System z and z/OS unique Characteristics

Wilhelm G. Spruth

WSI Technical Report
WSI-2010-03

Version 1.0, April 8, 2010

Wilhelm G. Spruth
Wilhelm Schickard Institute for Computer Science
Sand 13
D-72076 Tuebingen
Germany

email: spruth@informatik.uni-tuebingen.de
mobil: 0049-0172-8051-485

ISSN 0946-3852

Abstract

Many people still associate mainframes with obsolete technology.

Surprisingly, the opposite is true. Mainframes feature many hardware, software, and system integration technologies, that are either not at all, or only in an elementary form, available on other server platforms. On the other hand, we know of no advanced server features which are not available on mainframes.

This paper lists some 40 advanced mainframe technologies. There is a short description of each item together with a literature reference for more information.

Table of content

1. Introduction

1.1 Definition

There exist two different types of large systems.

Scientific large systems are optimized for maximum processing power (e.g. the petaflop computer). They use the same components found in low end servers and personal computers (commercial parts), and use similar operating systems. For example, the 10 fastest computer systems (November 2009) all use popular microprocessors manufactured in large quantities for other purposes [01].

Commercial large systems are used by industry and government and form the backbone of their IT infrastructure. According to a definition by Gartner Research they are characterised by a list price larger than 250 000 $. Commercial large systems use many advanced hardware- and software technologies not found in smaller machines.

Mainframes are commercial large systems that implement the IBM „System z“ Architecture. The dominating operating system is called “z/OS”. This paper contains a listing of leading edge hardware- and software technologies found exclusively in mainframes.

1.2 The Hype

For a long time, mainframes have been labeled technological dinosaurs.

The Death of the Mainframe

***A fairly well accepted notion in computing is that the mainframe is going the way of the dinosaur*.**
Forbes, March 20, 1989

The mainframe computer is rapidly being turned into a technological Dinosaur...
New York Times, April 4, 1989

***On March 15, 1996, an InfoWorld Reader will unplug the last mainframe*.**
Stewart Alsop, InfoWorld 1991

...the mainframe seems to be hurtling toward extinction.
New York Times, Feb. 9, 1993

Its the end of the end for the mainframes
George Colony, Forrester Research,
Business Week, Jan. 10, 1994

The above list reproduces some statement and predictions by "Industry experts".

The above advertisement was shown at the CEBIT, Hannover, March 13-20, 2002, promoting the Sun Fire 15K Server as "The fastest commercial Computer".

The picture shows an old sick women in front of an oil slick polluted beach. The caption reads: 30 years ago she was the trend to follow. Exactly like your mainframe.

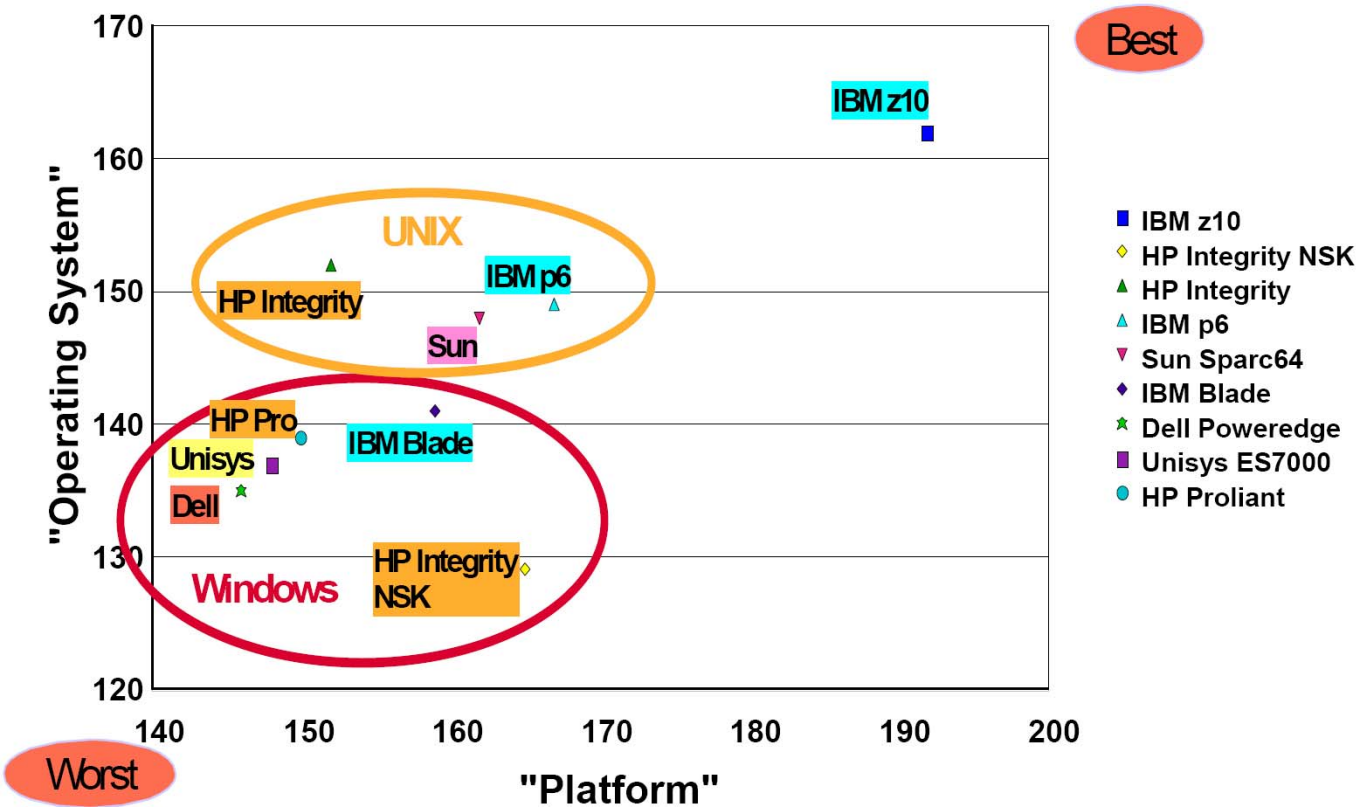
This is a terrific ad, communicating the message in a very forceful way. Unfortunately, the message was wrong.

