z/OS Basic Skills Information Center

Mainframe concepts

z/OS Basic Skills Information Center

Mainframe concepts

Note

Before using this information and the product it supports, read the information in "Notices" on page 45.

This edition applies to z/OS (product number 5694-A01).

We appreciate your comments about this publication. Comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Send your comments through this Web site: http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp?topic=/com.ibm.zcontact.doc/webqs.html

Contents

Introduction to the mainframe

As a technical professional, you need a general understanding of the mainframe computer and its place in today's information technology (IT) organization. **Mainframe concepts** is a collection of articles that provides a general introduction to mainframe concepts, usage, and architecture.

Explore the reasons why public and private enterprises throughout the world rely on the mainframe as the foundation of large-scale computing; discover what types of workloads are commonly associated with the mainframe; and marvel at the unique manner in which this work is processed by a widely used mainframe operating system— z/OS®. Also learn a little mainframe history, as well as current roles and responsibilities of the skilled staff of a mainframe IT organization.

For optimal learning, readers are assumed to have successfully completed an introductory course in computer system concepts, such as computer organization and architecture, operating systems, data management, or data communications. They should also have successfully completed courses in one or more programming languages, and be PC literate.

Readers who will benefit from this collection of articles include data processing professionals who have experience with non-mainframe platforms, or who are familiar with some aspects of the mainframe but want to become knowledgeable with other facilities and benefits of the mainframe environment.

Chapter 1. The value of the mainframe today

Today, mainframe computers play a central role in the daily operations of most of the world's largest corporations, including many Fortune 1000 companies. While other forms of computing are used extensively in business in various capacities, the mainframe occupies a coveted 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 form the foundation of modern business.

The long-term success of mainframe computers is without precedent in the information technology (IT) field. Periodic upheavals shake world economies and continuous— often wrenching— change in the Information Age has claimed many once-compelling innovations as victims in the relentless march of progress. As emerging technologies leap into the public eye, many are just as suddenly rendered obsolete by some even newer advancement. Yet today, as in every decade since the 1960s, mainframe computers and the mainframe **style** of computing dominate the landscape of large-scale business computing.

Why has this one form of computing taken hold so strongly among so many of the world's corporations? In this section, we look at the reasons why mainframe computers continue to be the popular choice for large-scale business computing. 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 gradually moved from a physical description of IBM's larger computers to the categorization of a style of computing. One defining characteristic of the mainframe has been a continuing compatibility that spans decades.

The S/360: A turning point in mainframe history

When did mainframe computers come into being? Mainframe development occurred in a series of generations starting in the 1950s. In those days, mainframe computers were not just the largest computers; they were the only computers and few businesses could afford them.

First generation systems, such as the IBM 705 in 1954 and the IBM 1401 in 1959, were a far cry from the enormously powerful machines that were to follow, but they clearly had characteristics of mainframe computers. These computers were sold as business machines and served then— as now— as the central data repository in a corporation's data processing center.

In the 1960s, the course of computing history changed dramatically when mainframe manufacturers began to standardize the hardware and software they offered to customers. The introduction of the IBM System/360 (or S/360™) in 1964 signaled the start of the third generation: the first general purpose computers. Earlier systems such as the 1401 were dedicated as either commercial or scientific computers. The revolutionary S/360 could perform both types of computing, as long as the customer, a software company, or a consultant provided the programs to do so. In fact, the name S/360 refers to the architecture's wide scope: 360 degrees to cover the entire circle of possible uses.

The S/360 was also the first of these computers to use *microcode* to implement many of its machine instructions, as opposed to having all of its machine instructions hard-wired into its circuitry. Microcode (or *firmware*, as it is sometimes called) consists of stored microinstructions, not available to users, that provide a functional layer between hardware and software. The advantage of microcoding is flexibility, where any correction or new function can be implemented by just changing the existing microcode, rather than replacing the computer.

With standardized mainframe computers to run their workloads, customers could, in turn, write business applications that didn't need specialized hardware or software. Moreover, customers were free to upgrade to newer and more powerful processors without concern for compatibility problems with their existing applications. The first wave of customer business applications were mostly written in Assembler, COBOL, FORTRAN, or PL/1, and a substantial number of these older programs are still in use today.

In the decades since the 1960s, mainframe computers have steadily grown to achieve enormous processing capabilities. **The New Mainframe** has an unrivaled ability to serve end users by the tens of thousands, manage petabytes of data, and reconfigure hardware and software resources to accommodate changes in workload— all from a single point of control.

Mainframe architecture: Secure, compatible, and still evolving

An *architecture* is a set of defined terms and rules that are used as instructions to build products. Each generation of mainframe computers has included improvements in architecture, while remaining the most stable, secure, and compatible of all computing platforms.

In computer science, an architecture describes the organizational structure of a system. An architecture can be recursively decomposed into 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 1960s and became known as "Big Iron" (in contrast to smaller departmental systems), each new generation of mainframe computers has included improvements in one or more of the following areas of the architecture:
- More and faster processors
- More physical memory and greater memory addressing capability
- Dynamic capabilities for upgrading both hardware and software
- Increased automation of hardware error checking and recovery
- Enhanced devices for input/output (I/O) and more and faster paths (channels) between I/O devices and processors
- More sophisticated I/O attachments, such as LAN adapters with extensive inboard processing
- A greater ability to divide the resources of one machine into multiple, logically independent and isolated systems, each running its own operating system
- Advanced 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 customer workloads, yet continue to run applications that were written in the 1970s or earlier.

How can a technology change so much, yet remain so stable? It can by evolving to meet new challenges. In the early 1990s, the client/server model of computing, with its distributed nodes of less powerful computers, emerged to challenge the dominance of mainframe computers. Industry pundits predicted a swift end for the mainframe computer and called it a “dinosaur.” In response, mainframe designers did what they have always done when confronted with changing times and a growing list of user requirements: They designed new mainframe computers to meet the demand. With a tip of the hat to the dinosaur naysayers, IBM, as the leading manufacturer of mainframe computers, code-named its then-current machine **T-Rex**.

With the expanded functions and added tiers of data processing capabilities such as Web-serving, autonomics, disaster recovery, and grid computing, the mainframe computer is poised to ride the next wave of growth in the IT industry. Mainframe manufacturers such as IBM are once again reporting annual sales growth in the double digits.

And the evolution continues. While the mainframe computer has retained its traditional, central role in the 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.

As the image of the mainframe computer continues to evolve, you might ask: Is the mainframe computer a self-contained computing environment, or is it one part of the puzzle in distributed computing? The answer is that **The New Mainframe** is both— a self-contained processing center, powerful enough to process the largest and most diverse workloads in one secure “footprint”, and one that is just as effective when implemented as the primary server in a corporation’s distributed server farm. In effect, the mainframe computer is the definitive server in the client/server model of computing.

Mainframes in our midst: You use one every day

Despite the predominance of mainframes in the business world, these machines are largely invisible to the general public, the academic community, and indeed many experienced IT professionals. Instead, other forms of computing attract more attention, at least in terms of visibility and public awareness. That this is so is perhaps not surprising. After all, who among us needs direct access to a mainframe? And, if we did, where would we find one to access? The truth, however, is that we are **all** mainframe users, whether we realize it or not.

Most of us with some personal computer (PC) literacy and sufficient funds can purchase a notebook computer and quickly put it to good use— running software, browsing Web sites, and perhaps even writing papers for college professors to grade. With somewhat greater effort and technical prowess, we can delve more deeply into the various facilities of a typical Intel-based workstation and learn its capabilities through direct, hands-on experience— with or without help from any of a multitude of readily available information sources in print or on the Web.

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

Furthermore, in a typical customer installation, the mainframe shares space with many other hardware devices: external storage devices, hardware network routers, channel controllers, and automated tape library ″robots,″ to name a few. **The New Mainframe** is physically no larger than many of these devices and generally does not stand out from the crowd of peripheral devices.

So, how can we explore the mainframe’s capabilities in the real world? How can we learn to interact with the 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.

Would we even know a mainframe if we saw one, given that these machines have evolved to flourish in the twenty-first century IT organization? What we need is an experienced guide to lead us on a **dinosaur safari**, which is where the z/OS Basic Skills Information Center comes in!

What is a mainframe? It’s a *style* of computing

Although the term *mainframe* first described the physical characteristics of early systems, today it can best be used to describe a **style** of operation, applications, and operating system facilities.

Today, computer manufacturers don’t always use the term *mainframe* to refer to mainframe computers. Instead, most have taken to calling any commercial-use computer— large or small— a *server*, with the mainframe simply being the largest type of server in use today. IBM, for example, refers to its latest mainframe as the IBM System z9® server. We use the term mainframe in this section to mean computers that can support thousands of applications and input/output devices to simultaneously serve thousands of users.

Servers are proliferating. A business might have a large server collection that includes transaction servers, database servers, e-mail servers and Web servers. Very large collections of servers are sometimes called *server farms* (in fact, some data centers cover areas measured in **acres**). The hardware required to perform a server function can range from little more than a cluster of rack-mounted personal computers to the most powerful mainframes manufactured today.

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, as opposed to a distributed form of computing. Centralizing the data in a single mainframe repository saves customers from having to manage updates to more than one copy of their business data, which increases the likelihood that the data is current.

The distinction between centralized and distributed computing, however, is rapidly blurring as smaller machines continue to gain in processing power and mainframes become ever more flexible and multipurpose. Market pressures require that today’s businesses continually reevaluate their IT strategies to find better ways of supporting a changing marketplace. As a result, mainframes are now frequently used in combination with networks of smaller servers in a multitude of configurations. The ability to dynamically reconfigure a mainframe's hardware and software resources (such as processors, memory, and device connections), while applications continue running, further underscores the flexible, evolving nature of the modern mainframe.

While mainframe hardware has become harder to pigeon-hole, so, too, have the operating systems that run on mainframes. Years ago, in fact, the terms defined each other: a mainframe was any hardware system that ran a major IBM operating system. This meaning has been blurred in recent years because these operating systems can be run on very small systems.

Computer manufacturers and IT professionals often use the term *platform* to refer to the hardware and software that are associated with a particular computer architecture. For example, a mainframe computer and its operating system (and their predecessors) are considered a platform; UNIX® on a Reduced Instruction Set Computer (RISC) system is considered a platform somewhat independently of exactly which RISC machine is involved; personal computers can be seen as several different platforms, depending on which operating system is being used.

So, let's return to our question now: "What is a mainframe?" Today, the term *mainframe* can best be used to describe a **style** of operation, applications, and operating system facilities. To start with a working definition, **a mainframe is what businesses use to host the commercial databases, transaction servers, and applications that require a greater degree of security and availability than is commonly found on smaller-scale machines**.

Early mainframe systems were housed in enormous, room-sized metal boxes or frames, which is probably how the term *mainframe* originated. The early mainframe required large amounts of electrical power and air-conditioning, and the room was filled mainly with I/O devices. Also, a typical customer site 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 3000 square meters). Some installations were even larger than this.

Starting around 1990, mainframe processors and most of their I/O devices became physically smaller, while their functionality and capacity continued to grow. Mainframe systems today are much smaller than earlier systems— about the size of a large refrigerator.

In some cases, it is now possible to run a mainframe operating system on a PC that emulates a mainframe. 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 detailed *operations procedure books* and highly organized procedures for backups, recovery, training, and disaster recovery at an alternative location.
- Hardware and operating systems that routinely work with hundreds or thousands of simultaneous I/O operations.
- Clustering technologies that allow the customer 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 systems to be added or removed as needed, while applications continue to run. This flexibility allows mainframe customers to introduce new applications, or discontinue the use of 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, with database access controlled at the **record level**.

As the performance and cost of such hardware resources as central processing unit (CPU) power and external storage media improve, and the number and types of devices that can be attached to the CPU increase, the operating system software can more fully take advantage of the improved hardware. Also, continuing improvements in software functionality help drive the development of each new generation of hardware systems.

Who uses mainframes and why do they do it?

So, who uses mainframes? Just about **everyone** has used a mainframe computer at one point or another. If you ever used an automated teller machine (ATM) to interact with your bank account, you used a mainframe.

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 extensively in business in various capacities, the mainframe occupies a coveted 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 be the foundation of modern business.

Until the mid-1990s, mainframes provided the **only** acceptable means of handling the data processing requirements of a large business. These requirements were then (and are often now) based on running large and complex programs, 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 that have been made since the introduction of the System/360 in 1964. No other computer architecture can claim as much continuous, evolutionary improvement, while maintaining compatibility with previous releases.

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

One common impression of a mainframe's user interface is the 80x24-character "green screen" terminal, named for the old cathode ray tube (CRT) monitors from years ago that glowed green. In reality, mainframe interfaces today look much the same as those for personal computers or UNIX systems. When a business application is accessed through a Web browser, there is often a mainframe computer performing crucial functions behind the scenes.

Many of today’s busiest Web sites store their production databases on a mainframe host. New mainframe hardware and software products 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 customer 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 (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 communication

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

Mainframe strengths: Reliability, availability, and serviceability

The *reliability*, *availability*, and *serviceability* (or “RAS”) of a computer system have always been important factors in data processing. When we say that a particular computer system “exhibits RAS characteristics,” we mean that its design 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 characteristics of hardware and software that are prized by mainframe users. The terms are defined as follows:

Reliability
The system’s hardware components have extensive self-checking and self-recovery capabilities. The system’s software reliability is a result of 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 term applies to hardware recovery (the automatic replacing of failed elements with spares) and software recovery (the layers of error recovery that are provided by the operating system).

