM.** Automatic Restart Manager.

**ASCII.** American Standard Code for Information Interchange.

**ASID.** address space identifier.

**ASSEM entry.** An SMP/E entry containing assembler statements that can be assembled to create an object module.

**assembler.** A computer program that converts assembler language instructions into binary machine language (object code).

**assembler language.** A symbolic programming language that comprises instructions for basic computer operations which are structured according to the data formats, storage structures, and registers of the computer.

**asynchronous processing.** A series of operations that are done separately from the job in which they were requested; for example, submitting a batch job from an interactive job at a work station.

**ATM.** automated teller machine.

**audit.** To review and examine the activities of a data processing system mainly to test the adequacy and effectiveness of procedures for data security and data accuracy.

**authority.** The right to access objects, resources, or functions.

**authorization checking.** The action of determining whether a user is permitted access to a RACF-protected resource.

**authorized program analysis report (APAR).** A request for correction of a problem caused by a defect in a current unaltered release of a program. The correction is called an *APAR fix*.

**authorized program facility (APF).** A facility that permits identification of programs authorized to use restricted functions.

**automated operations.** Automated procedures to replace or simplify actions of operators in both systems and network operations.

**automatic call.** The process used by the linkage editor to resolve external symbols left undefined after all the primary input has been processed. See also *automatic call library.*

**automatic call library.** Contains load modules or object decks that are to be used as secondary input to the linkage editor to resolve external symbols left undefined after all the primary input has been processed.
The automatic call library may be:
- Libraries containing object decks, with or without linkage editor control statements
- Libraries containing load modules
- The library containing Language Environment® run-time routines.

**automatic library call.** Automatic call. See also *automatic call library.*

**automatic restart.** A restart that takes place during the current run, that is, without resubmitting the job. An automatic restart can occur within a job step or at the beginning of a job step. Contrast with *deferred restart*.

**automatic restart management.** A z/OS recovery function that improves the availability of batch jobs and started tasks. When a job fails, or the system on which it is running unexpectedly fails, z/OS can restart the job without operator intervention.

**auxiliary storage.** All addressable storage other than processor storage.

## B

**background.** (1) In multiprogramming, the environment on which low-priority programs are executed. (2) Under TSO/E, the environment on which jobs submitted through the SUBMIT command or SYSIN are executed. One job step at a time is assigned to a region of main storage, and it remains in main storage to completion. Contrast with *foreground*.

**background job.** (1) A low-priority job, usually a batched or non-interactive job. (2) Under TSO, a job entered through the SUBMIT command or through SYSIN. Contrast with *foreground job.*

**backout.** A request to remove all changes to resources since the last commit or backout or, for the first unit of recovery, since the beginning of the application. Backout is also called *rollback* or *abort*.

**backup.** The process of creating a copy of a data set to ensure against accidental loss.

**BAL.** Basic Assembler Language.

**base function.** In SMP/E, a SYSMOD defining elements of the base z/OS system or other products that were not previously present in the target libraries. Base functions are identified to SMP/E by the ++FUNCTION statement. SMP/E itself is an example of a base function of z/OS.

**base level system.** In SMP/E, the level of the target system modules, macros, source, and DLIBs created by system generation, to which function and service modifications are applicable.

**batch.** A group of records or data processing jobs brought together for processing or transmission. Pertaining to activity involving little or no user action. Contrast with *interactive*.

**batch job.** A predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system.

**batch message processing (BMP) program.** An IMS batch processing program that has access to online databases and message queues. BMPs run online, but like programs in a batch environment, they are started with job control language (JCL).

**batch processing.** A method of running a program or a series of programs in which one or more records (a batch) are processed with little or no action from the user or operator.

**BCP.** base control program.

**big endian.** A format for the storage of binary data in which the most significant byte is placed first. Big endian is used by most hardware architectures including the z/Architecture. Contrast with *little endian*.

**binary data.** (1) Any data not intended for direct human reading. Binary data may contain unprintable characters, outside the range of text characters. (2) A type of data consisting of numeric values stored in bit patterns of 0s and 1s. Binary data can cause a large number to be placed in a smaller space of storage.

**bind.** (1) To combine one or more control sections or program modules into a single program module, resolving references between them. (2) In SNA, a request to activate a session between two logical units (LUs).

**binder.** The z/OS program that processes the output of the language translators and compilers into an executable program (load module or program object). It replaces the linkage editor and batch loader used in earlier forms of the z/OS operating system, such as MVS and OS/390®.

**BLK.** A subparameter of the SPACE parameter in a DD statement. It specifies that space is allocated by blocks.

**BLKSIZE.** block size.

**BLOB.** binary large object.

**block size.** (1) The number of data elements in a block. (2) A measure of the size of a block, usually specified in units such as records, words, computer words, or characters. (3) Synonymous with *block length*. (4) Synonymous with *physical record size*.

**BPAM.** basic partitioned access method.

**BSAM.** basic sequential access method.

**buffer.** A portion of storage used to hold input or output data temporarily.

**bypass.** In SMP/E, to circumvent errors that would otherwise cause SYSMOD processing to fail. This is done by using the BYPASS operand on an SMP/E command.

**byte.** The basic unit of storage addressability. It has a length of 8 bits.

**byte stream.** A simple sequence of bytes stored in a stream file. See also *record data*.

**C**

**C language.** A high-level language used to develop software applications in compact, efficient code that can be run on different types of computers with minimal change.

**cabinet.** Housing for panels organized into port groups of patchports, which are pairs of fibre adapters or couplers. Cabinets are used to organize long, complex cables between processors and controllers, which may be as far away as another physical site. Also known as *fiber management cabinets*.

**cable "in inventory."** Unused cables.

**cache.** A random access electronic storage in selected storage controls used to retain frequently used data for faster access by the channel.

**cache structure.** A coupling facility structure that enables high-performance sharing of cached data by multisystem applications in a sysplex. Applications can use a cache structure to implement several different types of caching systems, including a store-through or a store-in cache.

**called routine.** A routine or program that is invoked by another.

**carriage control character.** An optional character in an input data record that specifies a write, space, or skip operation.

**carriage return (CR).** (1) A key stroke generally indicating the end of a command line. (2) In text data, the action that indicates to continue printing at the left margin of the next line. (3) A character that will cause printing to start at the beginning of the same physical line in which the carriage return occurred.

**CART.** command and response token.

**case-sensitive.** Pertaining to the ability to distinguish between uppercase and lowercase letters.

**catalog.** (1) A directory of files and libraries, with reference to their locations. (2) To enter information about a file or a library into a catalog. (3) The collection of all data set indexes that are used by the control program to locate a volume containing a specific data set.

**cataloged data set.** A data set that is represented in an index or hierarchy of indexes that provide the means for locating it.

**cataloged procedure.** A set of job control language (JCL) statements placed in a library and retrievable by name.

**CCW.** channel command word.

**CEMT**. The CICS-supplied transaction that allows checking of the status of terminals, connections, and other CICS entities from a console or from CICS terminal sessions.

**central processor (CP).** The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

**central processor complex (CPC).** A physical collection of hardware that includes main storage, one or more central processors, timers, and channels.

**central processing unit (CPU).** Synonymous with *processor*.

**main storage.** (1) In z/OS, the storage of a computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results. (Formerly referred to as "real storage".) (2) Synonymous with *processor storage*.

**CF.** Coupling Facility

**CFRM.** Coupling Facility resource management.

**CGI.** Common Gateway Interface.

**channel adapter.** A device that groups two or more controller channel interfaces electronically.

**channel connection address (CCA).** The input/output (I/O) address that uniquely identifies an I/O device to the channel during an I/O operation.

**channel interface.** The circuitry in a storage control that attaches storage paths to a host channel.

**channel path identifier.** The logical equivalent of channels in the physical processor.

**channel subsystem (CSS).** A collection of subchannels that directs the flow of information between I/O devices and main storage. Logical partitions use subchannels to communicate with I/O devices. The maximum number of CSSs supported by a processor also depends on the processor type. If more than one CSS is supported by a processor, each CSS has a processor unique single hexadecimal digit CSS identifier (CSS ID).

**channel-to-channel (CTC).** The communication (transfer of data) between programs on opposite sides of a channel-to-channel adapter (CTCA).

**channel-to-channel adapter (CTCA).** An input/output device that is used a program in one system to communicate with a program in another system.

**channel-to-channel (CTC) connection.** A connection between two CHPIDs on the same or different processors, either directly or through a switch. When connecting through a switch, both CHPIDs must be connected through the same or a chained switch.

**character.** A letter, digit, or other symbol. A letter, digit, or other symbol that is used as part of the organization, control, or representation of data. A character is often in the form of a spatial arrangement of adjacent or connected strokes.

**checkpoint.** (1) A place in a routine where a check, or a recording of data for restart purposes, is performed. (2) A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.

**checkpoint data set.** A data set in which information about the status of a job and the system can be recorded so that the job step can be restarted later.

**checkpoint write.** Any write to the checkpoint data set. A general term for the primary, intermediate, and final writes that update any checkpoint data set.

**CHPID.** channel path identifier.

**CI.** control interval.

**CICS.** Customer Information Control System.

**CICSplex.** A configuration of interconnected CICS systems in which each system is dedicated to one of the main elements of the overall workload. See also *application owning region* and *terminal owning region*.

**CKD.** count-key data.

**client.** A functional unit that receives shared services from a server. See also *client-server*.

**client-server.** In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

**CLIST.** command list.

**CLOB.** character large object.

**CLPA.** create link pack area.

**CMOS.** Complementary Metal Oxide Semiconductor.

**CMS.** Conversational Monitor System.

**COBOL.** Common Business-Oriented Language.

**code page.** (1) An assignment of graphic characters and control function meanings to all code points; for example, assignment of characters and meanings to 256 code points for an 8-bit code, assignment of characters and meanings to 128 code points for a 7-bit code. (2) A particular assignment of hexadecimal identifiers to graphic characters.

**code point.** A 1-byte code representing one of 256 potential characters.

**coexistence.** Two or more systems at different levels (for example, software, service or operational levels) that share resources. Coexistence includes the ability of a system to respond in the following ways to a new function that was introduced on another system with which it shares resources: ignore a new function; terminate gracefully; support a new function.

**command.** A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

**command and response token (CART).** A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

**command prefix.** A one- to eight-character command identifier. The command prefix distinguishes the command as belonging to an application or subsystem rather than to z/OS.

**COMMAREA.** A communication area made available to applications running under CICS.

**commit.** A request to make all changes to resources since the last commit or backout or, for the first unit of recovery, since the beginning of the application.

**Common Business-Oriented Language (COBOL).** A high-level language, based on English, that is primarily used for business applications.

**common service area (CSA).** In z/OS, a part of the common area that contains data areas that are addressable by all address spaces.

**compatibility.** Ability to work in the system or ability to work with other devices or programs.

**compilation unit.** A portion of a computer program sufficiently complete to be compiled correctly.

**compiler.** A program that translates a source program into an executable program (an object deck).

**compiler options.** Keywords that can be specified to control certain aspects of compilation. Compiler options can control the nature of the load module generated by the compiler, the types of printed output to be produced, the efficient use of the compiler, and the destination of error messages. Also called compiler-time options.

**complementary metal oxide semiconductor (CMOS).** A technology that combines the electrical properties of positive and negative voltage requirements to use considerably less power than other types of semiconductors.