Installations who were sold on this promotion were subsequently in for a bad surprise. The Sun Fire line of systems experienced many reliability and availability problems, some of them due to a missing ECC at the L2 cache [02]. Market share decreased, and it took Sun Microsystems many years to recover.

1.3 The Reality

According to Gartner Research, Commercial Large Systems are defined by having a list price of more than $ 250 000.- . For a long time, Gartner has been publishing an annual report (Server Scorecard Evaluation Model), which evaluates the features of large systems. Mainframes always scored No. 1, both in total score as well as in nearly all individual characteristics [03].

One of the major reasons is leading edge technology. Mainframes feature many software and hardware characteristics not available on any other server platform. On the other hand, we know of no Unix or Windows server characteristics not available on mainframes, including all modern Java, Web Services, and SOA, developments.



Gartner's platform positioning 4Q08

Gartner, Server Scorecard Evaluation Model version 5, 4Q08, [04]

High end commercial systems have a very unique set of requirements, that cannot really be met using blade systems assembled from low cost commercial components, and their corresponding software (operating systems, subsystems, IDE, etc.).

Thus there will always be a separate and unique "mainframe" product line. As other systems implement existing mainframe features, mainframes will implement new and advanced functions to meet customer requirements.

This has been the trend for the last decades. In many instances, new technologies were first introduced on mainframes, and adopted later – sometimes decades later – on other platforms. I/O control units, virtualization, and the coupling facility are particiular examples.

2. Leading Edge Technologies

The many leading edge mainframe technologies are not well summarized. We list here technological features and facilities not (or only in an elementary form) available on other platforms.

2.1 Architecture

The mainframe architecture was originally developed in 1964. We owe it to the genius of three unusually gifted scientists:

Gene Amdahl,
Gerry Blaauw,
Fred Brooks,

and the management competence of the then IBM Vice President of Development, Bob O. Evans.

Two articles in Fortune magazine describe the dramatic development history of the System /360 [05], [06].

This is the surprising observation: Amdahl, Blaauw, and Brooks invented the S/360 architecture in 1964 [07]. Nobody has been able to improve the basic design during the following nearly 50 years! Many people have tried multiple times to develop an improved hardware architecture, claiming significant performance advantages. No development has ever been proven superior, and many (e.g. Burroughs B5000, VAX) have been a disappointment. In their first edition of the Alpha Architecture Handbook, the Digital Equipment engineers recognized this fact, and emphasized that the Alpha design was based on the principles laid down by Amdahl, Blaauw, and Brooks [08].

Mainframes have a very clean instruction set architecture. It is the only architecture in use today, that reasonably can be programmed in machine code. According to an unpublished internal IBM report, System z code is 20 – 30 % more compact than IA32, Sparc, and Itanium code. This is mostly due to an effective set of 16 bit instructions, while the maximum instruction length is limited to 48 bit.