Serviceability
The system can determine why a failure occurred. This capability allows for the replacement of hardware and software elements while impacting as little of the operational system as possible. This term 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; that is, easy to fix within a relatively short period of time.

Mean time between failure (MTBF) refers to the availability of a computer system. **The New Mainframe** and its associated software have evolved to the point that customers often experience months or even **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 in 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 tolerance*. 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.

Mainframe strengths: Security

One of a firm's most valuable resources is its data: Customer lists, accounting data, employee information, and so on. This critical data needs to be securely managed and controlled, and, simultaneously, made available to those users authorized to see it. The mainframe computer has extensive capabilities to simultaneously share, but still protect, the firm's data among multiple users.

In an IT environment, data security is defined as protection against unauthorized access, transfer, modification, or destruction, whether accidental or intentional. To protect data and to maintain the resources necessary to meet the security objectives, customers typically add a sophisticated security manager product to their mainframe operating system. The customer'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. **The New Mainframe** can provide a very secure system for processing large numbers of heterogeneous applications that access critical data.

Mainframe strengths: Scalability

It has been said that the only constant is change. Nowhere is that statement more true than in the IT industry. In business, positive results can often trigger a growth in IT infrastructure to cope with increased demand. The degree to which the IT organization can add capacity without disruption to normal business processes or without incurring excessive overhead (nonproductive processing) is largely determined by the *scalability* of the particular computing platform.

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, customers, 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 the cost to run it.

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*.

Mainframe strength: Continuing compatibility

Mainframe customers tend to have a very large financial investment in their applications and data. Some applications have been developed and refined over decades. Some applications were written many years ago, while others may have been written "yesterday." The ability of an application to work in the system or its ability to work with other devices or programs is called *compatibility*.

The need to support applications of varying ages imposes a strict compatibility demand on mainframe hardware and software, which have been upgraded many times since the first System/360 mainframe computer was shipped in 1964. Applications **must** continue to work properly. Thus, much of the design work for new hardware and system software revolves around this compatibility requirement.

The overriding need for compatibility is also the primary reason why many aspects of the system work as they do, for example, the syntax restrictions of the job control language (JCL) that is used to control batch jobs. Any new design enhancements made to JCL must preserve compatibility with older jobs so that they can continue to run without modification. The desire and need for continuing compatibility is one of the defining characteristics of mainframe computing.

Absolute compatibility across decades of changes and enhancements is not possible, of course, but the designers of mainframe hardware and software make it a top priority. When an incompatibility is unavoidable, the designers typically warn users **at least a year** in advance that software changes might be needed.

Mainframe workloads: Batch and online transaction processing

Most mainframe workloads fall into one of two categories: Batch processing or online transaction processing, which includes Web-based applications.

One key advantage of mainframe systems is their 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 perform end-of-quarter processing and produce reports that are necessary to customers (for example, quarterly stock statements or pension statements) or to the government (for example, financial results). With mainframe systems, retail stores can generate and consolidate nightly sales reports for review by regional sales managers. The applications that produce these statements are *batch* applications, which are illustrated at the top of Figure 1 on page 10.

In contrast to batch processing, transaction processing occurs interactively with the end user. This interaction is outlined at the bottom of Figure 1. Typically, mainframes serve a vast number of *transaction systems*. These systems 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.

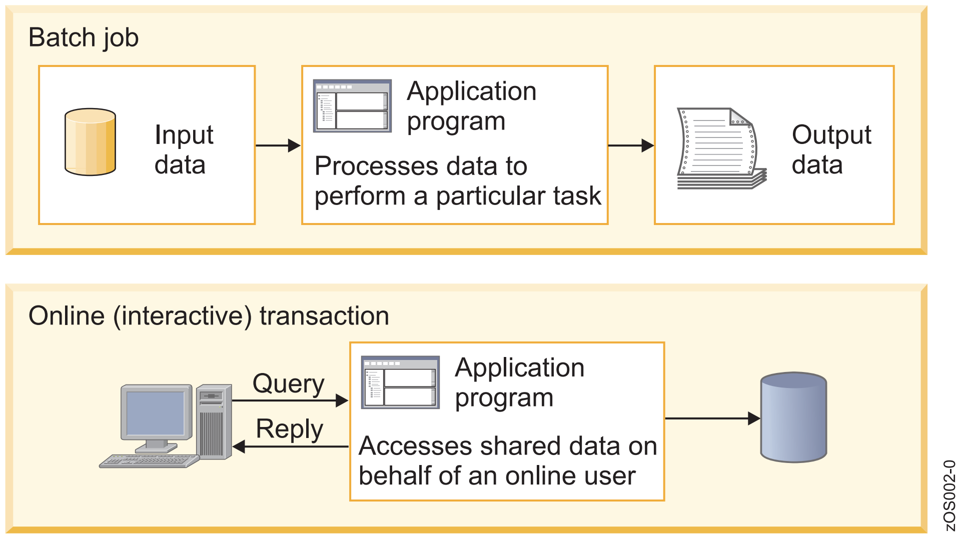
Batch job
Input data
Application program
Processes data to perform a particular task
Output data
Online (interactive) transaction
Query
Reply
Application program
Accesses shared data on behalf of an online user
zOS002-0

Figure 1. Typical mainframe workloads

Mainframes working after hours: Batch processing

Batch applications are processed on the mainframe without user interaction. A *batch job* is submitted on the computer; the job reads and processes data in bulk— perhaps terabytes of data— and produces output, such as customer billing statements. An equivalent concept can be found in a UNIX script file or a Windows® command file, but a z/OS batch job might process millions of records.

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

- Sufficient data storage
- Available processor capacity, or *cycles*
- Sysplex-wide management of system resources and job scheduling

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

Batch processes typically have the following characteristics:

- Large amounts of input data are processed and stored (perhaps terabytes or more), large numbers of records are accessed, and a large volume of output is produced.
- Immediate response time is usually not a requirement. However, batch jobs often must complete within a *batch window*, a period of less-intensive online activity, as prescribed by a *service level agreement* (SLA).
- Information is generated about large numbers of users or data entities (for example, customer 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 backups, processing of daily orders, and updating of inventories are common examples.

In Figure 2, consider the following elements at work in the scheduled batch process:

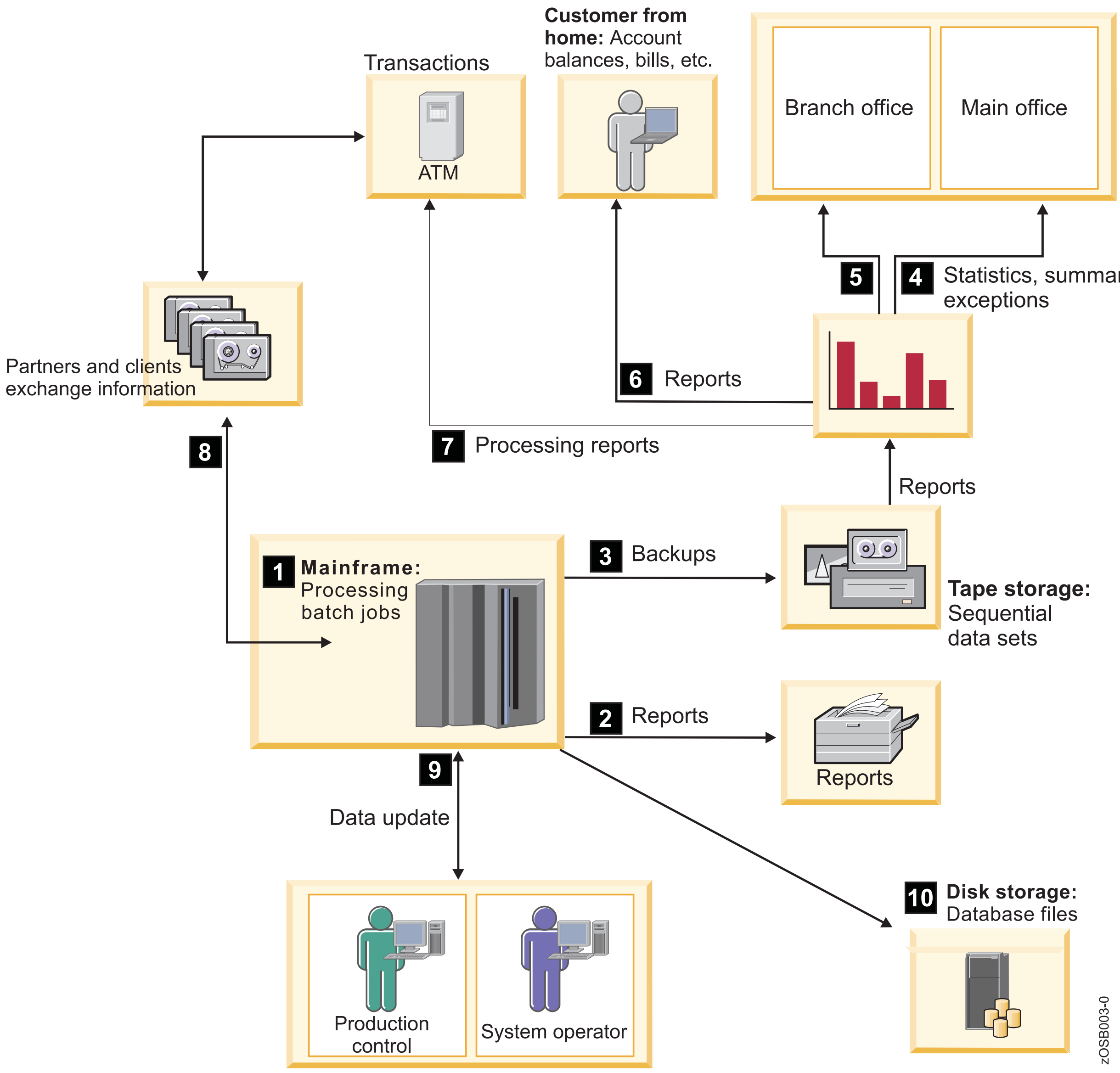
Transactions
ATM
Customer from home: Account balances, bills, etc.
Branch office
Main office
5
4 Statistics, summaries, exceptions
Partners and clients exchange information
6 Reports
7 Processing reports
8
Reports
1 Mainframe: Processing batch jobs
3 Backups
Tape storage: Sequential data sets
2 Reports
Reports
9
Data update
10 Disk storage: Database files
Production control
System operator
zOSB003-0

Figure 2. Typical batch use

1. At night, numerous batch jobs running programs and utilities are processed. These jobs consolidate the results of the online transactions that take place 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 the next day.
5. Reports with exceptions are sent to the branch offices.
6. Monthly account balance reports are generated and sent to all bank customers.
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 operations area is monitoring the messages on the system console and the execution of the jobs.
10. Jobs and transactions are reading or updating the database (the same one that is used by online transactions) and many files are written to tape.

Mainframes working with you: Online transaction processing (OLTP)

Transaction processing that occurs interactively with the end user is referred to as *online transaction processing* or OLTP.

One of the main characteristics of a transaction system is that the interactions between the user and the system are very short. The user will perform a complete business transaction through short interactions, with immediate 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, inside a bank branch office or on the Internet, customers are using online services when checking an account balance or directing fund balances.

In fact, an online system performs many of the same functions as an operating system:
- Managing and dispatching tasks
- Controlling user access authority to system resources
- Managing the use of memory
- Managing and controlling simultaneous access to data files
- Providing device independence

Some industry uses of mainframe-based online systems include:
- Banks – ATMs, teller systems for customer 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

How might the end users in these industries interact with their mainframe systems? 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?

Before personal computers and intelligent workstations became popular, the most common way to communicate with online mainframe applications was with 3270 terminals. These devices were sometimes known as ″dumb″ terminals, but they had enough intelligence to collect and display a full screen of data rather than interacting with the computer for each keystroke, saving processor cycles. The characters were green on a black screen, so the mainframe applications were nicknamed ″green screen″ applications.

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

In this section, there is no need to describe the process of interacting with the mainframe through a Web browser, as 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:

- A small amount of input data, a few stored records accessed and processed, and a small amount of data as output
- Immediate response time, usually less than one second
- Large numbers of users involved in large numbers of 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, customers use online services when checking an account balance or making an investment.

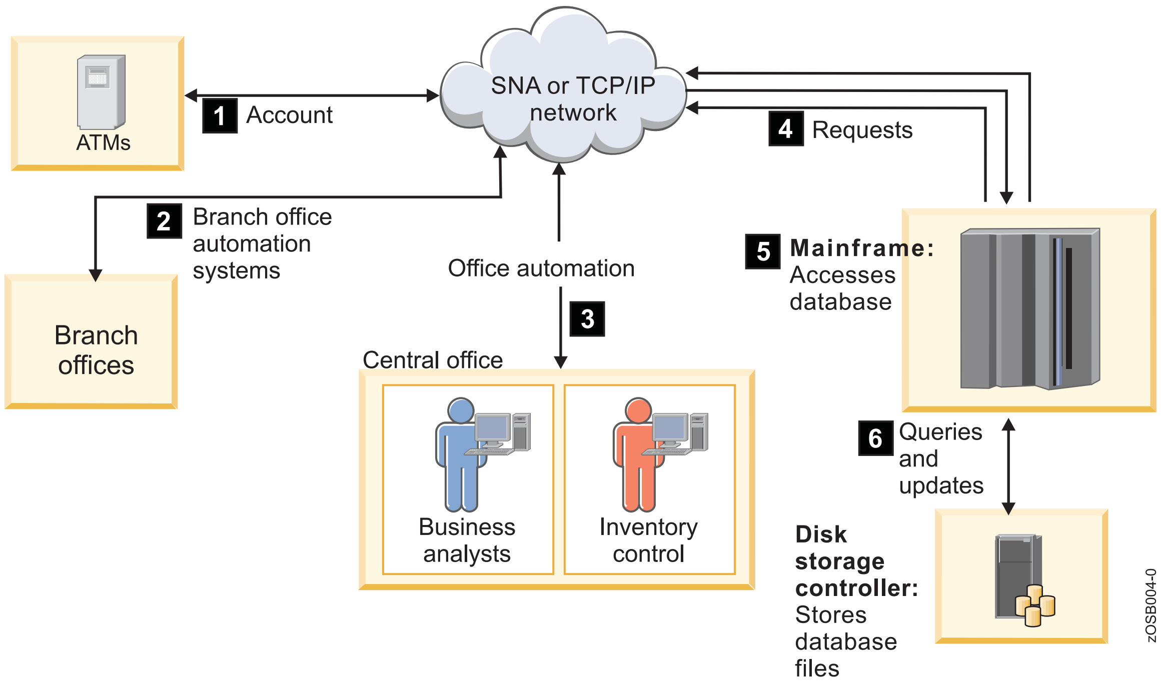

Figure 3. Typical online use

Figure 3 shows a series of common online transactions using a mainframe.