**component.** A functional part of an operating system; for example, the scheduler or supervisor.

**condition code.** A code that reflects the result of a previous input/output, arithmetic, or logical operation.

**configuration.** The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units.

**connection.** In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In Internet communications, a connection extends from a TCP application on one system to a TCP application on another system.

**consistent copy.** A copy of data entity (for example, a logical volume) that contains the contents of the entire data entity from a single instant in time.

**console.** Any device from which operators can enter commands or receive messages.

**console group.** In z/OS, a group of consoles defined in CNGRPxx, each of whose members can serve as an alternate console in console or hardcopy recovery or as a console to display synchronous messages.

**control block.** A storage area used by a computer program to hold control information.

**control interval (CI).** A fixed-length area or disk in which VSAM stores records and creates distributed free space. Also, in a key-sequenced data set or file, the set of records that an entry in the sequence-set index record points to. The control interval is the unit of information that VSAM transmits to or from disk. A control interval always includes an integral number of physical records.

**control region.** The main storage region that contains the subsystem work manager or subsystem resource manager control program.

**control section (CSECT).** The part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining main storage locations.

**control statement.** In programming languages, a statement that is used to alter the continuous sequential execution of statements; a control statement can be a conditional statement, such as IF, or an imperative statement, such as STOP. In JCL, a statement in a job that is used in identifying the job or describing its requirements to the operating system.

**control unit (CU).** Each physical controller contains one or more control units, which translate high level requests to low level requests between processors and devices. Synonymous with *device control unit*.

**control unit address**. The high order bits of the storage control address, used to identify the storage control to the host system.

**controller.** A device that translates high level requests from processors to low level requests for I/O devices, and vice versa. Each physical controller contains one or more logical control units, channel and device interfaces, and a power source. Controllers can be divided into segments, or grouped into subsystems.

**conversation.** A logical connection between two programs over an LU type 6.2 session that allows them to communicate with each other while processing a transaction.

**conversational.** Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

**conversational monitor system (CMS).** A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under the control of the VM/370 control program.

**CORBA.** Common Object Request Broker Architecture.

**corequisite SYSMODs.** SYSMODs each of which can be installed properly only if the other is present. Corequisites are defined by the REQ operand on the ++VER statement.

**corrective service.** Any SYSMOD used to selectively fix a system problem. Generally, corrective service refers to APAR fixes.

**count-key data.** A disk storage device for storing data in the format: count field normally followed by a key field followed by the actual data of a record. The count field contains, in addition to other information, the address of the record in the format: CCHHR (where CC is the two-digit cylinder number, HH is the two-digit head number, and R is the record number) and the length of the data. The key field contains the record's key.

**couple data set.** A data set that is created through the XCF couple data set format utility and, depending on its designated type, is shared by some or all of the z/OS systems in a sysplex. See also *sysplex couple data set*.

**Coupling Facility.** A special logical partition that provides high-speed caching, list processing, and locking functions in a sysplex.

**Coupling Facility channel.** A high bandwidth fiber optic channel that provides the high-speed connectivity required for data sharing between a coupling facility and the central processor complexes directly attached to it.

**coupling services.** In a sysplex, the functions of XCF that transfer data and status between members of a group residing on one or more z/OS systems in the sysplex.

**CP.** central processor.

**CPC.** central processor complex.

**CPU.** central processing unit.

**create link pack area (CLPA).** An option that is used during IPL to initialize the link pack pageable area.

**crossbar switch.** A static switch that can connect controllers to processors with parallel (bus and tag) interfaces. The crossbar contains a number of channel interfaces on its top, which can connect to objects above it such as processors or other crossbars. The crossbar switch also contains a number of control unit interfaces on its side, which can connect to objects below it such as controllers or other crossbars.

**cross-memory linkage.** A method for invoking a program in a different address space. The invocation is synchronous with respect to the caller.

**cross-system coupling facility (XCF).** A component of z/OS that provides functions to support cooperation between authorized programs running within a sysplex.

**cross-system extended services (XES).** A set of z/OS services that allow multiple instances of an application or subsystem, running on different systems in a sysplex environment, to implement high-performance, | high-availability data sharing by using a coupling facility.

**cross-system restart.** If a system fails, automatic restart management restarts elements on another eligible system in the sysplex.

**cryptographic key.** A parameter that determines cryptographic transformations between plaintext and ciphertext.

**cryptography.** The transformation of data to conceal its meaning.

**CSA.** common service area.

**CSI.** consolidated software inventory data set. See *SMPCSI*.

**CSS.** channel subsystem.

**CSECT.** control section.

**CTC.** channel-to-channel.

**CTC connection.** channel-to-channel connection.

**cumulative service tape.** A tape sent with a new function order, containing all current PTFs for that function.

**Customer Information Control System (CICS).** An online transaction processing (OLTP) system that provides specialized interfaces to databases, files and terminals in support of business and commercial applications. CICS enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

**D**

**daemon.** In UNIX systems, a long-lived process that runs unattended to perform continuous or periodic system-wide functions, such as network control. Some daemons are triggered automatically to perform their task; others operate periodically. An example is the cron daemon, which periodically performs the tasks listed in the crontab file. The z/OS equivalent is a started task.

**DASD.** direct access storage device.

**DASD volume.** A DASD space identified by a common label and accessed by a set of related addresses. See also *volume*.

**data class.** A collection of allocation and space attributes, defined by the storage administrator, that are used that are used when allocating a new SMS-managed data set.

**data control block (DCB).** A control block used by access method routines in storing and retrieving data.

**data definition name (ddname).** (1) The name of a data definition (DD) statement that corresponds to a data control block that contains the same name. (2) The symbolic representation for a name placed in the name field of a data definition (DD) statement.

**data definition (DD) statement.** A job control statement that describes a data set associated with a particular job step.

**data definition name.** See *ddname*.

**data definition statement.** A JCL control statement that serves as the connection between a file's logical name (the ddname) and the file's physical name (the data set name).

**data division.** In COBOL, the part of a program that describes the files to be used in the program and the records contained within the files. It also describes any WORKING-STORAGE data items, LINKAGE SECTION data items, and LOCAL-STORAGE data items that are needed.

**Data Facility Sort (DFSORT™).** An IBM licensed program that is a high-speed data-processing utility. DFSORT provides a method for sorting, merging, and copying operations, as well as providing versatile data manipulation at the record, field, and bit level.

**data in transit.** The update data on application system DASD volumes that is being sent to the recovery system for writing to DASD volumes on the recovery system.

**data integrity.** The condition that exists when accidental or intentional destruction, alteration, or loss of data does not occur.

**data set.** In z/OS, a named collection of related data records that is stored and retrieved by an assigned name. Equivalent to a file.

**data set backup.** Backup to protect against the loss of individual data sets.

**data set label.** (1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

**data sharing.** The ability of concurrent subsystems (such as DB2 or IMS DB) or application programs to directly access and change the same data, while maintaining data integrity.

**data stream.** (1) All information (data and control commands) sent over a data link usually in a single read or write operation. (2) A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

**data type.** The properties and internal representation that characterize data.

**data warehouse.** A system that provides critical business information to an organization. The data warehouse system cleanses the data for accuracy and currency, and then presents the data to decision makers so that they can interpret and use it effectively and efficiently.

**database.** A collection of tables, or a collection of table spaces and index spaces.

**database administrator (DBA).** An individual who is responsible for designing, developing, operating, safeguarding, maintaining, and using a database.

**database management system (DBMS).** A software system that controls the creation, organization, and modification of a database and the access to the data that is stored within it.

**DBCS.** double-byte character set.

**DBMS.** database management system.

**DB2.** DATABASE 2; generally, one of a family of IBM relational database management systems and, specifically, the system that runs under z/OS.

**DB2 data sharing group.** A collection of one or more concurrent DB2 subsystems that directly access and change the same data while maintaining data integrity.

**DCB.** data control block.

**DCLGEN.** declarations generator.

**ddname.** data definition name.

**DD statement.** data definition statement.

**deadlock.** (1) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by or a response from the other. (2) Unresolvable contention for the use of a resource. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that does not become available because it is being held by another process that is in a similar wait state.

**deallocate.** To release a resource that is assigned to a specific task.

**declarations generator (DCLGEN).** A subcomponent of DB2 that generates SQL table declarations and COBOL, C, or PL/I data structure declarations that conform to the table. The declarations are generated from DB2 system catalog information.

**dedicated.** Pertaining to the assignment of a system resource—a device, a program, or a whole system—to an application or purpose.

**default.** A value that is used or an action that is taken when no alternative is specified by the user.

**deferred restart.** A restart performed by the system when a user resubmits a job. The operator submits the restart deck to the system through a system input reader. See also *checkpoint restart*. Contrast with *automatic restart*.

**deleted function.** In SMP/E, a function that was removed from the system when another function was installed. This is indicated by the DELBY subentry in the SYSMOD entry for the deleted function.

**destination.** A combination of a node name and one of the following: a user ID, a remote printer or punch, a special local printer, or LOCAL (the default if only a node name is specified).

**destination node.** The node that provides application services to an authorized external user.

**device.** A computer peripheral or an object that appears to the application as such.

**device address**. The field of an ESCON device-level frame that selects a specific device on a control unit image. The one or two left-most digits are the address of the channel to which the device is attached. The two rightmost digits represent the unit address.

**device control unit.** A hardware device that controls the reading, writing, or displaying of data at one or more I/O devices or terminals.

**device number**. A four-hexadecimal-character identifier, for example 13A0, that you associate with a device to facilitate communication between the program and the host operator. The device number that you associate with a subchannel.

**Device Support Facilities program (ICKDSF).** A program used to initialize DASD volumes at installation and perform media maintenance.

**DFSMS.** Data Facility Storage Management Subsystem.

**DFSMShsm.** An IBM product used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

**device type.** The general name for a kind of device; for example, 3390.

**DFS™.** Distributed File Service.

**DFSORT.** Data Facility Sort.

**dialog.** An interactive pop-up window containing options that allow you to browse or modify information, take specific action relating to selected objects, or access other dialogs. For example, HCM provides a series of dialogs to help you create, edit, delete, and connect objects, as well as manipulate the configuration diagram.

**direct access storage device (DASD).** A device in which the access time is effectively independent of the location of the data.

**directory.** (1) A type of file containing the names and controlling information for other files or other directories. Directories can also contain subdirectories, which can contain subdirectories of their own. (2) A file that contains directory entries. No two directory entries in the same directory can have the same name. (POSIX.1). (3) A file that points to files and to other directories. (4) An index used by a control program to locate blocks of data that are stored in separate areas of a data set in direct access storage.

**disaster recovery.** Recovery after a disaster, such as a fire, that destroys or otherwise disables a system. Disaster recovery techniques typically involve restoring data to a second (recovery) system, then using the recovery system in place of the destroyed or disabled application system. See also *recovery*, *backup*, and *recovery system*.

**DISP.** Disposition (JCL DD parameter).

**display console.** In z/OS, an MCS console whose input/output function you can control.

**distributed computing.** Computing that involves the cooperation of two or more machines communicating over a network. Data and resources are shared among the individual computers.

**Distributed Computing Environment (DCE).** A comprehensive, integrated set of services that supports the development, use, and maintenance of distributed applications. DCE is independent of the operating system and network; it provides interoperability and portability across heterogeneous platforms.