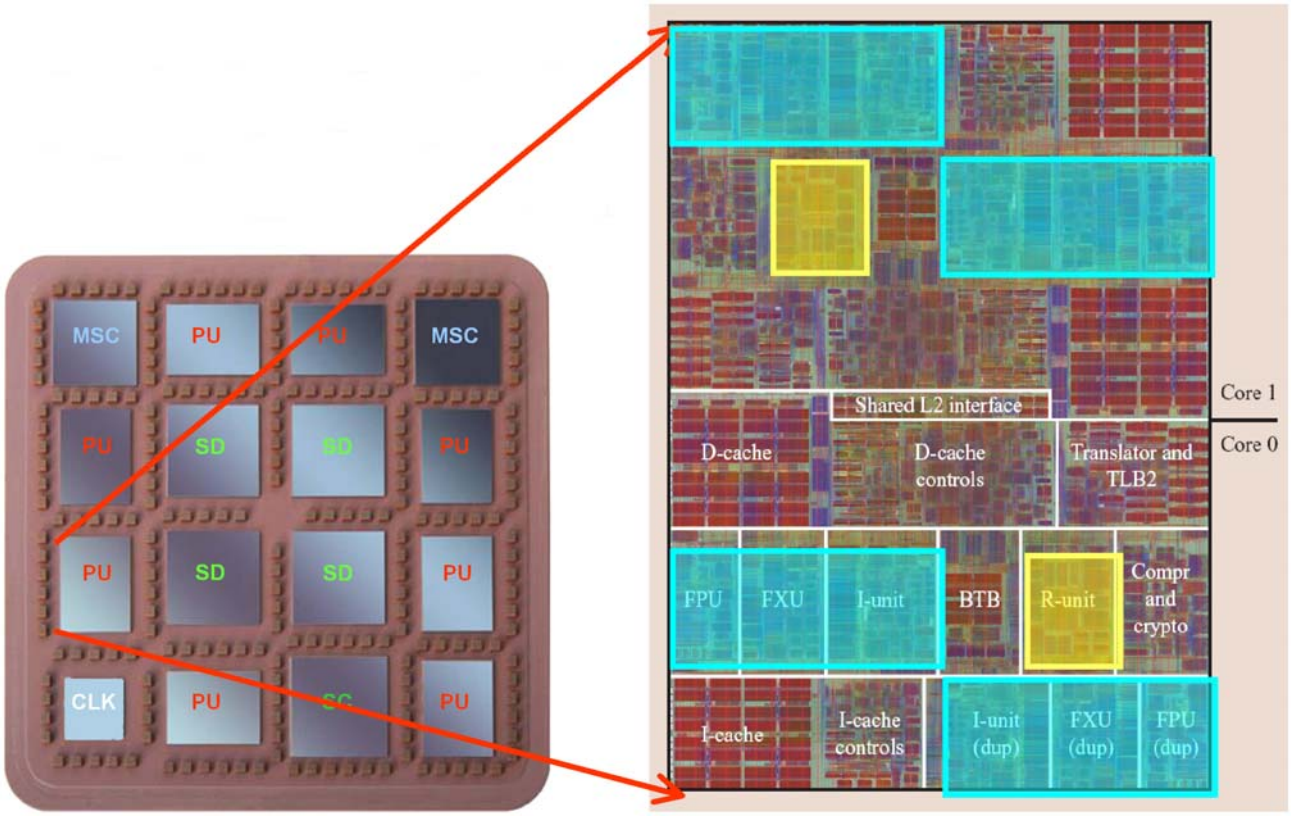
The result is a more effective L1 and L2 cache utilization, and reduced bandwidth requirements for cache – CPU code transfers.

2.2 Avalability and Reliability

Very high reliability and availabilty is one of the most important mainframe characteristics. Although very little hard data are available, there appears to be general agreement, that mainframe hardware and software feature indeed superior high reliablility and availability characteristics. IBM claims an availability for a large mainframe configuration of 99.999 %. This means a downtime of 5 mintutes/year, or less than ½ hour during an assumed lifetime of 5 years.

There is not a single feature responsible for high availability and reliability. Rather, there are hundreds or thousands of individual operating system, subsystem and hardware design characteristics that contribute. A few examples are listed below.

2.2.1 Redundancy

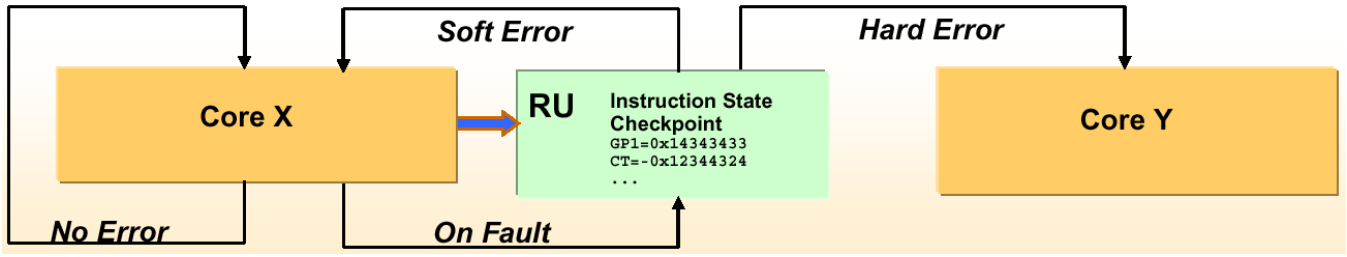


The dual core z9 CPU Chip shown above duplicates the Instrunction unit (I-unit) and both the fixed point (FXU) and floatig point (FPU) execution units to improve error detection and error recovery. This provides a mechanism by which the mirrored execution used for error detection occurs 1 cycle delayed from actual operation, and allows the operation execution logic and circuitry to be optimized for performance [09].