1. A customer 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 customer relationship management, budget planning, and stock control.
4. All requests are directed to the mainframe computer for processing.
5. Programs running on the mainframe computer perform updates and inquiries to the database management system (for example, DB2®).
6. Specialized disk storage systems store the database files.

Roles in the mainframe world

Mainframe systems are designed to be used by large numbers of people. Most of those who interact with mainframes are end users— people who use the applications that are hosted on the system. However, because of the large number of end users, applications running on the system, and the sophistication and complexity of the system software that supports the users and applications, a variety of roles are needed to operate and support the system.

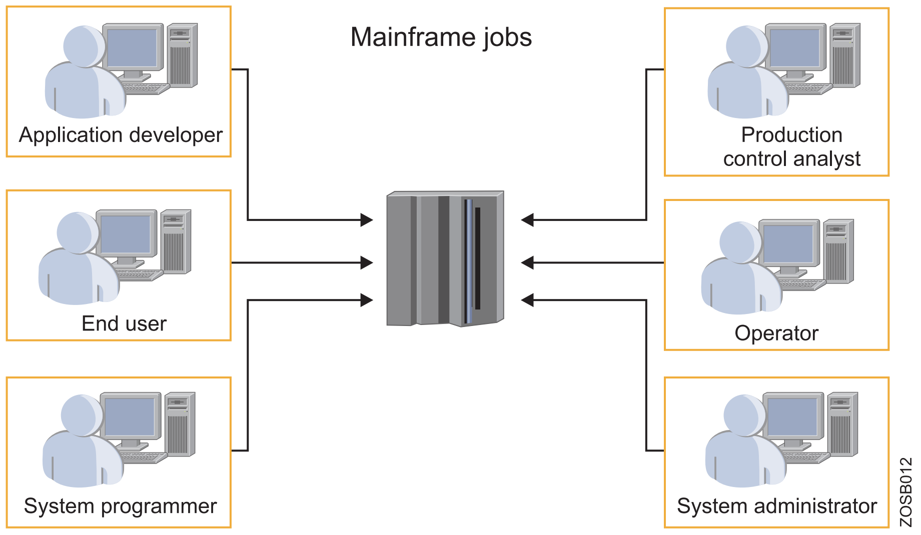

Figure 4. Who's who in the mainframe world?

In the IT field, these roles are referred to by a number of different titles, such as:

- System programmers
- System administrators
- Application designers and programmers
- System operators
- Production control analysts

In a distributed systems environment, many of the same roles are needed as in the mainframe environment. However, the job responsibilities are often not as well-defined. Since the 1960s, mainframe roles have evolved and expanded to provide an environment in which the system software and applications can function smoothly and effectively and serve many thousands of users efficiently. While it may seem that the size of the mainframe support staff is large and unwieldy, the numbers become comparatively small when one considers the number of users supported, the number of transactions run, and the high business value of the work that is performed on the mainframe.

Mainframe activities, such as the following, often require cooperation among the various roles:

- Installing and configuring system software
- Designing and coding new applications to run on the mainframe
- Introduction and management of new workloads on the system, such as batch jobs and online transaction processing
- Operation and maintenance of the mainframe software and hardware

Who is the system programmer?

In a mainframe IT organization, the *system programmer* (or system**s** programmer) plays a central role. The system programmer installs, customizes, and maintains the operating system, and also installs or upgrades products that run on the system.

The system programmer might be presented with the latest version of the operating system to upgrade the existing systems. Or, the installation might be as simple as upgrading a single program, such as a sort application.

The system programmer performs such tasks as the following:

- Planning hardware and software system upgrades and changes in configuration
- Training system operators and application programmers
- Automating operations
- Capacity planning
- Running installation jobs and scripts
- Performing installation-specific customization tasks
- Integration-testing the new products with existing applications and user procedures
- System-wide performance tuning to meet required levels of service

The system programmer must be skilled at debugging problems with system software. These problems are often captured in a copy of the computer's memory contents called a *dump* , which the system produces in response to a failing software product, user job, or transaction. Armed with a dump and specialized debugging tools, the system programmer can determine where the components have failed. When the error has occurred in a software product, the system programmer works directly with the software vendor's support representatives to discover whether the problem's cause is known and whether a patch is available.

System programmers are needed to install and maintain the *middleware* on the mainframe, such as database management systems, online transaction processing systems and Web servers. Middleware is a software "layer" between the operating system and the end user or end user application. It supplies major functions that are not provided by the operating system. Major middleware products such as DB2, CICS®, and IMS™ can be as complex as the operating system itself, if not more so.

Who is the system administrator?

The distinction between *system programmer* and *system administrator* varies widely among mainframe sites. In smaller IT organizations, where one person might be called upon to perform several roles, the terms may be used interchangeably.

In larger IT organizations with multiple departments, the job responsibilities tend to be more clearly separated. System administrators perform more of the day-to-day tasks related to maintaining the critical business data that resides on the mainframe, while the system programmer focuses on maintaining the system itself. One reason for the separation of duties is to comply with auditing procedures, which often require that no one person in the IT organization be allowed to have unlimited access to sensitive data or resources. Examples of system administrators include the *database administrator* (DBA) and the *security administrator*.

While system programmer expertise lies mainly in the mainframe hardware and software areas, system administrators are more likely to have experience with the applications. They often interface directly with the application programmers and end users to make sure that the administrative aspects of the applications are met. These roles are not necessarily unique to the mainframe environment, but they are key to its smooth operation nonetheless.

In larger IT organizations, the system administrator maintains the system software environment for business purposes, including the day-to-day maintenance of systems to keep them running smoothly. For example, the database administrator must ensure the integrity of, and efficient access to, the data that is stored in the database management systems.

Other examples of common system administrator tasks can include:
- Installing software
- Adding and deleting users and maintaining user profiles
- Maintaining security resource access lists
- Managing storage devices and printers
- Managing networks and connectivity
- Monitoring system performance

In matters of problem determination, the system administrator generally relies on the software vendor support center personnel to diagnose problems, read dumps, and identify corrections for cases in which these tasks aren't performed by the system programmer.

Who are the application designers and programmers?

The *application designer* and *application programmer* (or "application developer") design, build, test, and deliver mainframe applications for the company's end users and customers. Based on requirements gathered from business analysts and end users, the designer creates a design specification from which the programmer constructs an application. The process includes several iterations of code changes and compilation, application builds, and unit testing.

During the application development process, the designer and programmer must interact with other roles in the enterprise. For example, the programmer often works on a team of other programmers who are building code for related application program modules. When completed, each module is passed through a testing process that can include function, integration, and system-wide tests. Following the tests, the application programs must be acceptance tested by the user community to determine whether the code actually satisfies the original user requirement.

In addition to creating new application code, the programmer is responsible for maintaining and enhancing the company's existing mainframe applications. In fact, this is often the primary job for many of today's mainframe application programmers. While mainframe installations still create new programs with Common Business Oriented Language (COBOL) or PL/I, languages such as Java™ have become popular for building new applications on the mainframe, just as they have on distributed platforms.

Widespread development of mainframe programs written in high-level languages such as COBOL and PL/I continues at a brisk pace, despite rumors to the contrary. Many thousands of programs are in production on mainframe systems around the world, and these programs are critical to the day-to-day business of the corporations that use them. COBOL and other high-level language programmers are needed to maintain existing code and make updates and modifications to existing programs. Also, many corporations continue to build new application logic in COBOL and other traditional languages, and IBM continues to enhance their high-level language compilers to include new functions and features that allow those languages to continue to take advantage of newer technologies and data formats.

Who is the system operator?

The *system operator* monitors and controls the operation of the mainframe hardware and software. The operator starts and stops system tasks, monitors the system consoles for unusual conditions, and works with the system programming and production control staff to ensure the health and normal operation of the systems.

System console messages can be so voluminous that operators often have a difficult time determining whether a situation is really a problem. In recent years, tools to reduce the volume of messages and automate message responses to routine situations have made it easier for operators to concentrate on unusual events that might require human intervention.

As applications are added to the mainframe, the system operator is responsible for ensuring that they run smoothly. New applications from the Applications Programming Department are typically delivered to the Operations Staff with a *run book* of instructions. A run book identifies the specific operational requirements of the application, which operators need to be aware of during job execution. Run book instructions might include, for example:

- Application-specific console messages that require operator intervention
- Recommended operator responses to specific system events
- Directions for modifying job flows to accommodate changes in business requirements

The operator is also responsible for starting and stopping the major subsystems, such as transaction processing systems, database systems, and the operating system itself. These "restart operations" are not nearly as commonplace as they once were, as the availability of the mainframe has improved dramatically over the years. However, the operator must still perform an orderly shutdown and startup of the system and its workloads, when it is required.

In case of a failure or an unusual situation, the operator communicates with system programmers, who assist the operator in determining the proper course of action, and with the production control analyst, who works with the operator to make sure that production workloads are completing properly.

Who is the production control analyst?

The *production control analyst* is responsible for making sure that batch workloads run to completion— without error or delay.

Some mainframe installations run interactive workloads for online users, followed by batch updates that run after the prime shift when the online systems are not running. While this execution model is still common, worldwide operations at many companies— with live, Internet-based access to production data— are finding the "daytime online/night time batch" model to be obsolete. Batch workloads continue to be a part of information processing, however, and skilled production control analysts play a key role.

A common complaint about mainframe systems is that they are inflexible and hard to work with, specifically in terms of implementing changes. The production control analyst often hears this type of complaint, but understands that the use of well-structured rules and procedures to control changes— a strength of the mainframe environment— helps to prevent outages. In fact, one reason that mainframes have attained a strong reputation for high levels of availability and performance is that there are controls on change and it is difficult to introduce change without proper procedures.

What roles do vendors play?

A number of vendor roles are commonplace in the mainframe shop.

Because most mainframe computers are sold by IBM, and the operating systems and primary online systems are also provided by IBM, most vendor contacts are IBM employees. However, *independent software vendor* (ISV) products are also used in the IBM mainframe environment, and customers use *original equipment manufacturer* (OEM) hardware, such as disk and tape storage devices, as well.

Typical vendor roles follow:

Hardware support or customer engineer
Hardware vendors usually provide on-site support for hardware devices. The IBM hardware maintenance person is often referred to as the *customer engineer* (CE). The CE provides installation and repair service for the mainframe hardware and peripherals. The CE usually works directly with the operations teams when hardware fails or new hardware is being installed.

Software support
A number of vendor roles exist to support software products on the mainframe. IBM has a centralized Support Center that provides entitled and extra-charge support for software defects or usage assistance. There are also information technology specialists and architects who can be engaged to provide additional pre- and post-sales support for software products, depending upon the size of the enterprise and the particular customer situation.

Field technical sales support, systems engineer, or client representative
For larger mainframe accounts, IBM and other vendors provide face-to-face sales support. The vendor representatives specialize in various types of hardware or software product families and call on the part of the customer organization that influences the product purchases. At IBM, the technical sales specialist is referred to as the *field technical sales support* (FTSS) person, or by the older term, *systems engineer* (SE).

For larger mainframe accounts, IBM frequently assigns a client representative, who is attuned to the business issues of a particular industry sector, to work exclusively with a small number of customers. The client representative acts as the general “single point of contact” between the customer and the various organizations within IBM.

What are mainframe operating systems?

In simplest terms, an **operating system** is a collection of programs that manage a computer system’s internal workings— its memory, processors, devices, and file system. Mainframe operating systems are sophisticated products with substantially different characteristics and purposes.

Operating systems are designed to make the best use of the computer’s various resources, and ensure that the maximum amount of work is processed as efficiently as possible. Although an operating system cannot increase the speed of a computer, it can maximize use of resources, thereby making the computer seem faster by allowing it to do more work in a given period of time.

A computer's **architecture** consists of the functions the computer system provides. The architecture is distinct from the physical design, and, in fact, different machine designs might conform to the same computer architecture. In a sense, the architecture is the computer as seen by the user, such as a system programmer. For example, part of the architecture is the set of machine instructions that the computer can recognize and execute. In the mainframe environment, the system software and hardware comprise a highly advanced computer architecture, the result of decades of technological innovation.

Most of this information center teaches the fundamentals of z/OS, which is IBM's foremost mainframe operating system. It is useful for mainframe students, however, to have a working knowledge of other mainframe operating systems. One reason is that a given mainframe computer might run multiple operating systems. For example, the use of z/OS, z/VM®, and Linux® on the same mainframe is common.