**distributed data.** Data that resides on a DBMS other than the local system.

**Distributed File Service (DFS).** A DCE component. DFS joins the local file systems of several file server machines making the files equally available to all DFS client machines. DFS allows users to access and share files stored on a file server anywhere in the network, without having to consider the physical location of the file. Files are part of a single, global namespace, so that a user can be found anywhere in the network by means of the same name.

**distribution library (DLIB).** A library that contains the master copy of all the elements in a system. A distribution library can be used to create or back up a target library.

**distribution zone.** In SMP/E, a group of records in a CSI data set that describes the SYSMODs and elements in a distribution library.

**DLIB.** distribution library.

**DLL.** dynamic link library.

**double-byte character set (DBCS).** A set of characters in which each character is represented by a two-bytes code. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires two bytes, the typing, display, and printing of DBCS characters requires hardware and programs that support DBCS. Contrast with *single-byte character set*.

**doubleword.** A sequence of bits or characters that comprises eight bytes (two 4-byte words) and is referenced as a unit.

**downwardly compatible.** The ability of applications to run on previous releases of z/OS.

**drain.** Allowing a printer to complete its current work before stopping the device.

**driving system.** The system used to install the program. Contrast with *target system*.

**dsname.** data set name.

**DSORG.** Data set organization (parameter of DCB and DD and in a data class definition).

**dump.** A report showing the contents of storage. Dumps are typically produced following program failures, for use as diagnostic aids.

**dynamic allocation.** Assignment of system resources to a program at the time the program is executed rather than at the time it is loaded into main storage.

**dynamic link library (DLL).** A file containing executable code and data bound to a program at load time or run time. The code and data in a dynamic link library can be shared by several applications simultaneously.

**dynamic reconfiguration.** The ability to make changes to the channel subsystem and to the operating system while the system is running.

## E

**e-business.** (1) The transaction of business over an electronic medium such as the Internet. (2) The transformation of key business processes through the use of Internet technologies.

**EB.** See *exabyte*.

**EBCDIC.** Extended Binary Coded Decimal Interchange Code.

**EC.** engineering change.

**ECSA**. extended common service area.

**EDT.** eligible device table.

**element.** In SMP/E, part of a product, such as a macro, module, dialog panel, or sample code.

**eligible device table (EDT).** An installation defined representation of the devices that are eligible for allocation. The EDT defines the esoteric and generic relationship of these devices. During IPL, the installation identifies the EDT that z/OS uses. After IPL, jobs can request device allocation from any of the esoteric device groups assigned to the selected EDT. An EDT is identified by a unique ID (two digits), and contains one or more esoterics and generics.

**enclave.** A transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as a unit.

**encrypt.** To systematically encode data so that it cannot be read without knowing the coding key.

**endian.** An attribute of data representation that reflects how certain multi-octet data is stored in memory. See big endian and little endian.

**enterprise.** The composite of all operational entities, functions, and resources that form the total business concern.

**Enterprise Systems Connection (ESCON).** A set of products and services that provides a dynamically connected environment using optical cables as a transmission medium.

**entry area.** In z/OS, the part of a console screen where operators can enter commands or command responses.

**entry name.** In assembler language, a programmer-specified name within a control section that identifies an entry point and can be referred to by any control section. See also *entry point*.

**entry point.** The address or label of the first instruction that is executed when a routine is entered for execution. Within a load module, the location to which control is passed when the load module is invoked.

**entry point name.** The symbol (or name) that represents an entry point. See also *entry point*.

**esoteric.** Esoteric (or esoteric device group) is an installation-defined and named grouping of I/O devices of usually the same device group. Eligible device tables (EDTs) define the esoteric and generic relationship of these devices. The name you assign to an esoteric is used in the JCL DD statement. The job then allocates a device from that group instead of a specific device number or generic device group.

**EOF.** End of file.

**ESCON.** Enterprise Systems Connection.

**ETR.** External Time Reference. See also *Sysplex Timer*.

**exabyte.** For processor, real and virtual storage capacities and channel volume: 1 152 921 504 606 846 976 bytes or $2^{(60)}$.

**exception SYSMOD.** A SYSMOD that is in error or that requires special processing before it can be installed. ++HOLD and ++RELEASE statements identify exception SYSMODs.

**EXCP.** execute channel programs.

**executable.** A load module or program object which has yet to be loaded into memory for execution.

**executable program.** (1) A program in a form suitable for execution by a computer. The program can be an application or a shell script. (2) A program that has been link-edited and can therefore be run in a processor. (3) A program that can be executed as a self-contained procedure. It consists of a main program and, optionally, one or more subprograms. (4) See also *executable file*, *load module*.

**Extended Binary-Coded Decimal Interchange Code (EBCDIC).** An encoding scheme that is used to represent character data in the z/OS environment. Contrast with *ASCII* and *Unicode*.

**extended MCS console.** In z/OS, a console other than an MCS console from which operators or programs can issue system commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

**Extended Remote Copy (XRC)**. A hardware- and software-based remote copy service option that provides an asynchronous volume copy across storage subsystems for disaster recovery, device migration, and workload migration.

**external reference.** In an object deck, a reference to a symbol, such as an entry point name, defined in another program or module.

## F

**feature.** A part of an IBM product that may be ordered separately by a client.

**feature code.** A four-digit code used by IBM to process hardware and software orders.

**fetch.** The dynamic loading of a procedure.

**Fiber Connection Environment (FICON).** An optical fiber communication method offering channels with high data rate, high bandwidth, increased distance and a large number of devices per control unit for mainframe systems. It can work with, or replace, ESCON links.

**fiber link.** The physical fiber optic connections and transmission media between optical fiber transmitters and receivers. A fiber link can comprise one or more fiber cables and patchports in fiber management cabinets. Each connection in the fiber link is either permanent or mutable.

**FICON.** Fiber Connection Environment.

**FIFO.** first in, first out.

**file.** A named collection of related data records that is stored and retrieved by an assigned name. Equivalent to a z/OS data set.

**FILEDEF.** file definition statement.

**first in, first out.** A queuing technique in which the next item to be retrieved is the oldest item in the queue.

**firewall.** An intermediate server that functions to isolate a secure network from an insecure network.

**fix**. A correction of an error in a program, usually a temporary correction or bypass of defective code.

**fixed-length record.** A record having the same length as all other records with which it is logically or physically associated. Contrast with *variable-length record*.

**FlashCopy**. A point-in-time copy services function that can quickly copy data from a source location to a target location.

**FMID.** function modification identifier.

**foreground.** (1) In multiprogramming, the environment on which high-priority programs are executed. (2) Under TSO, the environment on which programs are swapped in and out of main storage to allow CPU time to be shared among terminal users. All command processor programs execute in the foreground. Contrast with *background*.

**foreground job.** (1) A high-priority job, usually a real-time job. (2) Under TSO, any job executing in a swapped region of main storage, such as a command processor or a terminal user's program. Contrast with *background job*.

**foreign key.** A column or set of columns in a dependent table of a constraint relationship. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table. Each foreign key value must either match a parent key value in the related parent table or be null.

**fork.** To create and start a child process. Forking is similar to creating an address space and attaching. It creates a copy of the parent process, including open file descriptors.

**Fortran.** A high-level language used primarily for applications involving numeric computations. In previous usage, the name of the language was written in all capital letters, that is, FORTRAN.

**frame.** For a mainframe microprocessor cluster, a frame contains one or two central processor complexes (CPCs), support elements, and AC power distribution.

**FTP.** File Transfer Protocol.

**fullword.** A sequence of bits or characters that comprises four bytes (one word) and is referenced as a unit.

**fullword boundary.** A storage location whose address is evenly divisible by 4.

**function.** In SMP/E, a product (such as a system component or licensed program) that can be installed in a user's system if desired. Functions are identified to SMP/E by the ++FUNCTION statement. Each function must have a unique FMID.

**function modification identifier (FMID).** A code that identifies the release levels of a z/OS licensed program.

## G

**gateway node.** A node that is an interface between networks.

**GB.** gigabyte (1 073 741 824 bytes).

**GDG.** generation data group.

**generalized trace facility (GTF).** Like system trace, gathers information used to determine and diagnose problems that occur during system operation. Unlike system trace, however, GTF can be tailored to record very specific system and user program events.

**generation data group (GDG).** A collection of historically related non-VSAM data sets that are arranged in chronological order; each data set is called a generation data set.

**generic.** A z/OS-defined grouping of devices with similar characteristics. For example: the device types 3270-X, 3277-2, 3278-2, -2A, -3, -4, and 3279-2a, -2b, -2c, -3a, -3b belong to the same generic. Every generic has a generic name that is used for device allocation in the JCL DD statement. z/OS interprets this name as “take any device in that group.” In a given z/OS configuration, each eligible device table (EDT) has the same list of generics.

**Geographically Dispersed Parallel Sysplex (GDPS).** An application that integrates Parallel Sysplex technology and remote copy technology to enhance application availability and improve disaster recovery. GDPS topology is a Parallel Sysplex cluster spread across two sites, with all critical data mirrored between the sites. GDPS manages the remote copy configuration and storage subsystems; automates Parallel Sysplex operational tasks; and automates failure recovery from a single point of control.

**gigabyte.** $2^{30}$ bytes, 1 073 741 824 bytes. This is approximately a billion bytes in American English.

**global access checking.** The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources.

**global resource serialization.** A function that provides a z/OS serialization mechanism for resources (typically data sets) across multiple z/OS images.

**global resource serialization complex.** One or more z/OS systems that use global resource serialization to serialize access to shared resources (such as data sets on shared DASD volumes).

**global zone.** A group of records in a CSI data set used to record information about SYSMODs received for a particular system. The global zone also contains information that (1) enables SMP/E to access target and distribution zones in that system, and (2) enables you to tailor aspects of SMP/E processing.

**Gregorian calendar.** The calendar in use since Friday, 15 October 1582 throughout most of the world.

**group.** A collection of RACF users who can share access authorities for protected resources.

## H

**hardcopy log.** In systems with multiple console support or a graphic console, a permanent record of system activity.

**hardware.** Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with *software*.

**hardware configuration dialog (HCD).** In z/OS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for z/OS system configurations.

**Hardware Management Console (HMC).** A console used to monitor and control hardware such as the mainframe microprocessors.

**hardware unit.** A central processor, storage element, channel path, device, and so on.

**HASP.** Houston Automatic Spooling Priority.

**HCD.** Hardware Configuration Definition.

**head of string.** The first unit of devices in a string. It contains the string interfaces which connect to controller device interfaces.

**hexadecimal.** A base 16 numbering system. Hexadecimal digits range from 0 through 9 (decimal 0 to 9) and uppercase or lowercase A through F (decimal 10 to 15) and A through F, giving values of 0 through 15.

**HFS.** hierarchical file system.

**hierarchical file system (HFS).** A data set that contains a POSIX-compliant file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using z/OS UNIX System Services.

**hierarchical file system (HFS) data set.** A data set that contains a POSIX-compliant hierarchical file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using z/OS UNIX System Services facilities.

**high-level language (HLL).** A programming language above the level of assembler language and below that of program generators and query languages. Examples are C, C++, COBOL, Fortran, and PL/I.

**HLL.** high-level language.

**highly parallel.** Refers to multiple systems operating in parallel, each of which can have multiple processors. See also *n-way*.