Notice also the Recovery Unit (R-unit).

2.2.2 Recovery Unit (RU)

Nothing is comparable to the Recovery Unit on System z.



The fully architected state of each processor is buffered in the Recovery Unit (R Unit) on each chip. The RU stores a complete ECC protected copy of the processor state. This includes all registers, independent if they are part of the architecture or not. It collects all error messages and supervises recovery actions for nearly all hardware errors [10].

The RU facilitates:

- **Precise core retry for almost all hardware errors,**
- **Dynamic, transparent core sparing in the event of a hard error in a core.**

In addition, the hardware features:

- **Fine - grained redundancy and checking throughout the design,**
- **Hamming Error Code Correction (ECC) on the 2 nd and 3 rd -level caches, store buffers, and the R - Unit state array,**
- **Parity on all other arrays and register files,**
- **Parity or residue checking on data/ address/ execution flow,**
- **Extensive functional, parity, and state checking on control logic,**
- **Over 20 000 error checkers on each chip.**

The Machine check architecture allows precise software recovery, and minimizes system impact in the rare case of an unrecoverable hard error.

It is estimaled that 30 – 40 % of the CPU chip transistor count is used for error checking or recovery functions.

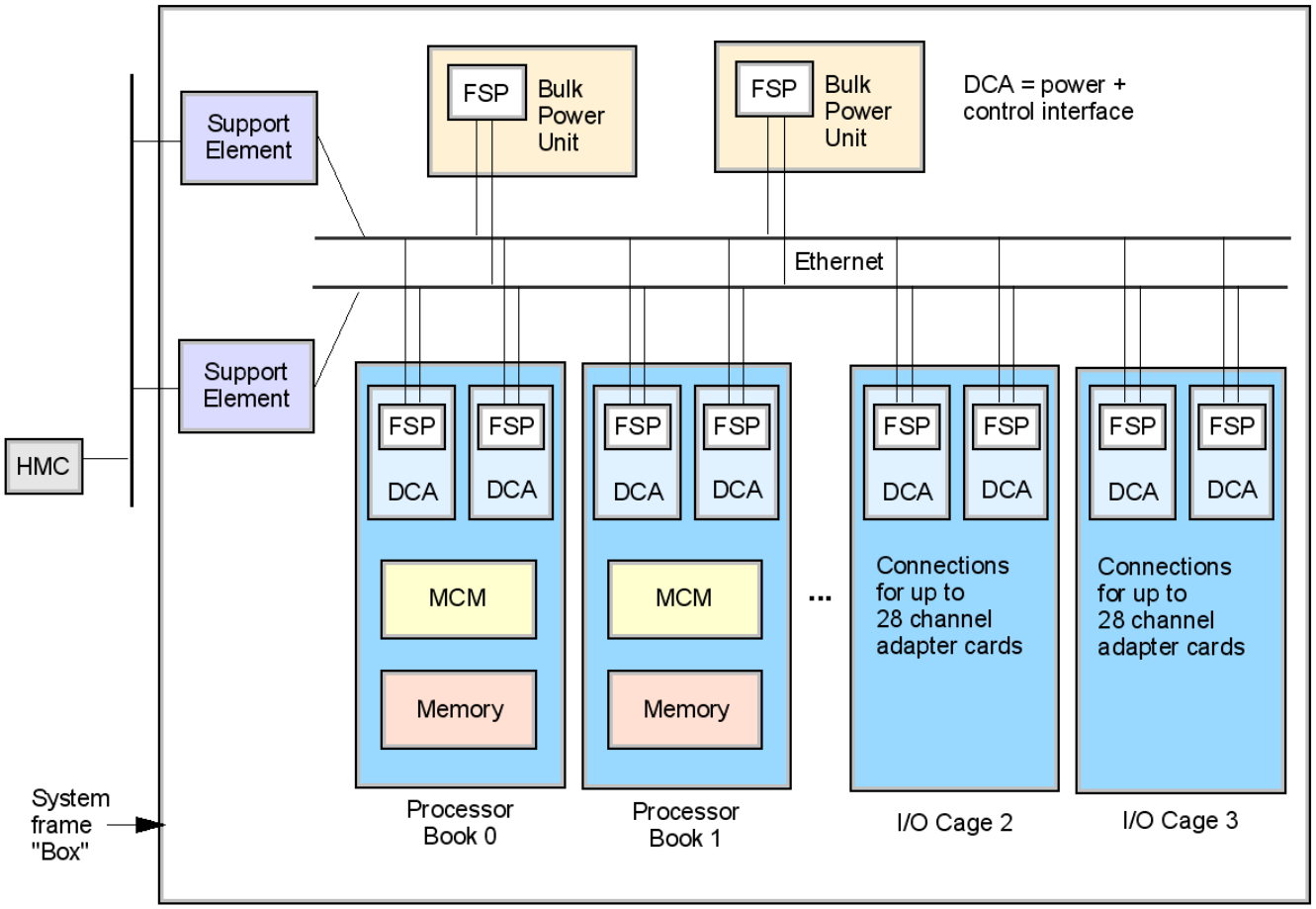
2.2.3 Smart memory card architecture

Parts of memory may stay unused over extensive periods of time. Scrubbing logic uses otherwise unused memory cycles to read out a bit line, perform error correction if necessary, and write the line back again. This way any accumulation of soft errors over time is avoided.