In addition to z/OS, four other operating systems dominate mainframe usage: z/VM, z/VSE™, Linux for System z™, and z/TPF.

Mainframe operating system: z/OS

z/OS, a widely used mainframe operating system, is designed to offer a stable, secure, and continuously available environment for applications running on the mainframe.

z/OS today is the result of decades of technological advancement. It evolved from an operating system that could process a single program at a time to an operating system that can handle many thousands of programs and interactive users concurrently. To understand how and why z/OS functions as it does, it is important to understand some basic concepts about z/OS and the environment in which it functions.

In most early operating systems, requests for work entered the system one at a time. The operating system processed each request or **job** as a unit, and did not start the next job until the one being processed had completed. This arrangement worked well when a job could execute continuously from start to completion. But often a job had to wait for information to be read in from, or written out to, a device such as a tape drive or printer. Input and output (I/O) take a long time compared to the electronic speed of the processor. When a job waited for I/O, the processor was idle.

Finding a way to keep the processor working while a job waited would increase the total amount of work the processor could do without requiring additional hardware. z/OS gets work done by dividing it into pieces and giving portions of the job to various system components and subsystems that function interdependently. At any point in time, one component or another gets control of the processor, makes its contribution, and then passes control along to a user program or another component.

Mainframe operating system: z/VM

As a control program, **z/Virtual Machine** (z/VM) is a **hypervisor** because it runs other operating systems in the virtual machines it creates.

Any of the IBM mainframe operating systems such as z/OS, Linux for zSeries®, z/VSE, and z/TPF can be run as **guest systems** in their own virtual machines, and z/VM can run any combination of guest systems.

z/VM has two basic components: a **control program** (CP) and a single-user operating system, CMS. The control program artificially creates multiple virtual machines from the real hardware resources. To end users, it appears as if they have dedicated use of the shared real resources. The shared real resources include printers, disk storage devices, and the CPU. The control program ensures data and application security among the guest systems. The real hardware can be shared among the guests, or dedicated to a single guest for performance reasons. The system programmer allocates the real devices among the guests. For most customers, the use of guest systems avoids the need for larger hardware configurations.

z/VM's other major component is the **Conversational Monitor System** or CMS. This component of z/VM runs in a virtual machine and provides both an interactive end-user interface and the general z/VM application programming interface.

Mainframe operating system: z/VSE

z/Virtual Storage Extended (z/VSE) is popular with users of smaller mainframe computers. Some of these customers eventually migrate to z/OS when they grow beyond the capabilities of z/VSE.

Compared to z/OS, the z/VSE operating system provides a smaller, less complex base for batch processing and transaction processing. The design and management structure of z/VSE is excellent for running routine production workloads consisting of multiple batch jobs (running in parallel) and extensive, traditional transaction processing. In practice, most z/VSE users also have the z/VM operating system and use this as a general terminal interface for z/VSE application development and system management.

z/VSE was originally known as **Disk Operating System** (DOS), and was the first disk-based operating system introduced for the System/360 mainframe computers. DOS was seen as a temporary measure until OS/360 would be ready. However, some mainframe customers liked its simplicity (and small size) and decided to remain with it after OS/360 became available. DOS became known as DOS/VS (when it started using virtual storage), then VSE/SP and later VSE/ESA™, and most recently z/VSE. The name VSE is often used collectively to refer to any of the more recent versions.

Mainframe operating system: Linux for System z

Several (non-IBM) Linux distributions can be used on a mainframe.

There are two generic names for these distributions:
- Linux for S/390® (uses 31-bit addressing and 32-bit registers)
- Linux for System z (uses 64-bit addressing and registers)

The phrase **Linux for System z** is used to refer to Linux running on an S/390 system or System z, when there is no specific need to refer explicitly to either the 31-bit version or the 64-bit version. We assume students are generally familiar with Linux and therefore we mention only those characteristics that are relevant for mainframe usage. These include the following:

- Linux uses traditional count key data (CKD) disk devices and SAN-connected SCSI-type devices. Other mainframe operating systems can recognize these drives as Linux drives, but cannot use the data formats on the drives. That is, there is no sharing of data between Linux and other mainframe operating systems.
- Linux does not use 3270 display terminals, while all other mainframe operating systems use 3270s as their basic terminal architecture. Linux uses X-Window System based terminals or X-Window System emulators on PCs; it also supports typical ASCII terminals, usually connected through the **telnet** protocol. The X-Window System is the standard for graphical interfaces in Linux. It is the middle layer between the hardware and the window manager.
- With the proper setup, a Linux system under z/VM can be quickly cloned to make another, separate Linux image. The z/VM emulated LAN can be used to connect multiple Linux images and to provide an external LAN route for them. Read-only file systems, such as a typical /**usr** file system, can be shared by Linux images.
- Linux on a mainframe operates with the ASCII character set, not the EBCDIC form of stored data that is typically used on mainframes. Here, EBCDIC is used only when writing to such character-sensitive devices as displays and printers. The Linux drivers for these devices handle the character translation.

Mainframe operating system: z/TPF

The **z/Transaction Processing Facility** (z/TPF) operating system is a special-purpose system that is used by companies with very high transaction volume, such as credit card companies and airline reservation systems.

z/TPF was once known as Airline Control Program (ACP). It is still used by airlines and has been extended for other very large systems with high-speed, high-volume transaction processing requirements.

z/TPF can use multiple mainframes in a loosely coupled environment to routinely handle tens of thousands of transactions per second, while experiencing uninterrupted availability that is measured in years. Very large terminal networks, including special-protocol networks used by portions of the reservation industry, are common.

Chapter 2. Mainframe hardware concepts

The original S/360 architecture— based on CPUs, memory, channels, control units, and devices, and they way these are addressed— is fundamental to understanding mainframe hardware, because the concepts and terminology of the original design still permeate today's mainframe descriptions and designs.

The ability to partition a large system into multiple smaller systems, called *logical partitions* or *LPARs*, is now a core requirement in practically all mainframe installations. The flexibility of the hardware design, allowing any processor to access and accept interrupts for any channel, control unit, and device connected to a given LPAR, contributes to the flexibility, reliability, and performance of the complete system.

Unique to mainframes is the availability of a pool of processors that can be configured, or characterized, to do specific work. Having this processor capability provides great flexibility in meeting customer requirements, some of which are based on the cost structures of some mainframe software.

In addition to this pool of primary processors, mainframes have a network of controllers (special microprocessors) that control the system as a whole. These controllers are not visible to the operating system or application programs.

Mainframe hardware: Terminology

In the early S/360 days a system had a single processor, which was also known as the central processing unit (CPU). The terms **system**, **processor**, and **CPU** were used interchangeably. However, these terms became confusing when systems became available with more than one processor.

This overlap is shown in Figure 5 on page 24.

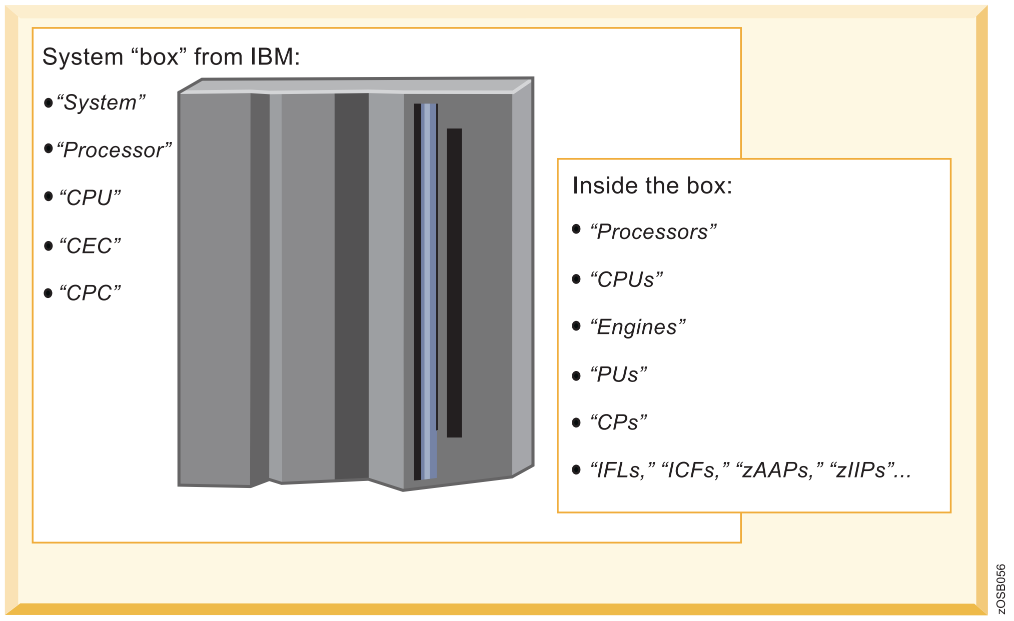

Figure 5. Terminology overlap

Processor and CPU can refer to either the complete system box, or to one of the processors (CPUs) within the system box. Although the meaning may be clear from the context of a discussion, even mainframe professionals must clarify which processor or CPU meaning they are using in a discussion. IBM uses the term **central processor complex (CPC)** to refer to the physical collection of hardware that includes main storage, one or more central processors, timers, and channels. (Some system programmers use the term **central electronic complex (CEC)** to refer to the mainframe "box," but the preferred term is CPC.)

Briefly, all the S/390 or z/Architecture® processors within a CPC are **processing units** (PUs). When IBM delivers the CPC, the PUs are characterized as CPs (for normal work), Integrated Facility for Linux (IFL), Integrated Coupling Facility (ICF) for Parallel Sysplex configurations, and so forth.

Mainframe professionals typically use **system** to indicate the hardware box, a complete hardware environment (with I/O devices), or an operating environment (with software), depending on the context. They typically use **processor** to mean a single processor (CP) within the CPC.

Mainframe hardware: Evolving design

Early mainframe designs help explain the terminology that still permeates mainframe discussions.

Figure 6 on page 25 presents a conceptual diagram of an early mainframe design: an S/360 system.

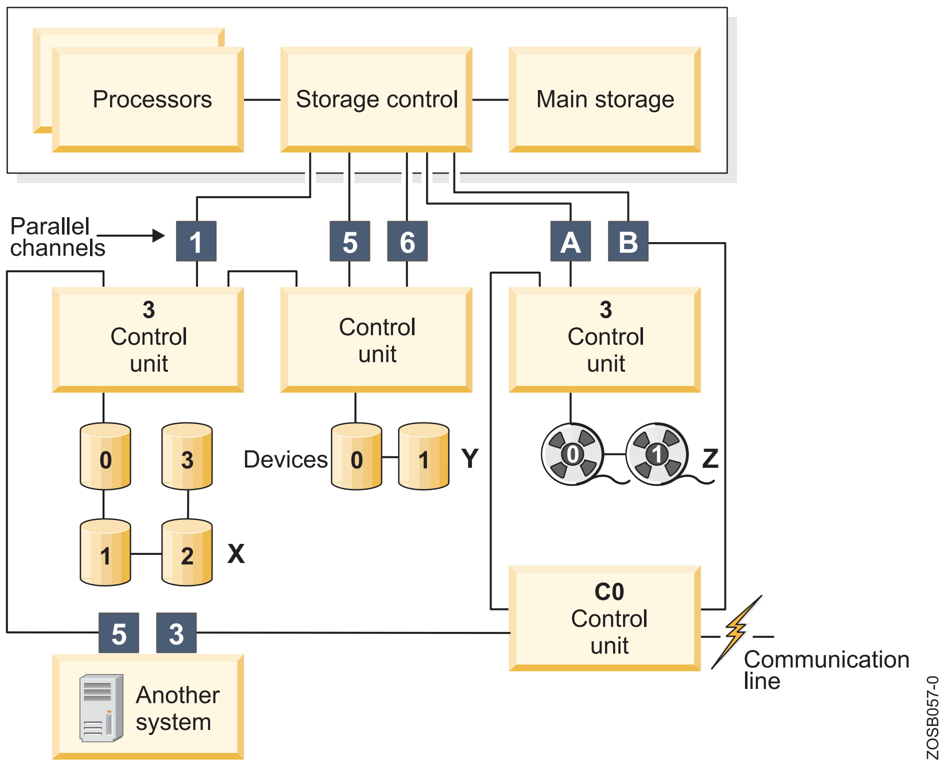

Figure 6. Conceptual S/360

The central processor contains the processors, memory, control circuits, and interfaces for **channels**. A channel provides an independent data and control path between I/O devices and memory. Early systems had up to 16 channels; the today's largest mainframe machines can have over 1000 channels.

Channels connect to **control units**. A control unit contains logic to work with a particular type of I/O device. For example, a control unit for a printer would have much different internal circuitry and logic than a control unit for a tape drive. Some control units can have multiple channel connections providing **multiple paths** to the control unit and its devices.

Control units connect to **devices**, such as disk drives, tape drives, communication interfaces, and so forth. The division of circuitry and logic between a control unit and its devices is not defined, but it is usually more economical to place most of the circuitry in the control unit.

The channels in Figure 6 are **parallel channels** (also known as **bus** and **tag channels**, named for the two heavy copper cables they use). A parallel channel can be connected to a maximum of eight control units. Most control units can be connected to multiple devices; the maximum depends on the particular control unit, but 16 is a typical number.