**HMC.** Hardware Management Console.

**HOLDDATA.** In SMP/E, one or more MCSs used to indicate that certain SYSMODs contain errors or require special processing before they can be installed. ++HOLD and ++RELEASE statements are used to define HOLDDATA. SYSMODs affected by HOLDDATA are called exception SYSMODs.

**Houston Automatic Spooling Priority (HASP).** A computer program that provides supplementary job management, data management, and task management functions, such as: control of job flow, ordering of tasks, and spooling. See also *JES2*.

**I**

**I/O.** Input/output.

**I/O cluster.** A sysplex that owns a managed channel path for a logically partitioned processor configuration.

**I/O device.** A printer, tape drive, hard disk drive, and so on. Devices are logically grouped inside units, which are in turn grouped into strings. The first unit, known as the head of string, contains string interfaces which connect to controller device interfaces and eventually to processor CHPIDs. Devices are represented as lines of text within the appropriate unit object in the configuration diagram.

**IBM Support Center.** The IBM organization responsible for software service.

**IBM systems engineer (SE).** An IBM service representative who performs maintenance services for IBM software in the field.

**ICSF.** Integrated Cryptographic Service Facility.

**IDCAMS.** An IBM program used to process access method services commands. It can be invoked as a job or jobstep, from a TSO terminal, or from within a user's application program.

**image.** A single instance of the z/OS operating system.

**IMS.** Information Management System.

**IMS DB.** Information Management System Database Manager.

**IMS DB data sharing group.** A collection of one or more concurrent IMS DB subsystems that directly access and change the same data while maintaining data integrity.

**Information Management System (IMS).** IBM product that supports hierarchical databases, data communication, translation processing, and database backout and recovery.

**initial program load (IPL).** The initialization procedure that causes the z/OS operating system to begin operation. During IPL, system programs are loaded into storage and z/OS is made ready to perform work. Synonymous with *boot*, *load*.

**initial storage allocation.** The amount of central and expanded storage to be assigned to a logical partition.

**initiator.** That part of an operating system that reads and processes operation control language statements from the system input device.

**initiator/terminator.** The job scheduler function that selects jobs and job steps to be executed, allocates input/output devices for them, places them under task control, and at completion of the job, supplies control information for writing job output on a system output unit.

**input/output configuration data set (IOCDS).** A file that contains different configuration definitions for the selected processor. Only one IOCDS is used at a time. The IOCDS contains I/O configuration data for the files associated with the processor controller on the host processor, as it is used by the channel subsystem. The channel subsystem (CSS) uses the configuration data to control I/O requests. The IOCDS is built from the production IODF.

**input/output definition file (IODF).** A VSAM linear data set that contains I/O definition information, including processor I/O definitions and operating system I/O definitions, including all logical objects and their connectivity in the hardware configuration.

**install.** In SMP/E, to apply a SYSMOD to the target libraries or to accept a SYSMOD into the distribution libraries.

**installation exit.** The means by which an IBM software product may be modified by a client's systems programmers to change or extend the functions of the product.

**instruction line.** In z/OS, the part of the console screen that contains messages about console control and input errors.

**interactive.** Pertaining to a program or system that alternately accepts input and responds. In an interactive system, a constant dialog exists between user and system. Contrast with *batch*.

**interactive problem control system (IPCS).** A component of z/OS that permits online problem management, interactive problem diagnosis, online debugging for dumps, problem tracking, and problem reporting.

**Interactive System Productivity Facility (ISPF).** A dialog manager for interactive applications. It provides control and services to permit execution of dialogs.

**internal reader.** A facility that transfers jobs to JES.

**interrupt.** A suspension of a process, such as the execution of a computer program, caused by an event external to that process, and performed in such a way that the process can be resumed.

**IOCDS.** input/output configuration data set.

**IODF.** input/output definition file.

**IPCS.** Interactive Problem Control System.

**IPL.** initial program load.

**IPv6.** Internet Protocol Version 6.

**ISMF.** interactive storage management facility.

**ISPF.** Interactive System Productivity Facility.

**ISPF/PDF.** Interactive System Productivity Facility/Program Development Facility.

**IVP.** installation verification procedure.

## J

**JCL.** job control language.

**JES.** job entry subsystem.

**JES2.** A z/OS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In an installation with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing. Contrast with *JES3*.

**JES3.** A z/OS subsystem that receives jobs into the system, converts them to internal format, selects them for execution, processes their output, and purges them from the system. In complexes that have several loosely-coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue. Contrast with *JES2*.

**job.** A unit of work for an operating system. Jobs are defined by JCL statements.

**job class.** Any one of a number of job categories that can be defined. With the classification of jobs and direction of initiator/terminators to initiate specific classes of jobs, it is possible to control the mixture of jobs that are performed concurrently.

**job control language (JCL).** A sequence of commands used to identify a job to an operating system and to describe a job's requirements.

**job control language (JCL) statements.** Statements placed into an input stream to define work to be done, methods to be used, and the resources needed.

**job entry subsystem (JES).** A system facility for spooling, job queuing, and managing I/O.

**job entry subsystem 2.** See *JES2*.

**job entry subsystem 3.** See *JES3*.

**job priority.** A value assigned to a job that is used as a measure of the job's relative importance while the job contends with other jobs for system resources.

**job separator pages.** Those pages of printed output that delimit jobs.

**job step.** The job control (JCL) statements that request and control execution of a program and that specify the resources needed to run the program. The JCL statements for a job step include one EXEC statement, which specifies the program or procedure to be invoked, followed by one or more DD statements, which specify the data sets or I/O devices that might be needed by the program.

**Julian date.** A date format that contains the year in positions 1 and 2, and the day in positions 3 through 5. The day is represented as 1 through 366, right-adjusted, with zeros in the unused high-order position.

**jumper cable.** Fiber used to make mutable connections between patchports.

## K

**kernel.** The part of an operating system that performs basic functions such as allocating hardware resources.

**key-sequenced data set (KSDS).** A VSAM file or data set whose records are loaded in ascending key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence by means of distributed free space. Relative byte addresses can change because of control interval or control area splits.

**keyword.** A part of a command operand that consists of a specific character string (such as DSNAME=).

**KSDS.** key-sequenced data set.

## L

**LAN.** local area network.

**Language Environment.** Short form of z/OS Language Environment. A set of architectural constructs and interfaces that provides a common run-time environment and run-time services for C, C++, COBOL, Fortran, PL/I, VisualAge® PL/I, and Java applications compiled by Language Environment-conforming compilers.

**last in, first out (LIFO).** A queuing technique in which the next item to be retrieved is the item most recently placed in the queue.

**LCSS.** logical channel subsystem.

**LCU.** logical control unit.

**LDAP.** Lightweight Directory Access Protocol.

**library.** A partitioned data set (PDS) that contains a related collection of named members. See *partitioned data set*.

**LIC.** Licensed Internal Code.

**licensed internal code (LIC**). Microcode that IBM does not sell as part of a machine, but licenses to the client. LIC is implemented in a part of storage that is not addressable by user programs. Some IBM products use it to implement functions as an alternative to hard-wired circuitry.

**licensed program.** A software package that can be ordered from the program libraries, such as IBM Software Distribution (ISMD). IMS and CICS are examples of licensed programs.

**Lightweight Directory Access Protocol (LDAP).** An Internet protocol standard, based on the TCP/IP protocol, which allows the access and manipulation of data organized in a Directory Information Tree (DIT).

**LIFO.** last in, first out.

**link library.** A data set containing link-edited object modules.

**link pack area (LPA).** An area of virtual storage that contains reenterable routines that are loaded at IPL (initial program load) time and can be used concurrently by all tasks in the system.

**linkage editor.** An operating system component that resolves cross-references between separately compiled or assembled modules and then assigns final addresses to create a single relocatable load module. The linkage editor then stores the load module in a load library on disk.

**linked list.** A list in which the data elements may be dispersed but in which each data element contains information for locating the next. Synonymous with *chained list*.

**link-edit.** To create a loadable computer program by means of a linkage editor or binder.

**list structure.** A Coupling Facility structure that enables multisystem applications in a sysplex to share information organized as a set of lists or queues. A list structure consists of a set of lists and an optional lock table, which can be used for serializing resources in the list structure. Each list consists of a queue of list entries.

**little endian.** A format for storage of binary data in which the least significant byte is placed first. Little endian is used by the Intel® hardware architectures. Contrast with *big endian*.

**LMOD.** In SMP/E, an abbreviation for load module.

**load module.** An executable program stored in a partitioned data set program library. See also *program object*.

**local area network (LAN).** A network in which communication is limited to a moderate-sized geographical area (1 to 10 km) such as a single office building, warehouse, or campus, and which does not generally extend across public rights-of-way. A local network depends on a communication medium capable of moderate to high data rate (greater than 1 Mbps), and normally operates with a consistently low error rate.

**local system queue area (LSQA).** In z/OS, one or more segments associated with each virtual storage region that contain job-related system control blocks.

**lock structure.** A Coupling Facility structure that enables applications in a sysplex to implement customized locking protocols for serialization of application-defined resources. The lock structure supports shared, exclusive, and application-defined lock states, as well as generalized contention management and recovery protocols.

**logical control unit (LCU).** A single control unit (CU) with or without attached devices, or a group of one or more CUs that share devices. In a channel subsystem (CSS), an LCU represents a set of CUs that physically or logically attach I/O devices in common.

**logical partition (LP).** A subset of the processor hardware that is defined to support an operating system. See also *logically partitioned (LPAR) mode*.

**logical partitioning.** A function of an operating system that enables the creation of logical partitions.

**logical subsystem**. The logical functions of a storage controller that allow one or more host I/O interfaces to access a set of devices. The controller aggregates the devices according to the addressing mechanisms of the associated I/O interfaces. One or more logical subsystems exist on a storage controller. In general, the controller associates a given set of devices with only one logical subsystem.

**logical unit (LU).** In SNA, a port through which an end user accesses the SNA network in order to communicate with another end user, and through which the end user accesses the functions provided by system services control points (SSCPs).

**logical unit type 6.2.** The SNA logical unit type that supports general communication between programs in a cooperative processing environment.

**logically partitioned (LPAR) mode.** A central processor complex (CPC) power-on reset mode that enables use of the PR/SM feature and allows an operator to allocate CPC hardware resources (including central processors, main storage, expanded storage, and channel paths) among logical partitions.

**logoff.** (1) The procedure by which a user ends a terminal session. (2) In VTAM®, a request that a terminal be disconnected from a VTAM application program.

**logon.** (1) The procedure by which a user begins a terminal session. (2) In VTAM, a request that a terminal be connected to a VTAM application program.

**loop.** A situation in which an instruction or a group of instructions execute repeatedly.

**loosely coupled.** A multisystem structure that requires a low degree of interaction and cooperation between multiple z/OS images to process a workload. See also *tightly coupled*.

**LP.** logical partition.

**LPA.** link pack area.

**LPAR.** logically partitioned (mode).

**LRECL.** logical record length.

**LSQA.** local system queue area

**LU.** logical unit.

## M

**machine check interruption.** An interruption that occurs as a result of an equipment malfunction or error.

**machine readable.** Pertaining to data a machine can acquire or interpret (read) from a storage device, a data medium, or other source.

**macro.** An instruction in a source language that is to be replaced by a defined sequence of instructions in the same source language.