A separate chip represents each bit position in a memory word. Dynamic Redundancy Bit logic activates a spare memory chip in case of a hard bit error.

IBM introduced the Smart memory card architecture in the mid-1980s.

2.2.4 Support Element



The Support Element (SE) is the central point of control in each Mainframe system. In todays z9 and z10 systems, the SE is implemented by a Think Pad lap top. It is permanently mounted into the mainframe box, and cannot be removed.

When you start a PC, during the power-on sequence you are given the option to hit a special key to get you into the BIOS menu. Consider the Support Element to provide a somewhat similar function, except that it is available during operation and not only during start up.

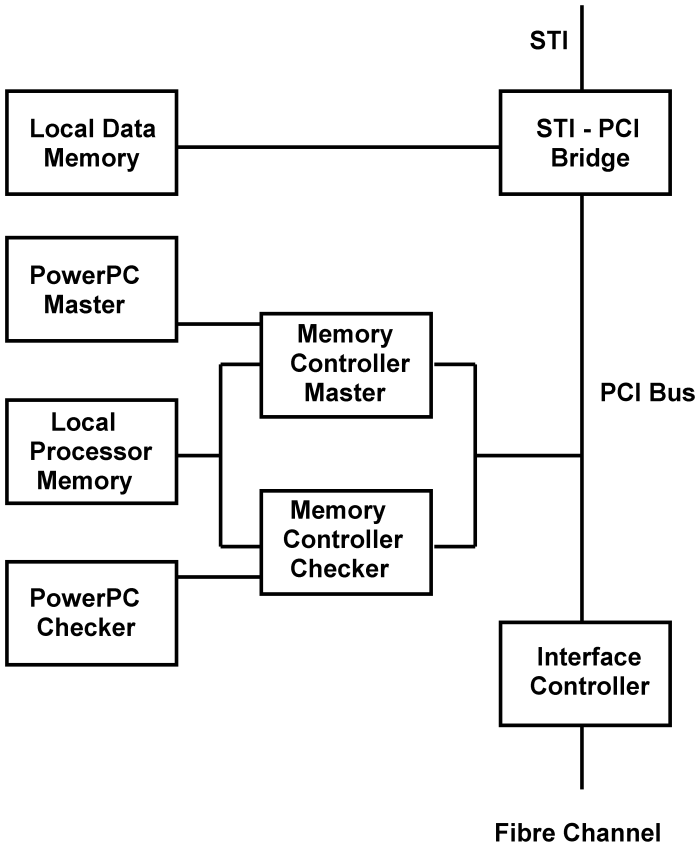
During start up, the SE initialises the system, and loads several GByte of microcode and control tables from its disk, including the PR/SM Hypervisor. You boot an opersting system in a particular LPAR (see section 2.8.3) by issuing the IPL command on the SE.

The support element connects to an Ethernet that is internal to the mainframe box. It connects the SE to all books, all I/O cages, and all power supplies, continuously collects diagnostic information, and transmits them to an IBM service center. Thus it may occur, that an IBM service technician shows up to fix a problem of which the system administrator is not yet aware.

The SE is connected via an encrypted link to a Hardware Management Console (HMC). The HMC is a special PC, which allows remote access to the SE. If a system fault occurs, the HMC places a call to the IBM Support System, alerting IBM and transferring error information and logs to reduce the impact of unplanned outages [09].

For reliability reasons, the SE and the internal Ethernet are available in duplicate.

2.2.5 I/O adapter card



99,999 % availability requires functions not affordable in low end systems.

The System z I/O adapter cards feature extensive reliability features and an unusual amount of error control . As an example, the z9 Fibre Channel card is shown. It uses the same Fibre Channel PCI Bus chip found in many PCs (Interface Controller). There it is the only complex chip on a PC adapter card.

The System z I/O card is more complex. Contrary to other systems, the PCI Bus output of this chip is processed by a PowerPC processor. It helps to prevent erroneous data transmission. For additional failure protection there is a second PowerPC comparing results on each cycle. The same applies to the PowerPC memory controller.

The most complex chip is the STI-PCI Bridge chip. It prevents the DMA controller to use incorrect memory addresses in case the address gets corrupted [11].

2.2.6 Software stability

z/OS operating system and subsystem software is very modular, with few, if any changes occuring to most modules between version upgrades. For example, there have been essentially no changes or modifications to the z/OS scheduler/dispatcher during the last 20 years.

z/OS has recovery code called “Functional Recovery Routines” for every major routine in the system. If a problem were to occur, the operating system has the ability to refresh the code and keep the system up and running.