Each channel, control unit, and device has an address, expressed as a hexadecimal number. The disk drive marked with an X in Figure 6 has address 132:
- The first digit is the channel number
- The second digit is the control unit number
- The last digit is the device number

The disk drive marked with a Y in the figure can be addressed as 171, 571, or 671 because it is connected through three channels. By convention the device is known by its lowest address (171), but all three addresses could be used by the operating system to access the disk drive. Multiple paths to a device are useful for performance and for availability. When an application wants to access disk 171, the operating system will first try channel 1. If it is busy (or not available), it will try channel 5, and so forth.

Figure 6 on page 25 contains another S/360 system with two channels connected to control units used by the first system. This sharing of I/O devices is common in all mainframe installations. Tape drive Z is address A31 for the first system, but is address 331 for the second system. Sharing devices, especially disk drives, is not a simple topic and there are hardware and software techniques used by the operating system to control exposures such as updating the same disk data at the same time from two independent systems.

Today's mainframe designs are more complex than shown in Figure 6 on page 25; differences include:

- Parallel channels are not available on the newest mainframes and are slowly being displaced on older systems.
- Parallel channels have been replaced with ESCON® (Enterprise Systems CONnection) and FICON® (FIber CONnection) channels. These channels connect to only one control unit or, more likely, are connected to a **director** (switch) and are optical fibers.
- Current mainframes have more than 16 channels and use two hexadecimal digits as the channel portion of an address.
- Channels are generally known as CHPIDs (channel path identifiers) or PCHIDs (physical channel identifiers) on later systems, although the term **channel** is also correct. The channels are all integrated in the main processor box.

The device address seen by software is more correctly known as a device number (although the term **address** is still widely used) and is indirectly related to the control unit and device addresses.

Mainframe hardware: I/O connectivity

A single System z9 mainframe can have up to 1024 individual channels for input and output (I/O) connectivity. This capacity is one factor that contributes to the mainframe's legendary scalability.

Although less complex than a real system with more channels and I/O devices, Figure 7 on page 27 illustrates key concepts related to I/O configurations and capacity.

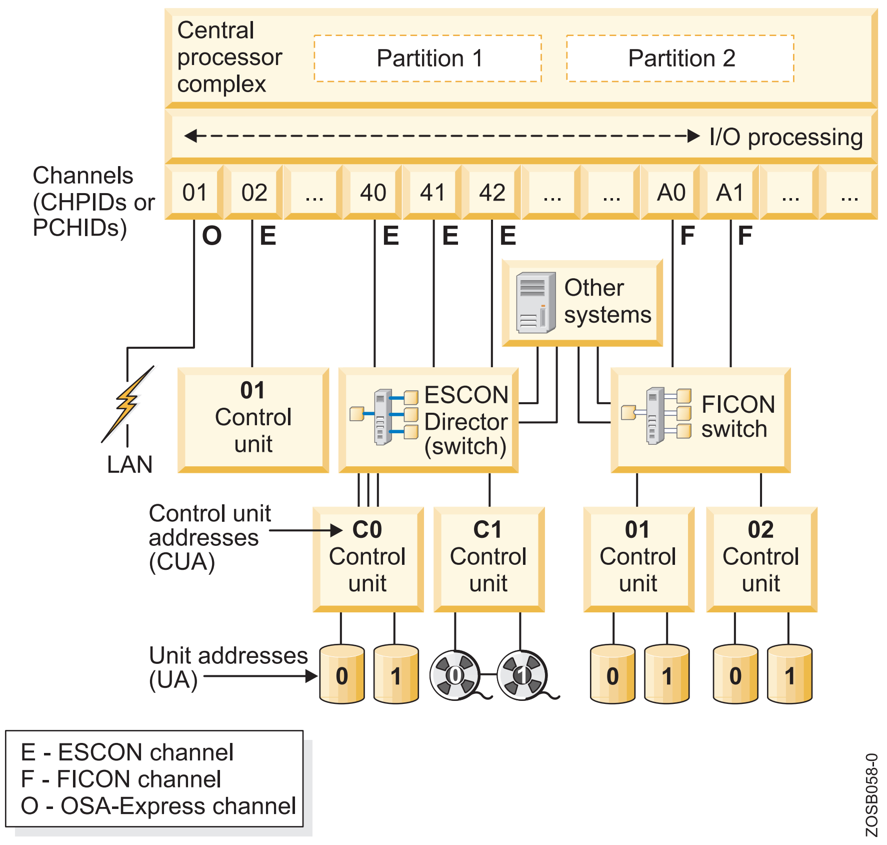

Figure 7. Recent system configuration

Briefly, partitions create separate logical machines in the central processor complex (CPC). ESCON and FICON channels are logically similar to parallel channels but they use fiber connections and operate much faster. A modern system might have 100-200 channels or channel path identifiers (CHPIDs). Key concepts partly illustrated here include the following:

- ESCON and FICON channels connect to only one device or one port on a switch.
- Most modern mainframes use switches between the channels and the control units. The switches may be connected to several systems, sharing the control units and some or all of its I/O devices across all the systems.
- CHPID addresses are two hexadecimal digits.
- Multiple partitions can sometimes share CHPIDs. Whether this is possible depends on the nature of the control units used through the CHPIDs. In general, CHPIDs used for disks can be shared.
- An I/O subsystem layer exists between the operating systems in partitions (or in the basic machine if partitions are not used) and the CHPIDs.

An ESCON director or FICON switch is a sophisticated device that can sustain high data rates through many connections. (A large director might have 200 connections, for example, and all of these can be passing data at the same time.) The director or switch must keep track of which CHPID (and partition) initiated which I/O operation so that data and status information is returned to the right place. Multiple I/O requests, from multiple CHPIDs attached to multiple partitions on multiple systems, can be in progress through a single control unit.

The I/O control layer uses a control file known as an IOCDS (I/O Control Data Set) that translates physical I/O addresses (composed of CHPID numbers, switch port numbers, control unit addresses, and unit addresses) into **device numbers** that are used by the operating system software to access devices. This is loaded into the Hardware Save Area (HSA) at power-on and can be modified dynamically. A device number looks like the addresses for early S/360 machines except that it can contain three or four hexadecimal digits.

Many users still refer to these as "addresses" although the device numbers are arbitrary numbers between x'0000' and x'FFFF'. Today's mainframes have two layers of I/O address translations between the real I/O elements and the operating system software. The second layer was added to make migration to newer systems easier.

Modern control units, especially for disks, often have multiple channel (or switch) connections and multiple connections to their devices, as shown in Figure 8. They can handle multiple data transfers at the same time on the multiple channels. Each device will have a unit control block (UCB) in each z/OS image.

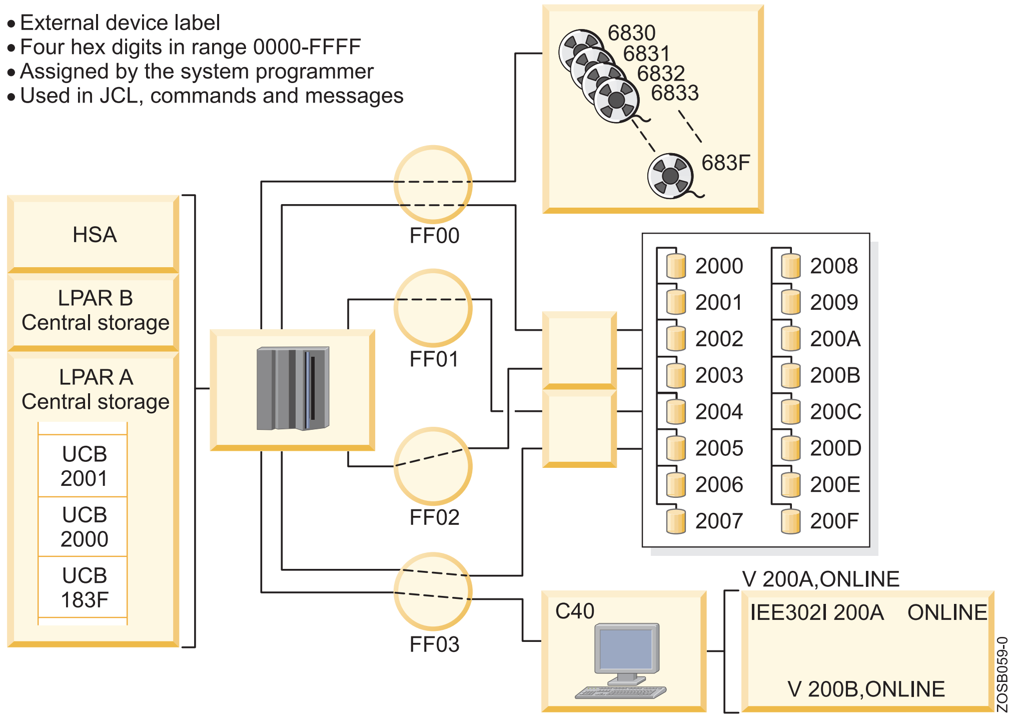

Figure 8. Device addressing

Mainframe hardware: System control and partitioning

Each System z model has several elements that constitute the hardware control system for the mainframe.

Figure 9 on page 29, while highly conceptual, shows several of the functions of the internal system controls on current mainframes. The internal controllers are microprocessors but use a much simpler organization and instruction set than zSeries processors. They are usually known as controllers to avoid confusion with zSeries processors.

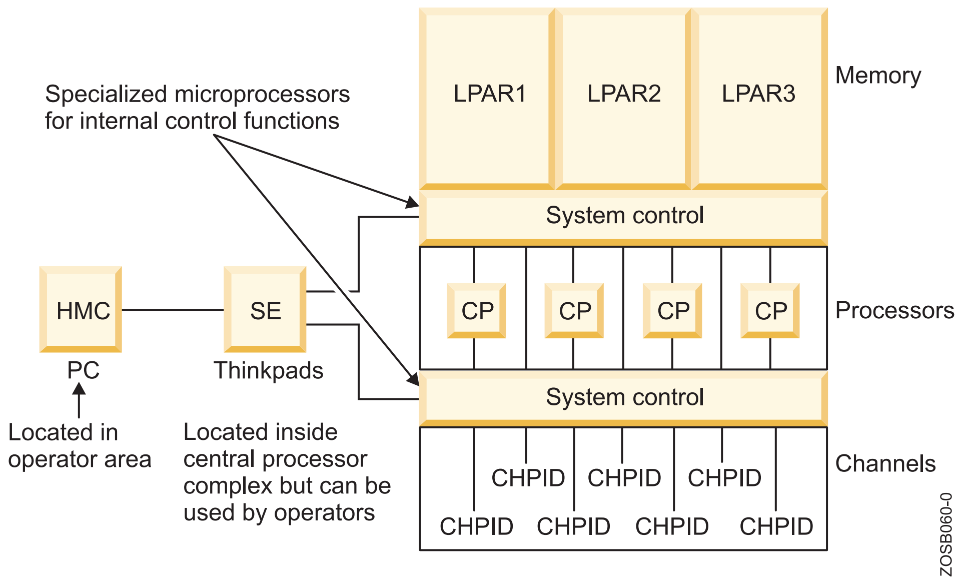

Figure 9. System control and partitioning

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

For many years there was a limit of 15 LPARs in a mainframe; more recent machines have 30 (and potentially more). Practical limitations of memory size, I/O availability, and available processing power usually limit the number of LPARs to less than these maximums.

The hardware and firmware that provides partitioning is known as PR/SM™ (Processor Resource/System Manager). It is the PR/SM functions that are used to create and run LPARs. This difference between PR/SM (a built-in facility) and LPARs (the result of using PR/SM) is often ignored and the term LPAR is used collectively for the facility and its results.

System administrators assign portions of memory to each LPAR; memory cannot be shared among LPARs. The administrators can assign processors (noted as CPs in Figure 9) to specific LPARs or they can allow the system controllers to dispatch any or all the processors to all the LPARs using an internal load-balancing algorithm. Channels (CHPIDs) can be assigned to specific LPARs or can be shared by multiple LPARs, depending on the nature of the devices on each channel.

A system with a single processor (CP processor) can have multiple LPARs. PR/SM has an internal dispatcher that can allocate a portion of the processor to each LPAR, much as an operating system dispatcher allocates a portion of its processor time to each process, thread, or task.

Partitioning control specifications are partly contained in the IOCDS and are partly contained in a system **profile**. The IOCDS and profile both reside in the Support Element (SE) which is simply a notebook computer inside the system. The SE can be connected to one or more Hardware Management Consoles (HMCs), which are desktop personal computers used to monitor and control hardware such as the mainframe microprocessors An HMC is more convenient to use than an SE and can control several different mainframes.

Working from an HMC (or from an SE, in unusual circumstances), an operator prepares a mainframe for use by selecting and loading a profile and an IOCDS. These create LPARs and configure the channels with device numbers, LPAR assignments, multiple path information, and so forth. This is known as a Power-on Reset (POR). By loading a different profile and IOCDS, the operator can completely change the number and nature of LPARs and the appearance of the I/O configuration. However, doing this is usually disruptive to any running operating systems and applications and is therefore seldom done without advance planning.

Mainframe hardware: Logical partitions (LPARs)

Logical partitions (LPARs) are, in practice, equivalent to separate mainframes.

Each LPAR runs its own operating system. This can be any mainframe operating system; there is no need to run z/OS, for example, in each LPAR. The installation planners may elect to share I/O devices across several LPARs, but this is a local decision.

The system administrator can assign one or more system processors for the exclusive use of an LPAR. Alternately, the administrator can allow all processors to be used on some or all LPARs. Here, the system control functions (often known as microcode or firmware) provide a dispatcher to share the processors among the selected LPARs. The administrator can specify a maximum number of concurrent processors executing in each LPAR. The administrator can also provide weightings for different LPARs; for example, specifying that LPAR1 should receive twice as much processor time as LPAR2.