**main task.** In the context of z/OS multitasking, the main program in a multitasking environment.

**MAS.** multi-access spool configuration.

**master catalog.** A catalog that contains extensive data set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

**master IODF.** A centrally kept IODF containing I/O definitions for several systems or even for a complete enterprise structure. Master IODFs help to maintain consistent I/O data and can provide comprehensive reports.

**master trace.** A centralized data tracing facility of the master scheduler, used in servicing the message processing portions of z/OS.

**MB.** megabyte.

**MCS.** (1) Multiple console support. (2) Modification control statement (in SMP/E).

**MCS console.** A non-SNA device defined to z/OS that is locally attached to a z/OS system and is used to enter commands and receive messages.

**megabyte (MB).** $2^{20}$ bytes, 1 048 576 bytes.,048,576 bytes.

**member.** A partition of a partitioned data set (PDS) or partitioned data set extended (PDSE).

**message processing facility (MPF).** A facility used to control message retention, suppression, and presentation.

**message queue.** A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

**message text.** The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

**microcode.** Stored microinstructions, not available to users, that perform certain functions.

**microprocessor.** A processor implemented on one or a small number of chips.

**migration.** Refers to activities, often performed by the system programmer, that relate to the installation of a new version or release of a program to replace an earlier level. Completion of these activities ensures that the applications and resources on a system will function correctly at the new level.

**mixed complex.** A global resource serialization complex in which one or more of the systems in the global resource serialization complex are not part of a multisystem sysplex.

**modification control statement (MCS).** An SMP/E control statement used to package a SYSMOD. MCSs describe the elements of a program and the relationships that program has with other programs that may be installed on the same system.

**modification level.** A distribution of all temporary fixes that have been issued since the previous modification level. A change in modification level does not add new functions or change the programming support category of the release to which it applies. Contrast with *release* and *version*. Whenever a new release of a program is shipped, the modification level is set to 0. When the release is reshipped with the accumulated services changes incorporated, the modification level is incremented by 1.

**module.** The object that results from compiling source code. A module cannot be run. To be run, a module must be bound into a program.

**monoplex.** A sysplex consisting of one system that uses a sysplex couple data set.

**multi-access spool configuration.** Multiple systems sharing the JES2 input, job and output queues (through a checkpoint data set or coupling facility).

**multiple console support (MCS).** The operator interface in a z/OS system.

**Multiple Virtual Storage (MVS).** An earlier form of the z/OS operating system.

**multiprocessing.** The simultaneous execution of two or more computer programs or sequences of instructions. See also *parallel processing*.

**multiprocessor (MP).** A CPC that can be physically partitioned to form two operating processor complexes.

**multisystem application.** An application program that has various functions distributed across z/OS images in a multisystem environment.

**multisystem console support.** Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

**multisystem environment.** An environment on which two or more z/OS images reside in one or more processors, and programs on one image can communicate with programs on the other images.

**multisystem sysplex.** A sysplex in which two or more z/OS images are allowed to be initialized as part of the sysplex.

**multitasking.** Mode of operation that provides for the concurrent, or interleaved, execution of two or more tasks, or threads. Synonymous with *multithreading*.

**mutable connection.** Connections made with fiber jumper cables between patchports in a cabinet or between cabinets and active objects such as CHPIDs, switches, converters and controllers with ESCON or FICON interfaces. Mutable connections are broken when the patchports they connect are not in use.

**MVS.** Multiple Virtual Storage.

**MVS/ESA™.** Multiple Virtual Storage/Enterprise Systems Architecture.

## N

**n-way.** The number (n) of CPs in a CPC. For example, a 6-way CPC contains six CPs.

**NCP.** Network Control Program.

**network.** A collection of data processing products connected by communications lines for exchanging information between stations.

**Network File System.** A component of z/OS that allows remote access to z/OS host processor data from workstations, personal computers, or any other system on a TCP/IP network that is using client software for the Network File System protocol.

**network job entry (NJE).** A JES2 facility that provides for the passing of selected jobs, system output data, operator commands, and messages between communicating job entry subsystems connected by binary-synchronous communication lines, channel-to-channel adapters, and shared queues.

**network operator.** (1) The person responsible for controlling the operation of a telecommunication network. (2) A VTAM application program authorized to issue network operator commands.

**next sequential instruction.** The next instruction to be executed in the absence of any branch or transfer of control.

**NIP.** nucleus initialization program

**nonpageable region.** In MVS, a subdivision of the nonpageable dynamic area that is allocated to a job step or system task that is not to be paged during execution. In a nonpageable region, each virtual address is identical to its real address. Synonymous with *V=R region*.

**nonreentrant.** A type of program that cannot be shared by multiple users.

**nonstandard labels.** Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

**nucleus.** That portion of a control program that always remains in main storage.

**nucleus initialization program (NIP).** The stage of z/OS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

**null.** Empty; having no meaning.

**O**

**object deck.** A collection of one or more control sections produced by an assembler or compiler and used as input to the linkage editor or binder. Also called object code or simply OBJ.

**object module.** A module that is the output from a language translator (such as a compiler or an assembler). An object module is in relocatable format with machine code that is not executable. Before an object module can be executed, it must be processed by the link-edit utility.

**offline.** Pertaining to equipment or devices not under control of the processor.

**offset.** The number of measuring units from an arbitrary starting point in a record, area, or control block, to some other point.

**online.** Pertaining to a user's ability to interact with a computer**.**

**operating system.** Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, I/O control, and data management. Although operating systems are predominantly software, partial hardware implementations are possible.

**operations log.** In z/OS, the operations log is a central record of communications and system problems for each system in a sysplex.

**operator commands.** Statements that system operators may use to get information, alter operations, initiate new operations, or end operations.

**operator message.** A message from an operating system directing the operator to perform a specific function, such as mounting a tape reel; or informing the operator of specific conditions within the system, such as an error condition.

**OS/390.** An earlier form of the z/OS operating system.

**output group.** A set of a job's output data sets that share output characteristics, such as class, destination, and external writer.

**output writer.** A part of the job scheduler that transcribes specified output data sets onto a system output device independently of the program that produced the data sets.

**overlay.** To write over existing data in storage.

**P**

**page.** (1) In virtual storage systems, a fixed-length block of instructions, data, or both, that can be transferred between main storage and external page storage. (2) To transfer instructions, data, or both, between main storage and external page storage.

**page fault.** In z/OS or S/390 virtual storage systems, a program interruption that occurs when a page that is marked "not in main storage" is referred to by an active page.

**pageable region.** In MVS, a subdivision of the pageable dynamic area that is allocated to a job step or a system task that can be paged during execution. Synonymous with *V=V region*.

**paging.** In z/OS, the process of transferring pages between main storage and external page storage.

**paging device.** In z/OS, a direct access storage device on which pages (and possibly other data) are stored.

**parallel processing.** The simultaneous processing of units of work by many servers. The units of work can be either transactions or subdivisions of large units of work (batch). See also *highly parallel*.

**Parallel Sysplex.** A sysplex that uses one or more coupling facilities.

**parameter.** Data item that is received by a routine.

**parmlib.** All the members in the SYS1.PARMLIB PDS that contain parameters setting the limits and controlling the behavior of z/OS.

**parmlib member.** One of the members in the SYS1.PARMLIB PDS that contain parameters setting the limits and controlling the behavior of z/OS.

**partially qualified data set name.** A data set name in which the qualifiers are not spelled out. Asterisks and percent signs are used in place of the undefined qualifiers.

**partitionable CPC.** A CPC that can be divided into two independent CPCs. See also *physical partition*, *single-image mode*, *side*.

**partitioned data set (PDS).** A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Synonymous with *program library*. Contrast with *sequential data set*.

**partitioned data set extended (PDSE).** A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

**partitioning.** The process of forming multiple configurations from one configuration.

**password.** A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it.

**patchport.** A pair of fibre adapters or couplers. Any number of patchports can participate in a fiber link. To determine the total number of patchports in a cabinet, you must add the number of patchports of each defined panel of the cabinet.

**PC.** personal computer.

**PCHID.** physical channel identifier.

**PE.** See *program error PTF*.

**Peer-to-Peer Remote Copy (PPRC).** Direct connection between DASD controller subsystems that is used primarily to provide a hot standby capability. These connections can be point-to-point from one DASD controller to another, or they can pass through switches, just as connections from CHPIDs to control units can.

**percolate.** The action taken by the condition manager when the returned value from a condition handler indicates that the handler could not handle the condition, and the condition will be transferred to the next handler.

**performance administration.** The process of defining and adjusting workload management goals and resource groups based on installation business objectives.

**permanent connection.** Permanent connections are usually made between cabinets with fiber trunk cables. Patchports that are permanently connected remain so even when they are not in use.

**permanent data set.** A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with *temporary data set*.

**PFK capability.** On a display console, indicates that program function keys are supported and were specified at system generation.

**physical channel identifier (PCHID).** The physical address of a channel path in the hardware. Logical CHPIDs have corresponding physical channels. Real I/O hardware is attached to a processor through physical channels. Channels have a physical channel identifier (PCHID) which determines the physical location of a channel in the processor. The PCHID is a three hexadecimal digit number and is assigned by the processor.

**physical partition.** Part of a CPC that operates as a CPC in its own right, with its own copy of the operating system.

**physical unit (PU).** (1) The control unit or cluster controller of an SNA terminal. (2) The part of the control unit or cluster controller that fulfills the role of a physical unit as defined by systems network architecture (SNA).

**physically partitioned (PP) mode.** The state of a processor complex when its hardware units are divided into two separate operating configurations or sides. The A-side of the processor controller controls side 0; the B-side of the processor controller controls side 1. Contrast with *single-image (SI) configuration*.

**PL/I.** A general purpose scientific/business high-level language. PL/I is a powerful procedure-oriented language especially well suited for solving complex scientific problems or running lengthy and complicated business transactions and record-keeping applications.

**platform.** The operating system environment on which a program runs

**PLPA.** pageable link pack area.

**pointer.** An address or other indication of location.

**portability.** The ability to transfer an application from one platform to another with relatively few changes to the source code.

**Portable Operating System Interface (POSIX).** Portable Operating System Interface for computing environments, an interface standard governed by the IEEE and based on UNIX. POSIX is not a product. Rather, it is an evolving family of standards describing a wide spectrum of operating system components ranging from C language and shell interfaces to system administration.

**POSIX.** Portable Operating System Interface.

**PPRC.** Peer-to-Peer Remote Copy.

**PPT.** In z/OS, the program properties table.

**preprocessor.** A routine that examines application source code for preprocessor statements that are then executed, resulting in the alteration of the source.

**preventive service.** (1) The mass installation of PTFs to avoid rediscoveries of the APARs fixed by those PTFs. (2) The SYSMODs delivered on the program update tape.

**preventive service planning (PSP).** Installation recommendations and HOLDDATA for a product or a service level. PSP information can be obtained from the IBM Support Center.

**primary key.** One or more characters within a data record used to identify the data record or control its use. A primary key must be unique.

**printer.** A device that writes output data from a system on paper or other media.

**procedure.** A set of self-contained high-level language (HLL) statements that performs a particular task and returns to the caller. Individual languages have different names for this concept of a procedure. In C, a procedure is called a function. In COBOL, a procedure is a paragraph or section that can only be performed from within the program. In PL/I, a procedure is a named block of code that can be invoked externally, usually through a a call.

**processor.** The physical processor, or machine, has a serial number, a set of channels, and a logical processor associated with it. The logical processor has a number of channel path IDs, or CHPIDs, which are the logical equivalent of channels. The logical processor may be divided into a number of logical partitions.