The automated restart manager (ARM) can restart failing software components.

Characteristics beyond failure detection allow to predict problems before they occur. The operating system has features to learn heuristically from its own environment to anticipate and report on system abnormalities using Predictive Failure Analysis (PFA) [12].

Essentially, PFA can help to avoid soft failures leading to outages. Unlike typical problems or hard failures that have a clear start and a clear cause, soft failures are caused by abnormal, but allowable behavior. Because the cause of the problem is dependent on a certain sequence or combination of events that are unique and infrequent, a solution is often difficult to determine. Multiple atypical, but legal actions performed by components on the z/OS image cause most soft failures. By design, most components of z/OS are stateless and are therefore unable to detect soft failures caused by atypical behavior.

A classic example is the exhaustion of common storage usage. A low priority authorized task obtains significantly more common storage than usual. Then, a critical authorized system component fails while attempting to obtain a normal amount of common storage. Although the problem occurs in the second critical component, this second component is actually the victim of a problem caused by the first component.

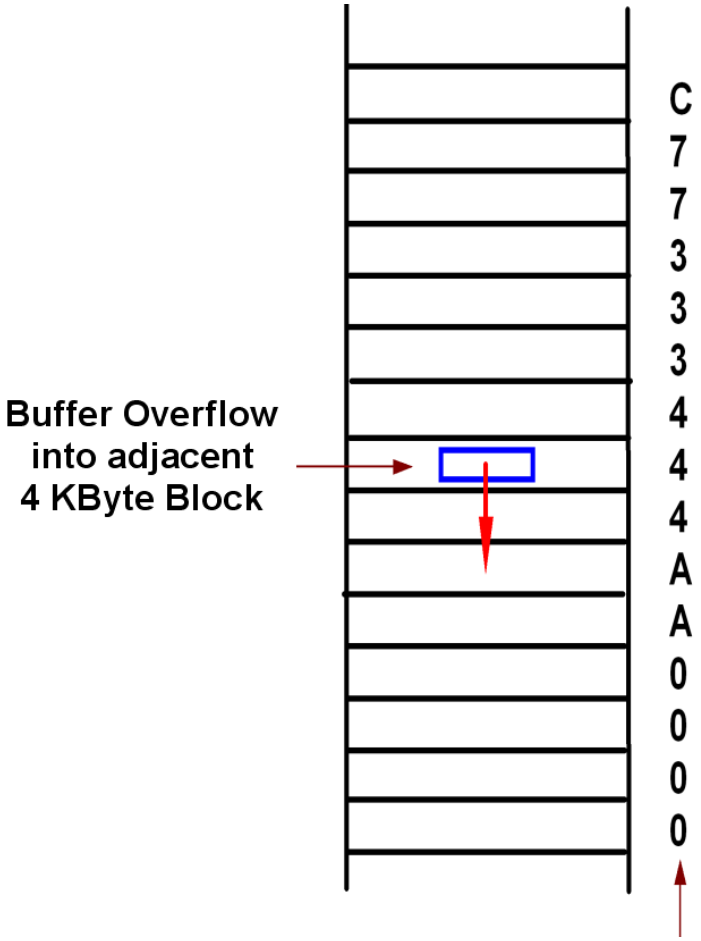
2.2.7 GDPS

The data center environment has become more complex, with applications spanning multiple platforms and operating systems. Managing a disaster recovery environment with the requirement to coordinate database recovery between the z/OS and UNIX systems data can be very challenging. The Geographically Dispersed Parallel Sysplex (GDPS), the System z premier continuous availability/disaster recovery solution, provides the required coordination, see section 2.10.8.

2.3 Security

2.3.1 Hardware key protection

For protection purposes, main storage is divided into blocks of 4096 bytes. A four-bit storage protection key is associated with each block. When data are accessed in a storage block, the storage block key is compared with a protection key. A specific protection key is assigned to each process. Problem state processes usually share protection key hex08. Supervisor state processes and system processes often use different protection keys.



A Buffer overflow into an adjacent 4 KByte block can only occur, if this block happens to have the same storage protection key assigned. A buffer overflow from a user program into a supervisor area is essentially impossible.

Hardware key protection is only available on System z. It effectively blocks buffer overflow and unauthorized kernel access [13].

2.3.2 Cryptography support

There exist two hardware cryptography implementations on system z:

- **The CP Assist for Cryptographic Functions (CPACF) is a facility implemented as a part of each CPU chip.**
- **The Crypto Express 2 (CEX2) card is external to the CPUs.**

Each processor chip includes assist processor functions in support of cryptography and compression. This CP Assist for Cryptographic Function (CPACF) provides

- **Symmetric DES. TDES and AES hardware encryption and decryption support - with "clear key" only (see below), – and**
- **one-way SHA-1 und SHA-256 hash algorithms.**

The Crypto Express 2 (CEX2) card provides "secure key" functions:

- **Symmetric DES, T-DES and AES encryption/decryption,**
- **Message authentication, hashing,**
- **PIN processing,**
- **Rivest-Shamir-Adelman RSA asymmetric encryption/decryption,**
- **digital signature generation/verification,**
- **random number generation,**
- **Key generation and management.**

"Secure key" means, when encrypting/decrypting data, the corresponding keys themselves are stored in system storage encrypted with a "master key". The corresponding keys are sent with the data to the CEX2 card. There the keys are decrypted with the master key. The master key itself is permanently stored within the CEX2 card and never appears outside the card.