The operating system in each LPAR is IPLed separately, has its own copy of its operating system, has its own operator console (if needed), and so forth. If the system in one LPAR crashes, there is no effect on the other LPARs.

In a mainframe system with three LPARs, for example, you might have a production z/OS in LPAR1, a test version of z/OS in LPAR2, and Linux for S/390 in LPAR3. If this total system has 8 GB of memory, we might have assigned 4 GB to LPAR1, 1 GB to LPAR2, 1 GB to LPAR3, and have kept 2 GB in reserve. The operating system consoles for the two z/OS LPARs might be in completely different locations.

For most practical purposes there is no difference between, for example, three separate mainframes running z/OS (and sharing most of their I/O configuration) and three LPARs on the same mainframe doing the same thing. With minor exceptions z/OS, the operators, and applications cannot detect the difference.

The minor differences include the ability of z/OS (if permitted when the LPARs were defined) to obtain performance and utilization information across the complete mainframe system and to dynamically shift resources (processors and channels) among LPARs to improve performance.

Mainframe hardware: Consolidation of mainframes

There are fewer mainframes in use today than there were 15 or 20 years ago. In some cases, all the applications were moved to other types of systems; however, in most cases the reduced number is due to consolidation. That is, several smaller mainframes have been replaced with a smaller number of larger systems.

There is a compelling reason for consolidation. Mainframe software (from many vendors) can be expensive and typically costs more than the mainframe hardware. It is usually less expensive (and sometimes **much** less expensive) to replace multiple software licenses (for smaller machines) with one or two licenses (for larger machines). Software license costs are often linked to the power of the system but the pricing curves favor a small number of large machines.

Software license costs for mainframes have become a dominant factor in the growth and direction of the mainframe industry. Several nonlinear factors make software pricing very difficult. We must remember that mainframe software is not a mass market situation like PC software. The growth of mainframe processing power in recent years has been nonlinear.

The relative power needed to run a traditional mainframe application (for example, a batch job written in COBOL) does not have a linear relation to the power needed for a new application (with a GUI interface, written in C and Java). The consolidation effect has produced very powerful mainframes. These mainframes might need 1% of their power to run an application, but the application vendor often sets a price based on the total power of the machine.

This pricing policy results in the odd situation where customers want the latest mainframe (to obtain new functions or to reduce maintenance costs associated with older machines) but they want the **slowest** mainframe that will run their applications (to reduce software costs based on total system processor power).

Mainframe hardware: Processing units

Early mainframes had a single processor, which was known as the central processing unit (CPU). Today's IBM mainframes have a central processor complex (CPC), which may contain several different types of z/Architecture processors that can be used for slightly different purposes.

Several of these purposes are related to software cost control, while others are more fundamental. All of the processors in the CPC begin as equivalent processor units (PUs) or engines that have not been characterized for use. Each processor is characterized by IBM during installation or at a later time. The potential characterizations are:

Central Processor (CP)
This processor type is available for normal operating system and application software.

System Assistance Processor (SAP)
Every modern mainframe has at least one SAP; larger systems may have several. The SAPs execute internal code to provide the I/O subsystem. An SAP, for example, translates device numbers and real addresses of channel path identifiers (CHPIDs), control unit addresses, and device numbers. It manages multiple paths to control units and performs error recovery for temporary errors. Operating systems and applications cannot detect SAPs, and SAPs do not use any ″normal″ memory.

Integrated Facility for Linux (IFL)
This is a normal processor with one or two instructions disabled that are used only by z/OS. Linux does not use these instructions and can be

executed by an IFL. Linux can be executed by a CP as well. The difference is that an IFL is not counted when specifying the model number of the system. This can make a substantial difference in software costs.

zAAP This is a processor with a number of functions disabled (interrupt handling, some instructions) such that no full operating system can be executed on the processor. However, z/OS can detect the presence of zAAP processors and will use them to execute Java code. The same Java code can be executed on a standard CP. Again, zAAP engines are not counted when specifying the model number of the system. Like IFLs, they exist only to control software costs.

zIIP The System z9 Integrated Information Processor (zIIP) is a specialized engine for processing eligible database workloads. The zIIP is designed to help lower software costs for select workloads on the mainframe, such as business intelligence (BI), enterprise resource planning (ERP) and customer relationship management (CRM). The zIIP reinforces the mainframe's role as the data hub of the enterprise by helping to make direct access to DB2 more cost effective and reducing the need for multiple copies of the data.

Integrated Coupling Facility (ICF)
These processors run only Licensed Internal Code. They are not visible to normal operating systems or applications. A coupling facility is, in effect, a large memory scratch pad used by multiple systems to coordinate work. ICFs must be assigned to LPARs that then become coupling facilities.

Spare An uncharacterized PU functions as a ″spare.″ If the system controllers detect a failing CP or SAP, it can be replaced with a spare PU. In most cases this can be done without any system interruption, even for the application running on the failing processor.

In addition to these characterizations of processors, some mainframes have models or versions that are configured to operate slower than the potential speed of their CPs. This is widely known as **kneecapping** , although IBM prefers the term **capacity setting**, or something similar. It is done by using microcode to insert null cycles into the processor instruction stream. The purpose, again, is to control software costs by having the minimum mainframe model or version that meets the application requirements. IFLs, SAPs, zAAPs, zIIPs, and ICFs always function at the full speed of the processor because these processors ″do not count″ in software pricing calculations.

Mainframe hardware: Multiprocessors

It is possible to purchase a current mainframe with a single processor (CP), but this is not a typical system. The term **multiprocessor** means several processors (CP processors) and implies that several processors are used by a copy of z/OS.

All operating systems today, from PCs to mainframes, can work in a multiprocessor environment. However, the degree of integration of the multiple processors varies considerably. For example, pending interrupts in a system (or in an LPAR) can be accepted by any processor in the system (or working in the LPAR). Any processor can initiate and manage I/O operations to any channel or device available to the system or LPAR. Channels, I/O devices, interrupts, and memory are owned by the system (or by the LPAR) and not by any specific processor.

This multiprocessor integration appears simple on the surface, but its implementation is complex. It is also important for maximum performance; the ability of any processor to accept any interrupt sent to the system (or to the LPAR) is especially important.

Each processor in a system (or in an LPAR) has a small private area of memory (8 KB starting at real address 0 and always mapped to virtual address 0) that is unique to that processor. This area is the Prefix Storage Area (PSA) and is used for interrupt handling and for error handling. A processor can access another processor's PSA through special programming, although this is normally done only for error recovery purposes. A processor can interrupt other processors by using a special instruction (SIGP, for Signal Processor). Again, this is typically used only for error recovery.

Mainframe hardware: Disk devices

IBM 3390 disk drives are commonly used on current mainframes.

Conceptually, this is a simple arrangement, as shown in Figure 10.

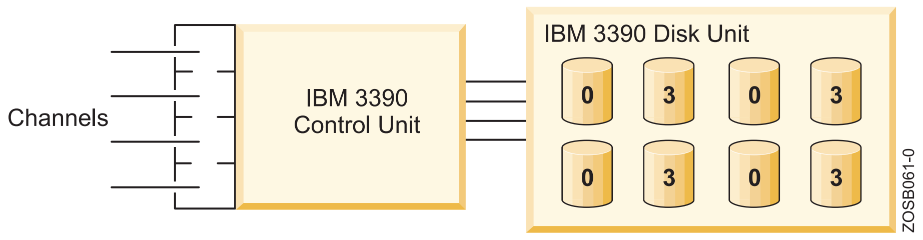

Figure 10. Initial IBM 3390 disk implementation

The associated control unit (3990) typically has four channels connected to one or more processors (probably with a switch), and the 3390 unit typically has eight or more disk drives. Each disk drive has the characteristics explained earlier. This illustration shows 3990 and 3390 units, and it also represents the concept or architecture of current devices.

The current equivalent device is an IBM 2105 Enterprise Storage Server®, simplistically illustrated in Figure 11 on page 34.

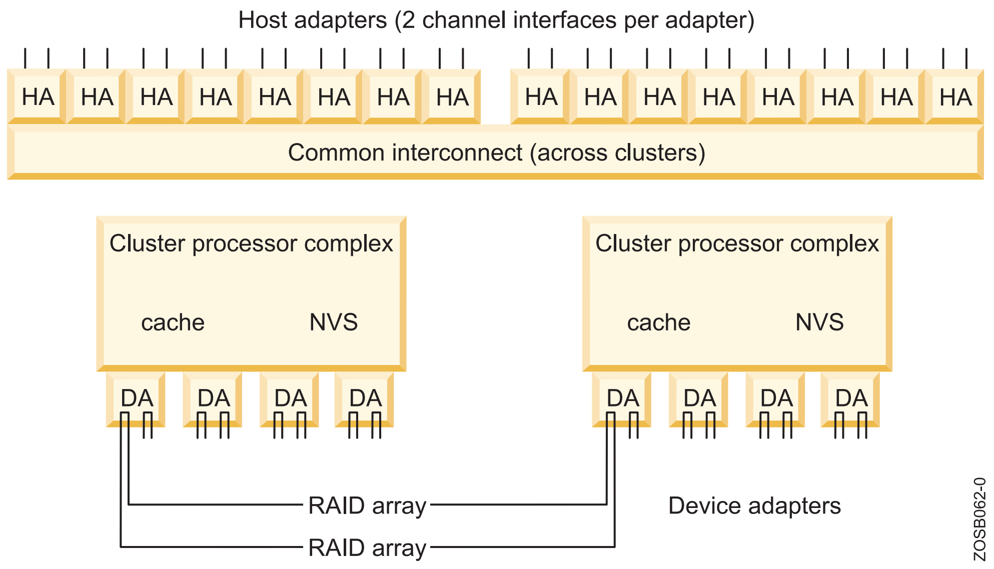
Host adapters (2 channel interfaces per adapter)
HA
Common interconnect (across clusters)
Cluster processor complex
cache
NVS
DA
RAID array
RAID array
Device adapters
ZOSB062-0

Figure 11. Current 3390 implementation

The 2105 unit is a very sophisticated device. It emulates a large number of control units and 3390 disk drives. It contains up to 11 TB of disk space, has up to 32 channel interfaces, 16 GB cache, and 284 MB of non-volatile memory (used for write queueing). The Host Adapters appear as control unit interfaces and can connect up to 32 channels (ESCON or FICON).

The physical disk drives are commodity SCSI-type units (although a serial interface, known as SSA, is used to provide faster and redundant access to the disks). A number of internal arrangements are possible, but the most common involves many RAID 5 arrays with hot spares. Practically everything in the unit has a spare or fallback unit. The internal processing (to emulate 3990 control units and 3390 disks) is provided by four high-end RISC processors in two processor complexes; each complex can operate the total system. Internal batteries preserve transient data during short power failures. A separate console is used to configure and manage the unit.

The 2105 offers many functions not available in real 3390 units, including FlashCopy®, Extended Remote Copy, Concurrent Copy, Parallel Access Volumes, Multiple Allegiance, a huge cache, and so forth.

A simple 3390 disk drive (with control unit) has different technology from the 2105 just described. However, the basic architectural appearance to software is the same. This allows applications and system software written for 3390 disk drives to use the newer technology with no revisions.

There have been several stages of new technology implementing 3390 disk drives; the 2105 is the most recent of these. The process of implementing an architectural standard (in this case the 3390 disk drive and associated control unit) with newer and different technology while maintaining software compatibility is characteristic of mainframe development. Maintaining application compatibility over long periods of technology change is an important characteristic of mainframes.

Chapter 3. Mainframe configurations: Large, medium and small (yes, small)

Diagrams and descriptions outline three different levels of system configurations.

The first two examples, in Figure 12, show that **mainframe** refers more to a **style** of computing rather than to unique hardware. Two different systems are illustrated and neither uses mainframe hardware in the generally accepted sense of the term.

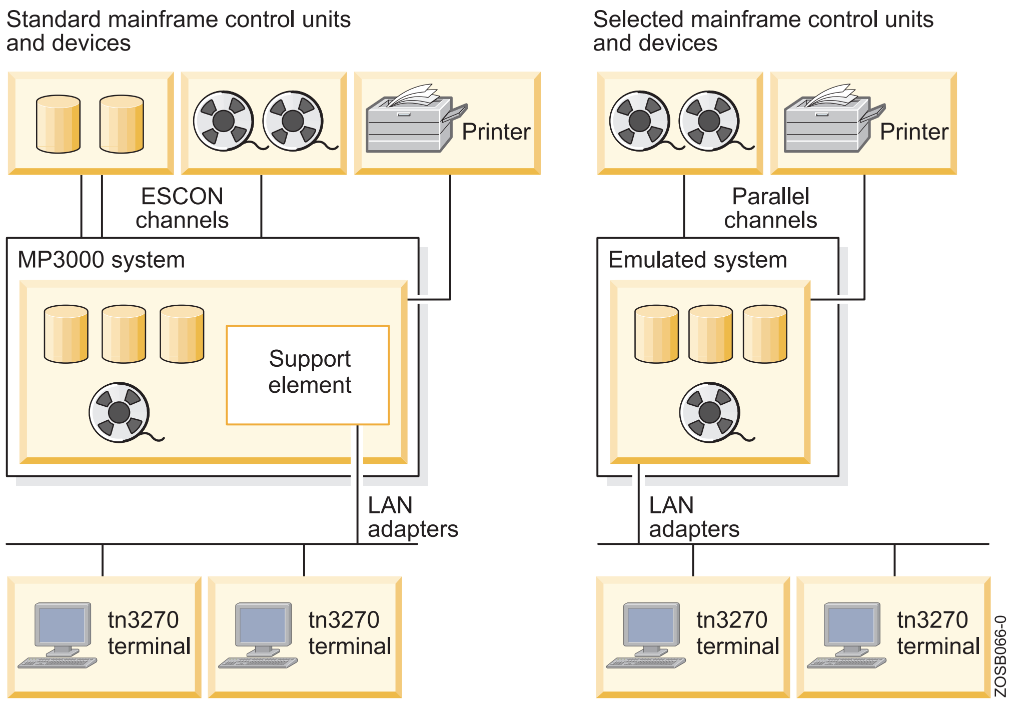

Figure 12. Very small mainframe configurations

The first system illustrated is an IBM Multiprise® 3000 system (MP3000). which IBM recently withdrew from marketing. It was the smallest S/390 system produced in recent years. The MP3000 has one or two S/390 processors plus a SAP processor. It also has internal disk drives that can be configured to operate as normal IBM 3390 disk drives. A minimal internal tape drive is normally used for software installation. The MP3000 can have a substantial number of ESCON or parallel channels for connection to traditional external I/O devices.