**processor storage.** See *main storage*.

**program error PTF (PE-PTF).** A PTF that has been found to contain an error. A PE-PTF is identified on a ++HOLD ERROR statement, along with the APAR that first reported the error.

**program fetch**. A program that prepares programs for execution by loading them at specific storage locations and readjusting each relocatable address constant.

**program library.** A partitioned data set or PDSE that always contains named members.

**program management.** The task of preparing programs for execution, storing the programs, load modules, or program objects in program libraries, and executing them on the operating system.

**program module.** The output of the binder. A collective term for program object and load module.

**program object.** All or part of a computer program in a form suitable for loading into virtual storage for execution. Program objects are stored in PDSE program libraries and have fewer restrictions than load modules. Program objects are produced by the binder.

**processor controller.** Hardware that provides support and diagnostic functions for the central processors.

**Processor Resource/Systems Manager (PR/SM).** The feature that allows the processor to use several z/OS images simultaneously and provides logical partitioning capability. See also *LPAR*.

**profile.** Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

**program function key (PFK).** A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

**program interruption.** The interruption of the execution of a program due to some event such as an operation exception, an exponent-overflow exception, or an addressing exception.

**program level.** The modification level, release, version, and fix level.

**program management.** The functions within the system that provide for establishing the necessary activation and invocation for a program to run in the applicable run-time environment when it is called.

**program mask.** In bits 20 through 23 of the program status word (PSW), a 4-bit structure that controls whether each of the fixed-point overflow, decimal overflow, exponent-overflow, and significance exceptions should cause a program interruption. The bits of the program mask can be manipulated to enable or disable the occurrence of a program interruption.

**program number.** The seven-digit code (in the format xxxx-xxx) used by IBM to identify each licensed program.

**program object.** All or part of a computer program in a form suitable for loading into virtual storage for execution. Program objects are stored in PDSE program libraries and have fewer restrictions than load modules. Program objects are produced by the binder.

**program status word (PSW).** A 64-bit structure in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program. See also *program mask*.

**program temporary fix (PTF).** A temporary solution or bypass of a problem diagnosed by IBM as resulting from a defect in a current unaltered release of the program.

**PSP.** preventive service planning.

**PSW.** program status word.

**PTF.** program temporary fix.

## Q

**QSAM.** queued sequential access method.

**qualified name.** A data set name consisting of a string of names separated by periods; for example, "TREE.FRUIT.APPLE" is a qualified name.

**qualifier.** A modifier in a qualified name other than the rightmost name. For example, “TREE” and “FRUIT” are qualifiers in “TREE.FRUIT.APPLE.”

**queue.** A line or list formed by items in a system waiting for processing.

**queued sequential access method (QSAM).** An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

## R

**RACF.** Resource Access Control Facility.

RAS. Reliability, availability, serviceability. Long-standing characteristics of mainframes.

**RDW.** record descriptor word.

**read access.** Permission to read information.

**reader.** A program that reads jobs from an input device or data base file and places them on the job queue.

**real address.** In virtual storage systems, the address of a location in main storage.

**real storage.** See *main storage*.

**reason code.** A return code that describes the reason for the failure or partial success of an attempted operation.

**receive.** In SMP/E, to read SYSMODs and other data from SMPPTFIN and SMPHOLD and store them on the global zone for subsequent SMP/E processing. This is done with the RECEIVE command.

**RECEIVE.** The SMP/E command used to read in SYSMODs and other data from SMPPTFIN and SMPHOLD.

**RECEIVE processing.** An SMP/E process necessary to install new product libraries. During this process, the code, organized as unloaded partition data sets, is loaded into temporary SMPTLIB data sets. SMP/E RECEIVE processing automatically allocates the temporary partitioned data sets that correspond to the files on the tape, and loads them from the tape.

**RECFM.** record format.

**record.** (1) A group of related data, words, or fields treated as a unit, such as one name, address, and telephone number. record. (2) A self-contained collection of information about a single object. A record is made up of a number of distinct items, called fields. A number of shell programs (for example, awk, join, and sort) are designed to process data consisting of records separated by newlines, where each record contains a number of fields separated by spaces or some other character. awk can also handle records separated by characters other than newlines. See *fixed-length record*, *variable-length record*.

**record data.** Data sets with a record-oriented structure that are accessed record by record. This data set structure is typical of data sets on z/OS and other mainframe operating systems. See also *byte stream*.

**recording format.** For a tape volume, the format of the data on the tape, for example, 18, 36, 128, or 256 tracks.

**recovery.** The process of rebuilding data after it has been damaged or destroyed, often by restoring a backup version of the data or by reapplying transactions recorded in a log.

**recovery system**. A system that is used in place of a primary application system that is no longer available for use. Data from the application system must be available for use on the recovery system. This is usually accomplished through backup and recovery techniques, or through various DASD copying techniques, such as remote copy.

**recursive routine.** A routine that can call itself or be called by another routine that it has called.

**redundant array of independent disk (RAID).** A disk subsystem architecture that combines two or more physical disk storage devices into a single logical device to achieve data redundancy.

**reenterable.** The reusability attribute that allows a program to be used concurrently by more than one task. A reenterable module can modify its own data or other shared resources, if appropriate serialization is in place to prevent interference between using tasks. See *reusability. reentrant*.

**reentrant.** The attribute of a routine or application that allows more than one user to share a single copy of a load module.

**refreshable.** The reusability attribute that allows a program to be replaced (refreshed) with a new copy without affecting its operation. A refreshable module cannot be modified by itself or any other module during execution. See *reusability*.

**register.** An internal computer component capable of storing a specified amount of data and accepting or transferring this data rapidly.

**register save area (RSA).** Area of main storage in which contents of registers are saved.

**related installation materials (RIMs).** In IBM custom-built offerings, task-oriented documentation, jobs, sample exit routines, procedures, parameters, and examples developed by IBM.

**release.** A distribution of a new product or new function and APAR fixes for an existing product. Contrast with *modification level* and *version*.

**remote copy**. A storage-based disaster recovery and workload migration function that can copy data in real time to a remote location. Two options of remote copy are available. See P*eer-to-Peer Remote Copy* and *Extended Remote Copy*.

**remote job entry (RJE).** Submission of job control statements and data from a remote terminal, causing the jobs described to be scheduled and executed as though encountered in the input stream.

**remote operations.** Operation of remote sites from a host system.

**reserved storage allocation.** The amount of central and expanded storage that you can dynamically configure online or offline to a logical partition.

**residency mode (RMODE).** The attribute of a program module that specifies whether the module, when loaded, must reside below the 16 MB virtual storage line or may reside anywhere in virtual storage.

**Resource Access Control Facility (RACF).** An IBM security manager product that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system and logging the detected accesses to protected resources.

**resource recovery services (RRS).** The z/OS system component that provides the services that a resource manager calls to protect resources. RRS is the z/OS system level sync point manager.

**RESTORE.** The SMP/E command used to remove applied SYSMODs from the target libraries.

**restore.** In SMP/E, to remove applied SYSMODs from the target libraries by use of the RESTORE command.

**restructured extended executor (REXX).** A general-purpose, procedural language for end-user personal programming, designed for ease by both casual general users and computer professionals. It is also useful for application macros. REXX includes the capability of issuing commands to the underlying operating system from these macros and procedures.

**resynchronization**. A track image copy from the primary volume to the secondary volume of only the tracks which have changed since the volume was last in duplex mode.

**return code.** A code produced by a routine to indicate its success or failure. It may be used to influence the execution of succeeding instructions or programs.

**reusability.** The attribute of a module or section that indicates the extent to which it can be reused or shared by multiple tasks within the address space. See *refreshable*, *reenterable*, and *serially reusable*.

**RIM.** related installation material.

**RJE.** remote job entry.

**RMF.** Resource Measurement Facility.

**RMODE.** residency mode.

**rollback.** The process of restoring data changed by an application to the state at its last commit point.

**routine.** (1) A program or sequence of instructions called by a program. Typically, a routine has a general purpose and is frequently used. CICS and programming languages use routines. (2) A database object that encapsulates procedural logic and SQL statements, is stored on the database server, and can be invoked from an SQL statement or by using the CALL statement. The three main classes of routines are procedures, functions, and methods. (3) In REXX, a series of instructions called with the CALL instruction or as a function. A routine can be either internal or external to a user's program. (4) A set of statements in a program that causes the system to perform an operation or a series of related operations.

**routing.** The assignment of the communications path by which a message will reach its destination.

**routing code.** A code assigned to an operator message and used to route the message to the proper console.

**RSA.** register save area.

**run.** To cause a program, utility, or other machine function to be performed.

**run time.** Any instant at which a program is being executed. Synonymous with *execution time*.

**run-time environment.** A set of resources that are used to support the execution of a program. Synonymous with *execution environment*.

## S

**SAF.** system authorization facility.

**SAP.** System Assistance Processor.

**save area.** Area of main storage in which contents of registers are saved.

**SDSF.** System Display and Search Facility.

**security administrator.** A programmer who manages, protects, and controls access to sensitive information.

**sequential data set.** (1) A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Contrast with direct data set. (2) A data set in which the contents are arranged in successive physical order and are stored as an entity. The data set can contain data, text, a program, or part of a program. Contrast with *partitioned data set (PDS)*.

**serially reusable.** The reusability attribute that allows a program to be executed by more than one task in sequence. A serially reusable module cannot be entered by a new task until the previous task has exited. See *reusability*.

**server.** (1) In a network, the computer that contains programs, data, or provides the facilities that other computers on the network can access. (2) The party that receives remote procedure calls. Contrast with *client*.

**server address space.** Any address space that does work on behalf of a transaction manager or a resource manager. For example, a server address space could be a CICS AOR, or an IMS control region.

**service.** PTFs and APAR fixes.

**service level.** The FMID, RMID, and UMID values for an element. The service level identifies the owner of the element, the last SYSMOD to replace the element, and all the SYSMODs that have updated the element since it was last replaced.

**service level agreement (SLA).** A written agreement of the information systems (IS) service to be provided to the users of a computing installation.

**service processor.** The part of a processor complex that provides for the maintenance of the complex.

**service unit.** The amount of service consumed by a work request as calculated by service definition coefficients and CPU, SRB, I/O, and storage service units.

**session.** (1) The period of time during which a user of a terminal can communicate with an interactive system; usually, the elapsed time from when a terminal is logged on to the system until it is logged off the system. (2) The period of time during which programs or devices can communicate with each other. (3) In VTAM, the period of time during which a node is connected to an application program.

**severity code.** A part of operator messages that indicates the severity of the error condition (I, E, or S).

**shared DASD option.** An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

**shared storage.** An area of storage that is the same for each virtual address space. Because it is the same space for all users, information stored there can be shared and does not have to be loaded in the user region.

**side.** One of the configurations formed by physical partitioning.

**SIGP.** signal processor.

**simultaneous peripheral operations online (spool).** The reading and writing of input and output streams on auxiliary storage devices, concurrently while a job is running, in a format convenient for later processing or output operations.

**single point of control.** The characteristic a sysplex displays when you can accomplish a given set of tasks from a single workstation, even if you need multiple IBM and vendor products to accomplish that particular set of tasks.

**single-image (SI) mode.** A mode of operation for a multiprocessor (MP) system that allows it to function as one CPC. By definition, a uniprocessor (UP) operates in single-image mode. Contrast with *physically partitioned (PP) configuration*.