In addition, the Crypto Express 2 (CEX2) card provides clear key RSA functions for SSL/TLS acceleration [14] .

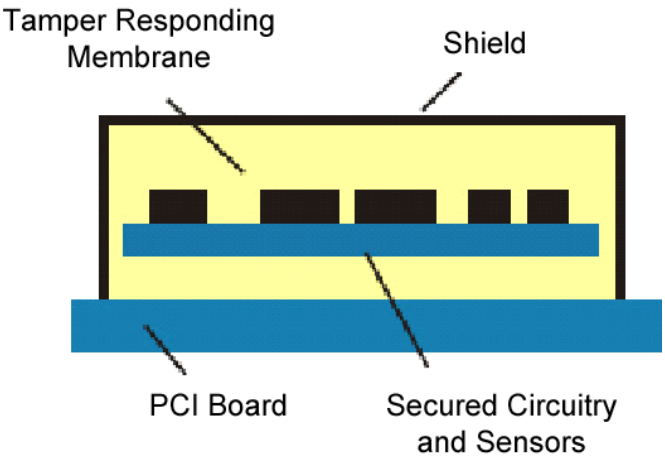
2.3.3 Master Key

The IBM Common Cryptographic Architecture (CCA) introduces the notion of a "secure coprocessor" and a secret "Master key". Master keys provide additional protection for stored keys. With a secure coprocesor, the application keys are kept themselves encrypted under a Master Key when they are outside the physical enclosure of the coprocessor.

There exists only one instance of the Master Key and it resides inside the coprocessor of the CEX2 card. When invoking data encryption, the application passes to the secure coprocessor a copy of the application key to use, which is decrypted under the secret Master key. The coprocessor decrypts the application key inside its physcally secure enclosure and proceeds with data encryption/decryption. The notion of "secure" refers to the fact that in no place outside the coprocessor, be it in system storage or on disk devices, somebody will see the actual value of the application key [15].

The z/OS Integrated Cryptographic Services Facility (ICSF) is a software package that acts as the device interface for the cryptographic hardware on z systems. It supports, in addition to the above, Elliptic Curve Cryptography (ECC), which is regarded as a faster algorithm and a smaller key as compared to similar RSA cryptography, and Internet Key Exchange version 2 (IKEv2) for IPSec dynamic key exchange.

2.3.4 Tamper proof Crypto cards



The Crypto Express 2 (CEX2) card uses a Tamper Resistant Security Module (TSRM). Tamper proof crypto cards are an integrated part of System z.

The CEX2 card is designed to withstand most physical attacks (that is, removing the hardware) as well as the subtlest attacks (that is, measuring the electromagnetic pulses and radiation emitted by the hardware) without revealing any of their secrets. If tampering is detected, the typical response for a TSRM is to erase any secret information (typically, the secret master keys) from its memory.

2.3.5 z/OS Security Management

z/OS Security Management has two major components:

- **The Security Authorisation Facility (SAF) is a Kernel function, that identifies security relevant events,**
- **The Resource Access Control Facility (RACF) is a subsystem running in a system address space (see section 2.5.1).**

SAF provides only minimal functions. It supervises how resources are accessed. Decisions as to permit or deny access to the resources thus identified are usually relegated to an access engine, for example RACF. RACF is supplied by IBM; since it is implemented as a subsystem, it can be easily replaced by a user written or third party supplied security engine [16].

RACF works with a rules engine. Rules can be established in form of a script with if-then-else characterisics. Rules are called profiles and stored in a RACF data base.

There are no known cases of z/OS virus infections or successful hacker attacks.

2.3.6 z/OS Authorized Program Facility

With the OS/370 VS2 release in 1973, IBM established and defined its System Integrity Strategy [17]. One part is the acceptance of Authorized Program Analysis Reports (APARs) for integrity exposures found by customers. Another part is the Authorized Program Facility (APF).

APF allows authorization of system-level programs that need to modify or extend the basic functions of the operating system. Typical applications might include access control software, database managers, online transaction processors, scheduling systems, or tape and DASD storage management.

APF serves as the gatekeeper. It determines which services authorized programs may perform on behalf of non-authorized programs. Some powerful services are restricted to being requested only by authorized programs. This is important since system integrity is lost if non-authorized programs can manipulate authorized programs into performing malicious actions [18].

On other operating systems, components either have, or have not, “root access”, and usually there are too many items that need root access. APF provides a fine grained mechanism to grant only as much authorization as needed.