The MP3000 is completely compatible with S/390 mainframes, but lacks later zSeries features. It can run early versions of z/OS and all prior versions of the operating system. It is typically used with z/VM or z/VSE operating systems.

The second system shown, the emulated zSeries system, has no mainframe hardware. It is based on a personal computer (running Linux or UNIX) and uses software to emulate z/OS. Special PCI channel adapters can be used to connect to selected mainframe I/O devices. The personal computer running the emulated z/OS can have substantial internal disks (typically in a RAID array) for emulating IBM 3390 disk drives.

Both of these systems lack some features found in ″real″ mainframes. Nevertheless, both are capable of doing quality work. Typical application software cannot distinguish these systems from real mainframes. In fact, these are considered mainframes because their operating systems, their middleware, their applications, and their style of usage are the same as for larger mainframes. The MP3000 can be configured with LPARs and might run both test and production systems. The emulated system does not provide LPARs, but can accomplish much the same thing by running multiple copies of the emulator software.

A key attraction of these systems is that they can be a ″mainframe in a box.″ In many cases no external traditional I/O devices are needed. This greatly reduces the entry-level price for a mainframe system.

Figure 13 on page 37 shows a modest mainframe system and shows the typical external elements needed. The particular system shown is an IBM z890 system with two recent external disk controllers, a number of tape drives, printers, LAN attachments, and consoles.

This configuration is somewhat idealized in that no older devices are involved. The systems outlined here might have a number of LPARs active, for example:

- A production z/OS system running interactive applications.
- A second production z/OS devoted to major batch applications. (These applications also could be run in the first LPAR, but some installations prefer a separate LPAR for management purposes.)
- A test z/OS version for testing new software releases, new applications, and so forth.
- One or several Linux partitions, perhaps running Web-related applications.

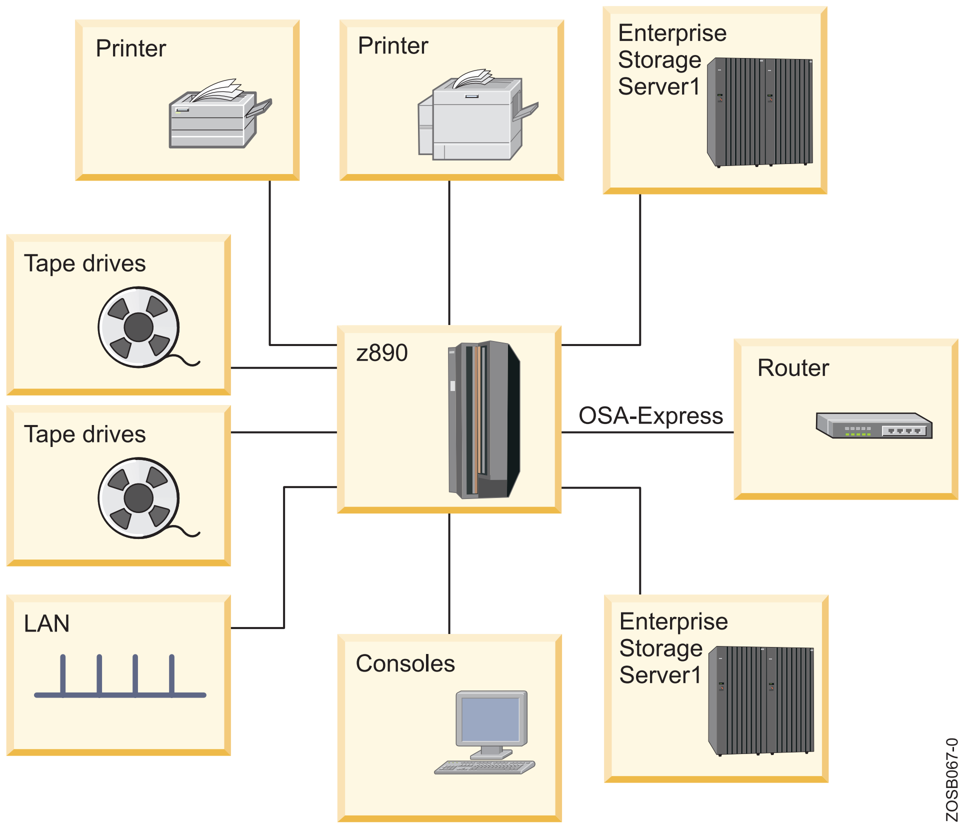

Figure 13. Medium mainframe configuration

The disk controllers in Figure 13 contain a large number of commodity drives running in multiple RAID configurations. The control unit transforms their interfaces to appear as standard IBM 3390 disk drives, which is the most common disk appearance for mainframes. These disk control units have multiple channel interfaces and can all operate in parallel.

Figure 14 on page 38 shows a larger mainframe, although this is still a modest configuration when compared to a **large** mainframe installation. This example is typical in that both older and newer mainframes are present, along with channel switches allowing all systems to access most I/O devices. Likewise, new and older disk controllers (and devices) and tape controllers (and devices) are present. The total system is in a modest Parallel Sysplex configuration.

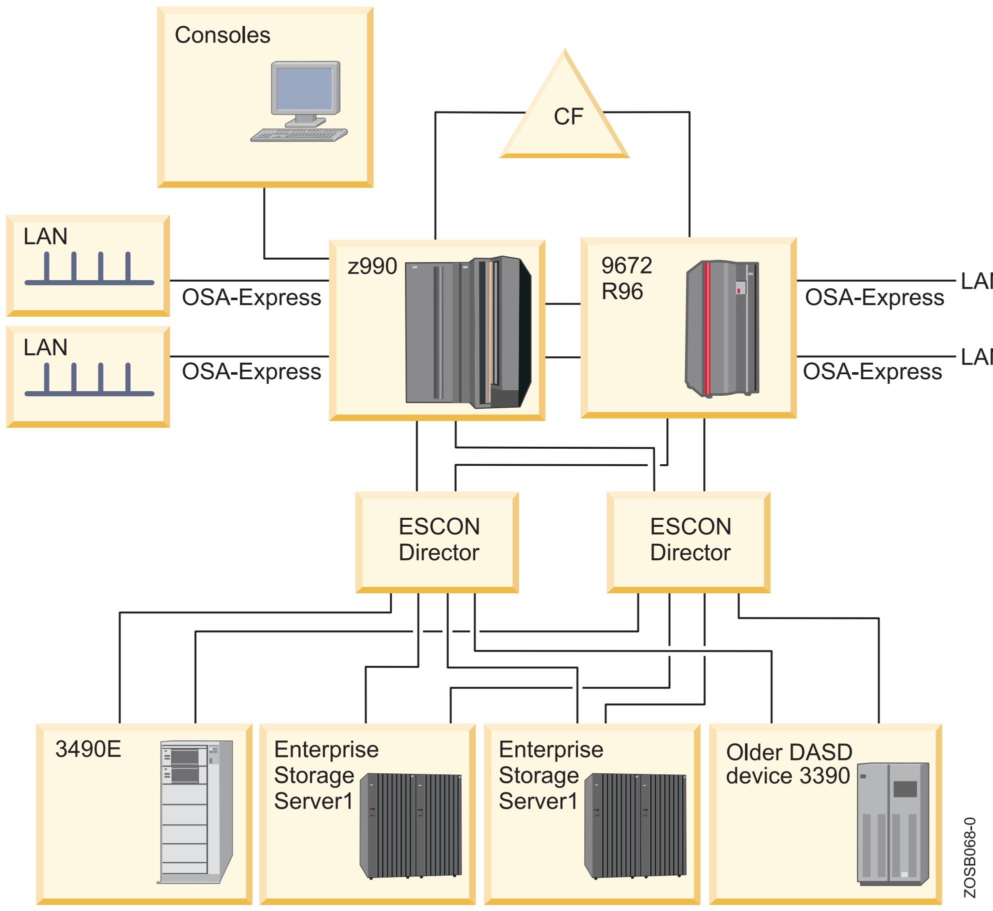

Figure 14. Moderately large mainframe configuration

Briefly, the devices in Figure 14 include:

- An IBM 3745, which is a communications controller optimized for connection to remote terminals and controllers, and LANs. A 3745 appears as a control unit to the mainframe.
- IBM 3490E tape drives, which, though somewhat outdated, handle the most widely used mainframe-compatible tape cartridges.
- A sixth-generation mainframe design (G6).
- A newer z990 mainframe.
- An Enterprise Storage Server (ESS).
- ESCON directors.
- OSA Express connections to several LANs.
- A coupling facility (CF), which is shown as a separate box, but it might be an LPAR in the mainframe.

Chapter 4. Clustering on the mainframe: Parallel Sysplex and other techniques

Since the early 1970s mainframes have been designed as multiprocessor systems, even when only a single processor is installed. All but the smallest mainframe installations typically use clustering techniques, although they do not normally use the terms **cluster** or **clustering**.

A clustering technique can be as simple as a shared DASD configuration where manual control or planning is needed to prevent unwanted data overlap. More common today are configurations that allow sharing of locking and enqueueing controls among all systems. Among other benefits, this automatically manages access to data sets so that unwanted concurrent usage does not occur.

The most sophisticated of the clustering techniques is a Parallel Sysplex. This technology allows the linking of up to 32 servers with nearly linear scalability to create a powerful commercial processing clustered system. Every server in a Parallel Sysplex cluster has access to all data resources, and every ″cloned″ application can run on every server. When used with coupling technology, Parallel Sysplex provides a ″shared data″ clustering technique that permits multisystem data sharing with high performance read/write integrity.

Sysplex design characteristics help businesses to run continuously, even during periods of dramatic change. Sysplex sites can dynamically add and change systems in a sysplex, and configure the systems for no single points of failure. Through this state-of-the-art cluster technology, multiple z/OS systems can be made to work in concert to more efficiently process the largest commercial workloads.

Clustering technique: Basic shared direct access storage devices (DASD)

A basic shared DASD system is typically used where the Operations staff controls which jobs go to which system and ensures that there is no conflict such as both systems trying to update the same data at the same time. Despite this limitation, a basic shared DASD environment is very useful for testing, recovery, and careful load balancing.

A basic shared DASD environment is illustrated in Figure 15 on page 40; a real system would have many more control units and devices. The figure shows z/OS images, but these could be any earlier version of the operating system. This could be two LPARs in the same system or two separate systems; there is absolutely no difference in the concept or operation.

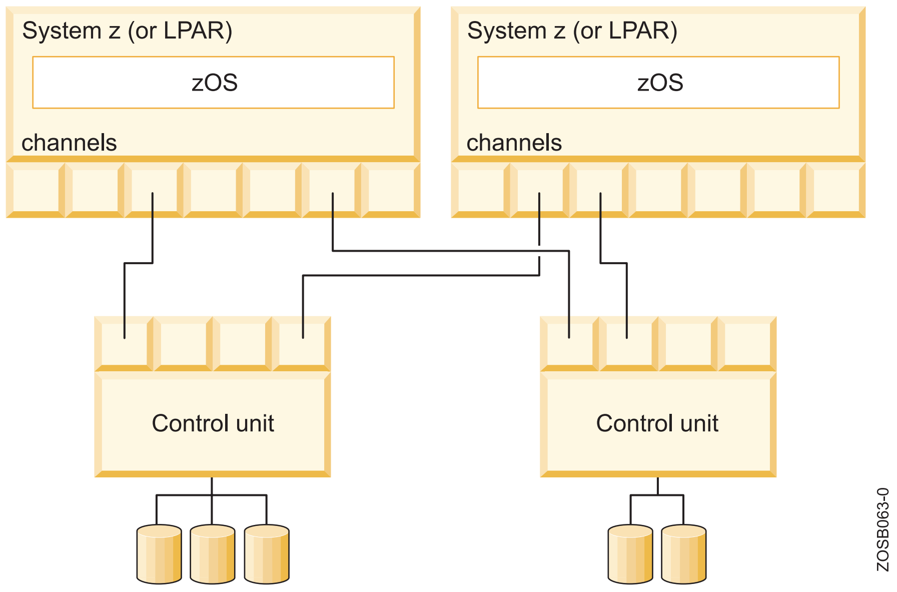

Figure 15. Basic shared DASD

The capabilities of a basic shared DASD system are limited. The operating systems automatically issue RESERVE and RELEASE commands to a DASD before updating the volume table of contents (VTOC), or catalog. The VTOC and catalog are structures that contain metadata for the DASD, indicating where various data sets reside. The RESERVE command limits access to the entire DASD to the system issuing the command, and this lasts until a RELEASE command is issued. These commands work well for limited periods (such as updating metadata). Applications can also issue RESERVE/RELEASE commands to protect their data sets for the duration of the application. This is not automatically done in this environment and is seldom done in practice because it would lock out other systems' access to the DASD for too long.