**single-processor complex.** A processing environment on which only one processor (computer) accesses the spool and comprises the entire node.

**single system image.** The characteristic a product displays when multiple images of the product can be viewed and managed as one image.

**single-system sysplex.** A sysplex in which only one z/OS system is allowed to be initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system but does not provide signalling services between z/OS systems. See also *multisystem sysplex*.

**SLA.** service level agreement.

**small computer system interface (SCSI).** A standard hardware interface that enables a variety of peripheral devices to communicate with one another.

**SMF.** system management facilities.

**SMP/E.** System Modification Program/Extended.

**SMPCSI.** The SMP/E data set that contains information about the structure of a user's system as well as information needed to install the operating system on a user's system. The SMPCSI DD statement refers specifically to the CSI that contains the global zone. This is also called the master CSI.

**SMS.** Storage Management Subsystem.

**SNA.** Systems Network Architecture.

**software.** (1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library routines, manuals, circuit diagrams. Contrast with *hardware*.

**sort/merge program.** A processing program that can be used to sort or merge records in a prescribed sequence.

**source code.** The input to a compiler or assembler, written in a source language.

**source program.** A set of instructions written in a programming language that must be translated to machine language before the program can be run.

**spin data set.** A data set that is deallocated (available for printing) when it is closed. Spin off data set support is provided for output data sets just prior to the termination of the job that created the data set.

**spool.** simultaneous peripheral operations online.

**spooled data set.** A data set written on an auxiliary storage device and managed by JES.

**spooling.** The reading and writing of input and output streams on auxiliary storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

**SPUFI.** SQL Processing Using File Input.

**SQA.** system queue area.

**SQL.** Structured Query Language.

**SREL.** system release identifier.

**SRM.** system resources manager.

**SSID.** subsystem identifier.

**started task.** In z/OS, an address space that runs unattended as the result of a START command. Started tasks are generally used for critical applications. The UNIX equivalent is a daemon.

**status-display console.** An MCS console that can receive displays of system status but from which an operator cannot enter commands.

**step restart.** A restart that begins at the beginning of a job step. The restart may be automatic or deferred, where deferral involves resubmitting the job. Contrast with *checkpoint restart*.

**storage administrator.** A person in the data processing center who is responsible for defining, implementing, and maintaining storage management policies.

**storage class.** A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

**storage group.** A collection of storage volumes and attributes, defined the storage administrator. The collections can be a group of DASD volume or tape volumes, or a group of DASD, optical, or tape volumes treated as single object storage hierarchy.

**storage management.** The activities of data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These can be done either manually or by using automated processes. The Storage Management Subsystem automates these processes for you, while optimizing storage resources. See also *Storage Management Subsystem*.

**Storage Management Subsystem (SMS).** A facility used to automate and centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

**string.** A collection of one or more I/O devices. The term usually refers to a physical string of units, but may mean a collection of I/O devices which are integrated into a control unit.

**structure.** A construct used by z/OS to map and manage storage on a Coupling Facility. See *cache structure*, *list structure*, and *lock structure*.

**subchannel set.** Installation-specified structure that defines the placement of devices either relative to a channel subsystem or to an operating system.

**subpool storage.** All of the storage blocks allocated under a subpool number for a particular task.

**subsystem.** A secondary or subordinate system, or programming support, usually capable of operating independently of or asynchronously with a controlling system. Examples are CICS and IMS.

**subsystem interface (SSI).** A component that provides communication between z/OS and its job entry subsystem.

**subtask.** In the context of z/OS multitasking, a task that is initiated and terminated by a higher order task (the main task). Subtasks run the parallel functions, those portions of the program that can run independently of the main task program and each other.

**superuser.** (1) A system user who operates without restrictions. A superuser has the special rights and privileges needed to perform administrative tasks. The z/OS equivalent is a user in privileged, or supervisor, mode. (2) A system user who can pass all z/OS UNIX security checks. A superuser has the special rights and privileges needed to manage processes and files.

**superuser authority.** The unrestricted ability to access and modify any part of the operating system, usually associated with the user who manages the system.

**supervisor.** The part of z/OS that coordinates the use of resources and maintains the flow of processing unit operations.

**supervisor call (SVC).** An instruction that interrupts a program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

**support element.** A hardware unit that provides communications, monitoring, and diagnostic functions to a central processor complex (CPC).

**suspended state**. When only one of the devices in a dual copy or remote copy volume pair is being updated because of either a permanent error condition or an authorized user command. All writes to the remaining functional device are logged. This allows for automatic resynchronization of both volumes when the volume pair is reset to the active duplex state.

**SVC.** supervisor call instruction.

**SVC interruption.** An interruption caused by the execution of a supervisor call instruction, causing control to be passed to the supervisor.

**SVC routine.** A control program routine that performs or begins a control program service specified by a supervisor call instruction.

**SWA.** scheduler work area.

**swap data set.** A data set dedicated to the swapping operation.

**swapping.** A z/OS paging operation that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into main storage.

**switch.** A device that provides connectivity capability and control for attaching any two ESCON or FICON links together.

**synchronous messages.** WTO or WTOR messages issued by a z/OS system during certain recovery situations.

**sync point manager.** A function that coordinates the two-phase commit process for protected resources, so that all changes to data are either committed or backed out. In z/OS, RRS can act as the system level sync point manager. A sync point manager is also known as a transaction manager, sync point coordinator, or a commit coordinator.

**syntax.** The rules governing the structure of a programming language and the construction of a statement in a programming language.

**SYSIN.** A system input stream; also, the name used as the data definition name of a data set in the input stream.

**SYSLIB.** (1) A subentry used to identify the target library in which an element is installed. (2) A concatenation of macro libraries to be used by the assembler. (3) A set of routines used by the link-edit utility to resolve unresolved external references.

**SYSLOG.** system log.

**SYSMOD.** system modification.

**SYSOUT.** A system output stream; also, an indicator used in data definition statements to signify that a data set is to be written on a system output unit.

**SYSOUT class.** A category of output with specific characteristics and written on a specific output device. Each system has its own set of SYSOUT classes, designated by a character from A to Z, a number from 0 to 9, or a *.

**sysplex.** A set of z/OS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process client workloads. See also Parallel Sysplex.

**sysplex couple data set.** A couple data set that contains sysplex-wide data about systems, groups, and members that use XCF services. All z/OS systems in a sysplex must have connectivity to the sysplex couple data set. See also *couple data set*.

**Sysplex Timer.** An IBM unit that synchronizes the time-of-day (TOD) clocks in multiple processors or processor sides.

**SYSRES.** system residence disk.

**system.** The combination of a configuration (hardware) and the operating system (software). Often referred to simply as the z/OS system.

**system abend.** An abend caused by the operating system's inability to process a routine; may be caused by errors in the logic of the source routine.

**system console.** In z/OS, a console attached to the processor controller used to initialize a z/OS system.

**system control element (SCE).** Hardware that handles the transfer of data and control information associated with storage requests between the elements of the processor.

**system data.** The data sets required by z/OS or its subsystems for initialization.

**system library.** A collection of data sets or files in which the parts of an operating system are stored.

**system-managed data set.** A data set that has been assigned a storage class.

**system-managed storage.** Storage managed by the Storage Management Subsystem (SMS) of z/OS.

**system management facilities (SMF).** A z/OS component that provides the means for gathering and recording information for evaluating system usage.

**system modification (SYSMOD).** The input data to SMP/E that defines the introduction, replacement, or updating of elements in the operating system and associated distribution libraries to be installed. A system modification is defined by a set of MCS.

**System Modification Program Extended (SMP/E).** An IBM program product, or an element of OS/390 or z/OS, used to install software and software changes on z/OS systems. SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets. SMP/E is the primary means of controlling changes to the z/OS operating system.

**Systems Network Architecture (SNA).** A description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of networks.

**system queue area (SQA).** In z/OS, an area of virtual storage reserved for system-related control blocks.

## T

**tape volume.** Storage space on tape, identified by a volume label, which contains data sets or objects and available free space. A tape volume is the recording space on a single tape cartridge or reel. See also *volume*.

**target library.** In SMP/E, a collection of data sets in which the various parts of an operating system are stored. These data sets are sometimes called system libraries.

**target zone.** In SMP/E, a collection of VSAM records describing the target system macros, modules, assemblies, load modules, source modules, and libraries copied from DLIBs during system generation, and the SYSMODs applied to the target system.

**task.** In a multiprogramming or multiprocessing environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer.

**task control block (TCB).** A data structure that contains information and pointers associated with the task in process.

**TCB.** task control block.

**TCP/IP.** Transmission Control Protocol/Internet Protocol.

**temporary data set.** A data set that is created and deleted in the same job.

**terminal.** A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

**terminal owning region (TOR).** A CICS region devoted to managing the terminal network.

**TGTLIB.** target library.

**tightly coupled.** Multiple CPs that share storage and are controlled by a single copy of z/OS. See also *loosely coupled, tightly coupled multiprocessor*.

**tightly coupled multiprocessing.** Two computing systems operating simultaneously under one control program while sharing resources.

**tightly coupled multiprocessor.** Any CPU with multiple CPs.

**Time Sharing Option/Extensions (TSO/E).** The facility in z/OS that allows users to interactively share computer time and resources.

**timeout**. The time in seconds that the storage control remains in a “long busy” condition before physical sessions are ended.

**TLIB.** target library.

**transaction. A** unit of work performed by one or more transaction programs, involving a specific set of input data and initiating a specific process or job.

**Transmission Control Protocol/Internet Protocol (TCP/IP).** A hardware independent communication protocol used between physically separated computers. It was designed to facilitate communication between computers located on different physical networks.

**Transport Layer Security (TLS).** A protocol that provides communications privacy over the Internet.

**TRK.** A subparameter of the SPACE parameter in a DD statement. It specifies that space is to be allocated by tracks.

**trunk cable.** Cables used to make permanent connections between cabinets and which remain in place even when not in use.

**TSO.** Time-sharing option. See *Time Sharing Option/ Extensions (TSO/E)*.

**TSO/E.** Time Sharing Option/Extensions.

## U

**UCB.** unit control block.

**UCLIN.** In SMP/E, the command used to initiate changes to SMP/E data sets. Actual changes are made by subsequent UCL statements.

**UIM.** unit information module.

**Unicode Standard.** A universal character encoding standard that supports the interchange, processing, and display of text that is written in any of the languages of the modern world. It can also support many classical and historical texts and is continually being expanded.

**uniprocessor (UP).** A processor complex that has one central processor.

**unit of recovery (UR).** A set of changes on one node that is committed or backed out as part of an ACID transaction. A UR is implicitly started the first time a resource manager touches a protected resource on a node. A UR ends when the two-phase commit process for the ACID transaction changing it completes.

**UNIX.** See *z/OS UNIX System Services*.

**UNIX file system.** A section of the UNIX file tree that is physically contained on a single device or disk partition and that can be separately mounted, dismounted, and administered. See also *hierarchical file system*.

**UNLOAD.** The SMP/E command used to copy data out of SMP/E data set entries in the form of UCL statements.

**unload.** In SMP/E, to copy data out of SMP/E data set entries in the form of UCL statements, by use of the UNLOAD command.