Other types of devices or control units can be attached to both systems. For example, a tape control unit, with multiple tape drives, can be attached to both systems. In this configuration the operators can then allocate individual tape drives to the systems as needed.

Clustering technique: Channel-to-channel (CTC) rings

The next level of clustering uses basic shared DASD, but also has two **channel-to-channel** (CTC) connections between the systems. This clustering technique is known as a **CTC ring**.

The ring aspect is more obvious when more than two systems are involved. Figure 16 on page 41 illustrates this clustering technique.

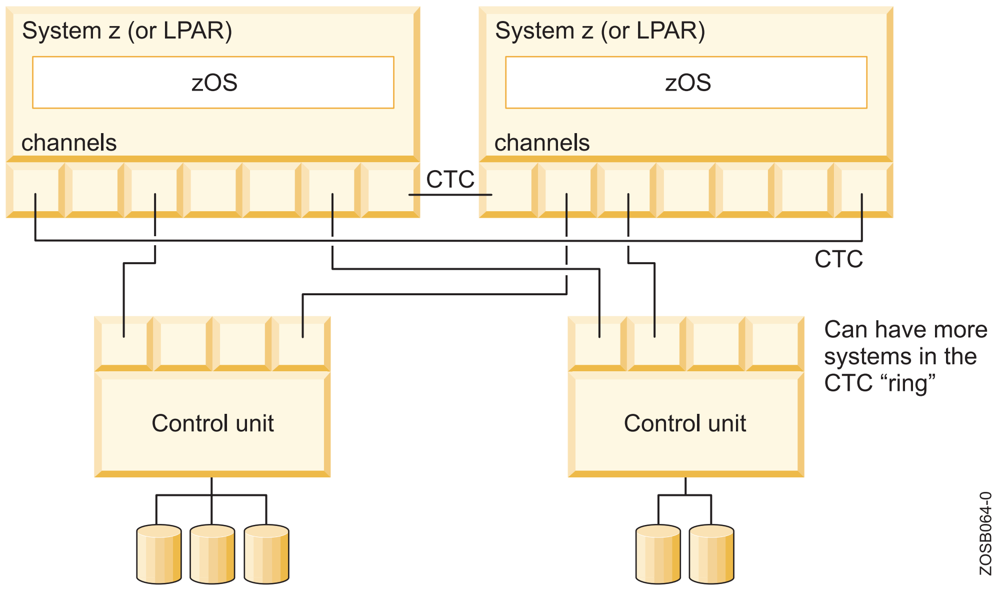

Figure 16. Basic sysplex

z/OS can use the CTC ring to pass control information among all systems in the ring. The information that can be passed this way includes:

- Usage and locking information for data sets on disks. This allows the system to automatically prevent unwanted duplicate access to data sets. This locking is based on JCL specifications provided for jobs sent to the system.
- Job queue information such that all the systems in the ring can accept jobs from a single input queue. Likewise, all systems can send printed output to a single output queue.
- Security controls that allow uniform security decisions across all systems.
- Disk metadata controls so that RESERVE and RELEASE disk commands are not necessary.

To a large extent, batch jobs and interactive users can run on any system in this configuration because all disk data sets can be accessed from any z/OS image. Jobs (and interactive users) can be assigned to whichever system is most lightly loaded at the time.

When the CTC configurations were first used, the basic control information shared was locking information. The z/OS component doing this is called global resource serialization function; this configuration is called a GRS ring. The primary limitation of a GRS ring is the latency involved in sending messages around the ring.

A different CTC configuration was used before the ring technique was developed. This required two CTC connections from every system to every other system in the configuration. When more than two or three systems were involved, this became complex and required a considerable number of channels.

The earlier CTC configurations (every-system-to-every-system or a ring configuration) were later developed into a basic **sysplex** configuration. This includes control data sets on the shared DASD. These are used for consistent operational specifications for all systems and to retain information over system restarts.

Configurations with shared DASD, CTC connections, and shared job queues are known as **loosely coupled systems**. Multiprocessors, where several processors are used by the operating system, are sometimes contrasted as **tightly coupled systems** but this terminology is seldom used. These are also known as Symmetrical MultiProcessors (SMPs); the SMP terminology is common with RISC systems, but is not normally used for mainframes.

Clustering technique: Parallel Sysplex

A **sysplex** is a collection of z/OS systems that cooperate, using certain hardware and software products, to process work. It is a clustering technology that can provide near-continuous availability.

A conventional large computer system also 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 growth potential and level of availability in a sysplex. The sysplex increases the number of processing units and z/OS operating systems that can cooperate, which in turn increases the amount of work that can be processed. To facilitate this cooperation, new products were developed and old products were enhanced.

A **Parallel Sysplex** is a sysplex that uses multisystem data-sharing technology. It allows direct, concurrent read/write access to shared data from all processing nodes (or servers) in the configuration without impacting performance or data integrity. As many as 32 servers can concurrently cache shared data in local processor memory through hardware-assisted cluster-wide serialization and coherency controls. As a result, work requests that are associated with a single workload, such as business transactions or database queries, can be dynamically distributed for parallel execution on nodes in the sysplex cluster based on available processor capacity.

Figure 17 on page 43 shows the visible parts of a Parallel Sysplex, namely the hardware. These parts are the key components of Parallel Sysplex as implemented in the hardware architecture.

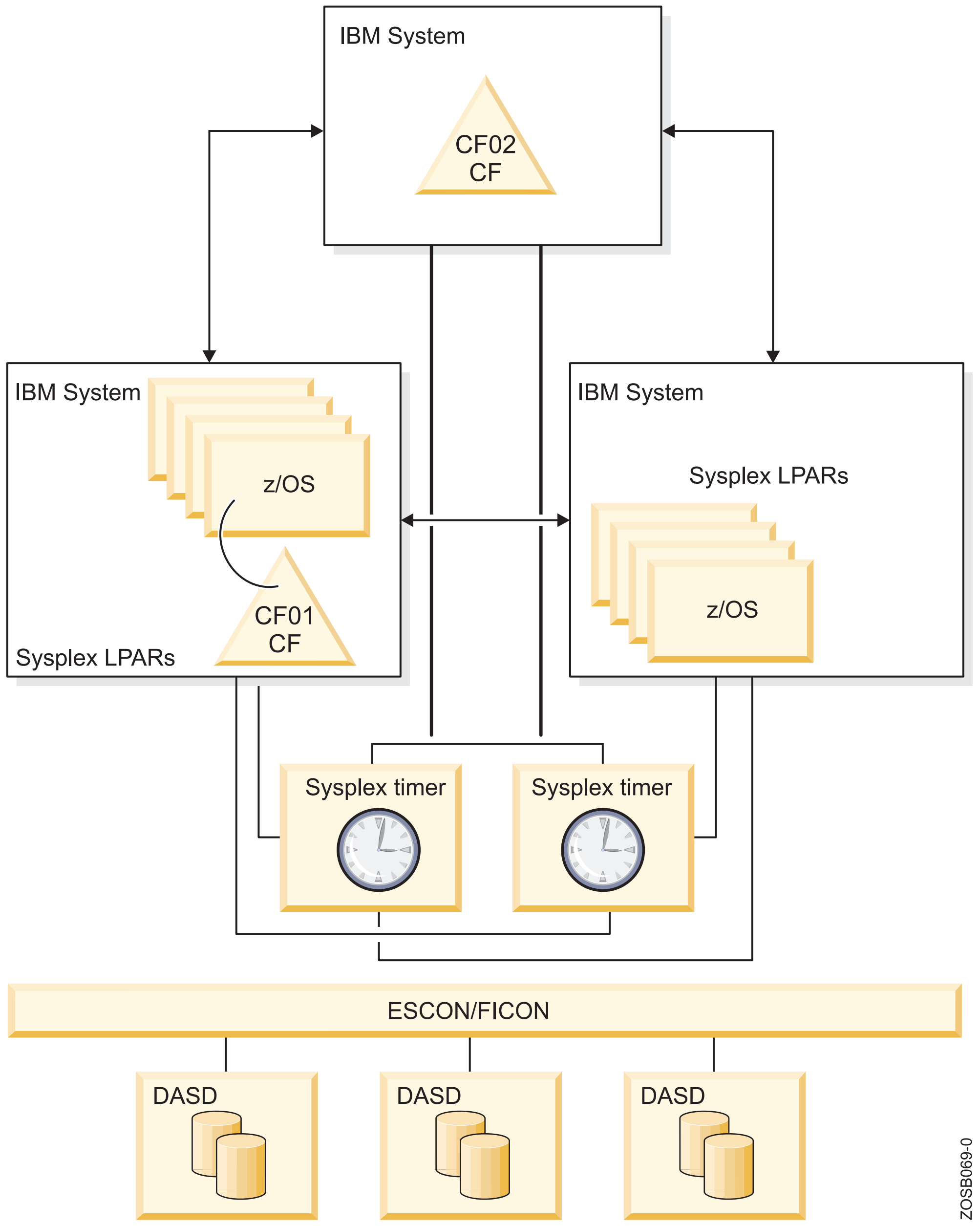

Figure 17. Parallel Sysplex hardware overview

A Parallel Sysplex relies on one or more coupling facilities (CFs). A coupling facility is a mainframe processor, with memory and special channels, and a built-in operating system. It has no I/O devices, other than the special channels, and the operating system is very small.

A CF functions largely as a fast scratch pad. It is used for three purposes:

- Locking information that is shared among all attached systems
- Cache information (such as for a database) that is shared among all attached systems
- Data list information that is shared among all attached systems

The information in the CF resides in memory and a CF typically has a large memory. A CF can be a separate system or an logical partition (LPAR). Figure 18 on page 44 on page 44 illustrates a small Parallel Sysplex with two z/OS images. Again, this whole configuration could be in three LPARs of a single system, in three separate systems, or in a mixed combination.

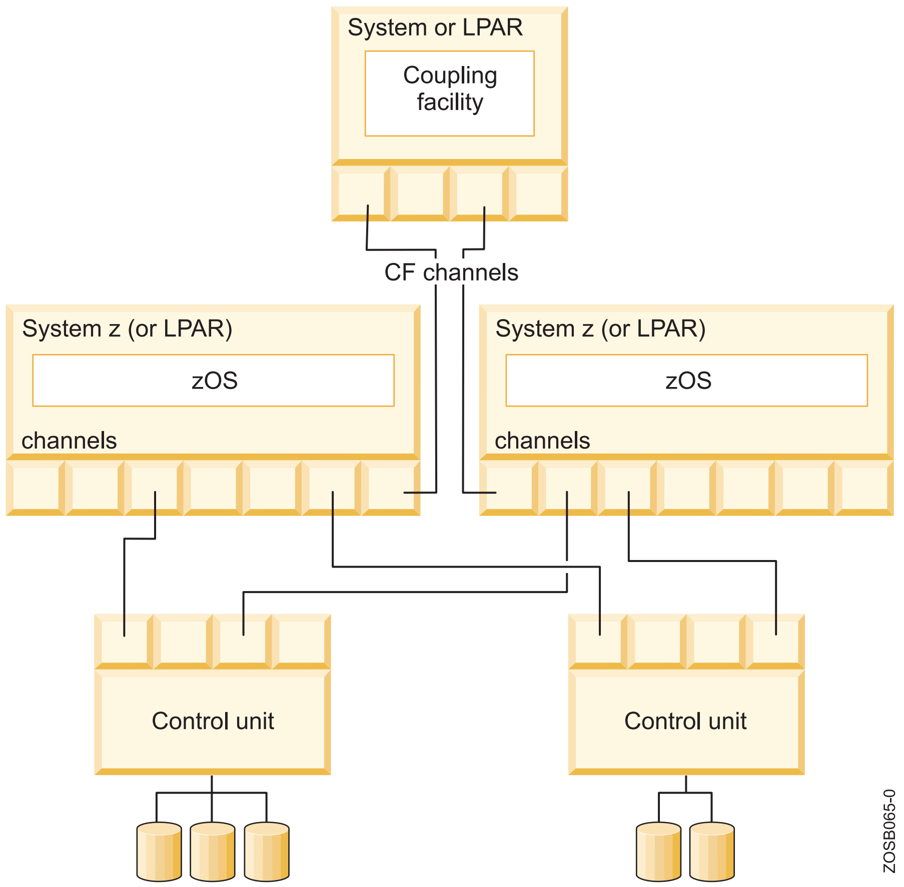

Figure 18. Parallel Sysplex with two z/OS images

In many ways a Parallel Sysplex system appears as a single large system. It has a single operator interface (which controls all systems). With proper planning and operation (neither of which is trivial), complex workloads can be shared by any or all systems in the Parallel Sysplex, and recovery (by another system in the Parallel Sysplex) can be automatic for many workloads.

Another unique advantage of using Parallel Sysplex technology is the ability to perform hardware and software maintenance and installation in a nondisruptive manner. Through data sharing and dynamic workload management, servers can be dynamically removed from or added to the cluster, allowing installation and maintenance activities to be performed while the remaining systems continue to process work. Furthermore, by adhering to the IBM software and hardware coexistence policy, software or hardware upgrades— or both— can be introduced one system at a time. This capability allows customers to roll changes through systems at a pace that makes sense for their business.

The ability to perform rolling hardware and software maintenance in a nondisruptive manner allows business to implement critical business function and react to rapid growth without affecting customer availability.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Programming interface information

This book documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

Printed in USA