**unused cable.** Physical cables that have been recently disconnected, but not yet placed in inventory.

**upwardly compatible.** The ability for applications to continue to run on later releases of z/OS, without the need to recompile or relink.

**user abend.** A request made by user code to the operating system to abnormally terminate a routine. Contrast with *system abend*.

**user catalog.** An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It also lessens the contention for the master catalog and facilitates volume portability.

**user exit.** A routine that takes control at a specific point in an application. User exits are often used to provide additional initialization and termination functions.

**user ID.** user identification.

**user identification (user ID).** A 1-8 character symbol identifying a system user.

**user modification (USERMOD).** A change constructed by a user to modify an existing function, add to an existing function, or add a user-defined function. USERMODs are identified to SMP/E by the ++USERMOD statement.

**USERMOD.** user modification.

## V

**V=R region.** Synonymous with *nonpageable region*.

**V=V region.** Synonymous with *pageable region*.

**variable-length record.** A record having a length independent of the length of other records with which it is logically or physically associated. Contrast with *fixed-length record*.

**VB.** Variable blocked.

**vendor.** A person or company that provides a service or product to another person or company.

**version.** A separate licensed program that is based on an existing licensed program and that usually has significant new code or new functions. Contrast with *release* and *modification level*.

**VIO.** virtual input/output.

**virtual address space.** In virtual storage systems, the virtual storage assigned to a job, terminal user, or system task. See also *address space*.

**virtual input/output (VIO).** The allocation of data sets that exist in paging storage only.

**Virtual Storage Access Method (VSAM).** An access method for direct or sequential processing of fixed-length and varying-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

**virtual storage.** (1) The storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations. (2) An addressing scheme that allows external disk storage to appear as main storage.

**virtual telecommunications access method (VTAM).** A set of programs that maintain control of the communication between terminals and application programs running under z/OS.

**VM.** Virtual Machine.

**VOLSER.** volume serial number.

**volume.** (1) The storage space on DASD, tape or optical devices, which is identified by a volume label. (2) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (3) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape or a disk pack.

**volume backup.** Backup of an entire volume to protect against the loss of the volume.

**volume serial number.** A number in a volume label that is assigned when a volume is prepared for use in the system.

**volume table of contents (VTOC).** A table on a direct access storage device (DASD) volume that describes the location, size, and other characteristics of each data set on the volume.

**VPN.** virtual private network.

**VSAM.** virtual storage access method.

**VTAM.** Virtual Telecommunications Access Method.

**VTOC.** volume table of contents.

## W

**WAP.** wireless access point.

**wait state.** Synonymous with *waiting time*.

**waiting time.** (1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

**wild carding.** The use of an asterisk (*) as a multiple character replacement in classification rules.

**WLM.** workload manager. A subsystem that optimizes z/OS application throughput based on your requirements.

**workload.** A group of work to be tracked, managed and reported as a unit.

**work request.** A piece of work, such as a request for service, a batch job, an APPC, CICS, or IMS transaction, a TSO LOGON, or a TSO command.

**wrap mode.** The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

**write-to-operator (WTO) message.** A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

**write-to-operator-with-reply (WTOR) message.** A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

**WTO.** write-to-operator.

**WTOR.** write-to-operator-with-reply.

## X

**XA.** Extended Architecture.

**XCF.** cross-system coupling facility.

## Z

**zAAP.** zSeries Application Assist Processor.

**z/Architecture.** An IBM architecture for mainframe computers and peripherals. The zSeries family of servers uses the z/Architecture.

**zFS.** zSeries file system.

**z/OS.** A widely used operating system for IBM mainframe computers that uses 64-bit main storage.

**z/OS Language Environment.** An IBM software product that provides a common run-time environment and common run-time services for conforming high-level language compilers.

**z/OS UNIX System Services (z/OS UNIX).** z/OS services that support a UNIX-like environment. Users can switch between the traditional TSO/E interface and the shell interface. UNIX-skilled users can interact with the system, using a familiar set of standard commands and utilities. z/OS-skilled users can interact with the system, using familiar TSO/E commands and interactive menus to create and manage hierarchical file system files and to copy data back and forth between z/OS data sets and files. Application programmers and users have both sets of interfaces to choose from and, by making appropriate trade-offs, can choose to mix these interfaces.

**zSeries Application Assist Processor (zAAP).** A specialized processing assist unit configured for running Java programming on selected zSeries machines.

**zSeries File System (zFS).** A z/OS UNIX file system that stores files in VSAM linear data sets.

## Numerics

**3270 pass-through mode.** A mode that lets a program running from the z/OS shell send and receive a 3270 data stream or issue TSO/E commands.

# Notices

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ™
eServer™
ibm.com®
z/Architecture™
z/OS®
z/VSE™
zSeries®
z9™
AIX®
Chipkill™
CICS/ESA®
CICS®
CICSPlex®
DB2®
DFS™
DFSMSdfp™
DFSMSdss™
DFSMShsm™
DFSORT™
DS6000™
DS8000™
Electronic Service Agent™
Enterprise Systems Architecture/390®
Extended Services®
ESCON®
FlashCopy®
FICON®
Geographically Dispersed Parallel Sysplex™
GDPS®
HiperSockets™
IBM®
IMS™
IMS/ESA®
Language Environment®
MQSeries®
MVS™
MVS/ESA™
NetView®
Nways®
OS/390®
Parallel Sysplex®
Processor Resource/Systems Manager™
PR/SM™
Redbooks™
Resource Link™
RACF®
RMF™
S/360™
S/390®
Sysplex Timer®
System p™
System z™
System z9™
System Storage™
System/360™
System/370™
Tivoli®
TotalStorage®
VisualAge®
VTAM®
WebSphere®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Java, RSM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# 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 about ordering these publications, see "How to get IBM Redbooks" on page 212. Note that some of the documents referenced here may be available in softcopy only.

- *GDPS Family - An Introduction to Concepts and Capabilities,* SG24-6374
- *IBM System z9 and eServer zSeries Connectivity Handbook,* SG24-5444
- *IBM zSeries 990 Technical Guide,* SG24-6947
- *IBM System z9 Business Class Technical Introduction,* SG24-7241
- *IBM System z9 Enterprise Class Technical Guide,* SG24-7124
- *IBM System z9 109 Technical Guide,* SG24-7124
- *IBM System z9 and eServer zSeries Connectivity Handbook,* SG24-5444
- *OS/390 Workload Manager Implementation and Exploitation,* SG24-5326
- *ABCs of z/OS System Programming Volume 3 - Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, System-managed storage, catalogs, and DFSMStvs,* SG24-6983
- *ABCs of z/OS System Programming Volume 5 - Base and Parallel Sysplex, GRS, RRS; System Logger, z/OS system operations; ARM, GDPS, zSeries availability,* SG24-6985
- *ABCs of z/OS System Programming Volume 10 - Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and DS8000,* SG24-6990
- *ABCs of z/OS System Programming Volume 11 - Capacity planning, performance management, WLM, RMF, SMF,* SG24-6327
- *Introduction to the New Mainframe: z/OS Basics,* SG24-6366
- *Introduction to the New Mainframe: Security,* SG24-6776
- *The IBM TotalStorage DS8000 Series: Concepts and Architecture,* SG24-6452
- *DFSMShsm Fast Replication Technical Guide,* SG24-7069

- *The IBM TotalStorage® DS6000™ Series: Copy Services with IBM eserver zSeries,* SG24-6782
- *A Practical Guide to the IBM Autonomic Computing Toolkit,* SG24-6635
- *Problem Determination Using Self-Managing Autonomic Technology,* SG24-6665
- *S/390 Time Management and IBM 9037 Sysplex Timer,* SG24-2070

## IBM white papers

These documents can be found on the Web at:

http://www.ibm.com/support/techdocs/

This site provides access to the Technical Sales Support organization's technical information databases. It gives you access to the most current installation, planning and technical support information available from IBM pre-sales support, and is constantly updated. You can browse or search these databases by date, document number, product, platform, keywords, and so on.

- *Autonomic Computing: An architectural blueprint for autonomic computing*

  http://www.ibm.com/autonomic

## Other publications

These publications are also relevant as further information sources:

- *z/OS V1R1.0 Parallel Sysplex Application Migration,* SA22-7662
- *z/OS V1R1.0 Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism,* SA22-7661
- *z/OS V1R7.0 MVS Setting Up a Sysplex,* SA22-7625
- *Large Systems Performance Reference,* SC28-1187
- *Security on z/OS: Comprehensive, current, and flexible,* Guski et al.

  IBM Systems Journal, Vol.40, No.3, 2001, pp.696-720, G321-0142

  http://www.research.ibm.com/journal/sj/403/guski.html
- *IBM eServer zSeries Security Leadership for the On Demand World,* GM13-0644

  http://www.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/Security_TL.pdf

- *IBM eServer z990,* IBM Journal Research and Development, Vol. 48, No. 3/4, 2004
- *Cluster architectures and S/390 Parallel Sysplex scalability* by G. M. King, D. M. Dias, and P. S. Yu, IBM System Journal Volume 36, Number 2, 1997
- *z/OS: MVS System Management Facilities (SMF)*, SA22-7630
- *IBM SMP/E for z/OS User's Guide*, SA22-7773

## Online resources

These Web sites and URLs are also relevant as further information sources:

- IBM terminology

  http://www.ibm.com/ibm/terminology
- General information about IBM System z

  http://www.ibm.com/systems/z
- IBM Geographically Dispersed Parallel Sysplex

  http://www.ibm.com/servers/eserver/zseries/gdps
- IBM Storage Solutions: Overview - IBM System Storage

  http://www.ibm.com/servers/storage/solutions/
- Parallel Sysplex home page

  http://www.ibm.com/servers/eserver/zseries/pso/
- z/OS basic skills information center

  http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp
- General mainframe information

  http://www.mainframe.typepad.com/
- Scaling - us or out

  http://www.redbooks.ibm.com/abstracts/redp0436.html
- z/OS Workload Manager - how it works and how to use it

  http://www.ibm.com/servers/eserver/zseries/zos/wlm/pdf/zWLM.pdf
- The Autonomic Computing home page

  http://www.ibm.com/autonomic

## 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

Introduction to the New Mainframe: Large-Scale Commercial Computing

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

IBM®

# Introduction to the New Mainframe: Large-Scale Commercial Computing

Redbooks

**Learn why mainframes process much of the world's commercial workload**

**Find out why mainframes are so reliable**

**Understand mainframe popularity**

Today, mainframe computers play a central role in the daily operations of most of the world's largest corporations. While other forms of computing are used in business in various capacities, the mainframe occupies a prominent place in today's e-business environment. In banking, finance, health care, insurance, utilities, government, and a multitude of other public and private enterprises, the mainframe computer continues to provide the foundation of large-scale computing to modern business.

The reasons for mainframe use are many, but generally fall into one or more of the following categories: capacity, scalability, integrity and security, availability, access to large amounts of data, system management, and autonomic capabilities. This IBM Redbook is designed for readers who already possess a basic knowledge of mainframe computing, but need a clearer understanding of how these concepts relate to mainframe planning, implementation, and operation. For readers who need more introductory information about mainframe concepts, usage, and architecture, we recommend that you complete *Introduction to the New Mainframe: z/OS Basics*, SG24-6366, prior to starting this book. And for more detailed information about z/OS programming topics, refer to the 11-volume IBM Redbook series *ABCs of z/OS System Programming*.

SG24-7175-00 ISBN 0738496626