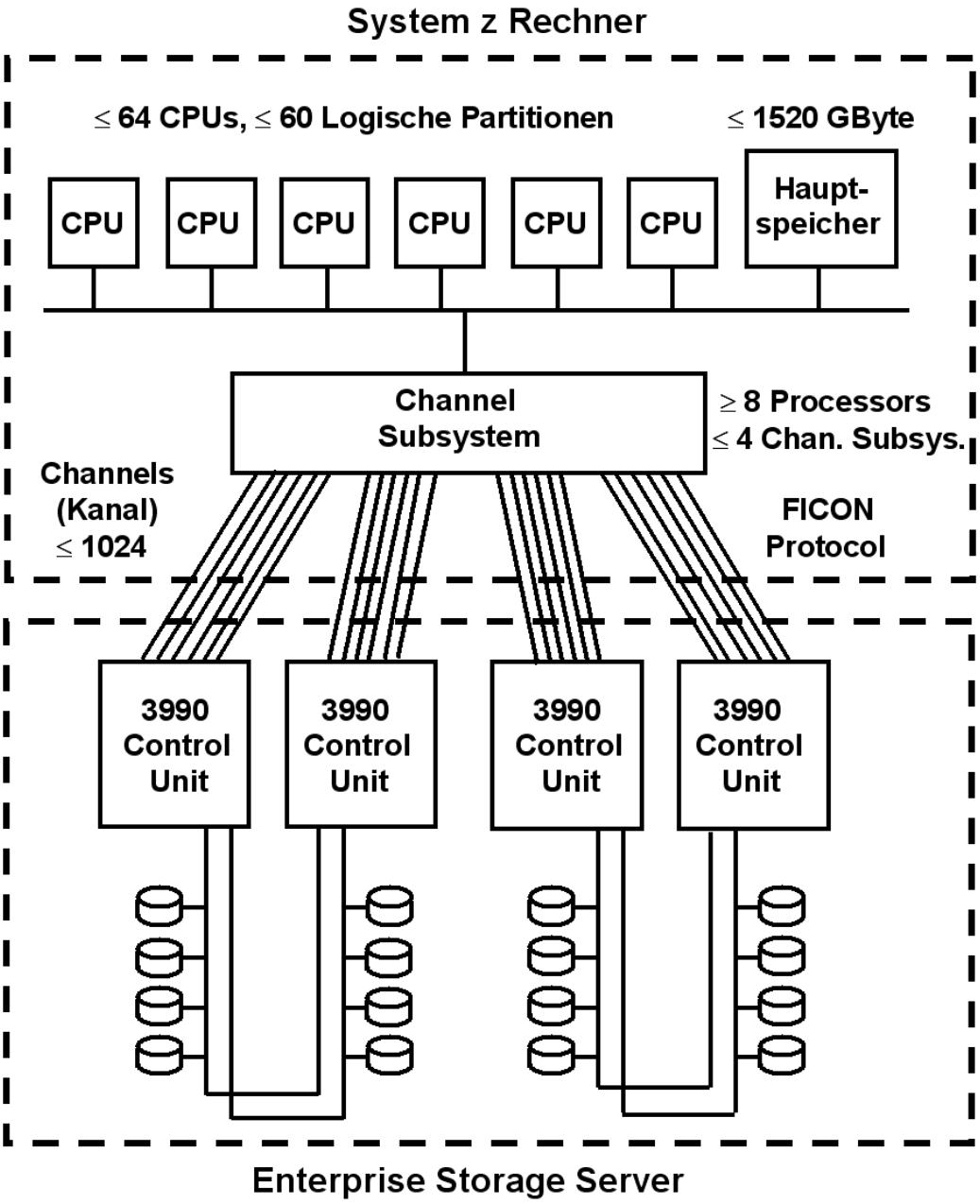
An additional integrity contributor is the System Modification Program (SMP/E). It provides automated program maintenance for z/OS system software and has been available for decades. SMP/E makes certain that Program Temporary Fixes (PTF) prerequisites and other dependencies are met, and allows the testing and rollback of fixes.

2.3.7 Security Updates

We have met so far no z/OS system administrator, who remembers the day he has installed a “security update”. It must have occurred a long time ago.

2.4 Input/Output (I/O)

2.4.1 Control Units



For transaction- and data base applications, mainframes are able to handle a higher data transfer volume between main memory and external storage than any other platform. This appears to be universally accepted, although comparative figures have never been published.

A number of characteristics contribute to the superior I/O Performance:

Mainframes do not know I/O drivers. The z/OS component known as the I/O driver performs only very elemntary generic functions. I/O driver processing, as understood in Unix or Windows platforms, is performed instead by a contol unit [19].

2.4.2 I/O Scheduling

As an example, an I/O Scheduler determines in which sequence a disk access arm serves multiple I/O requests. Algorithmns like Elevator Disk Arm or Shortest Seek First are being used.

Linux (and zLinux) provide 4 different I/O Schedulers: noop, anticipatory, deadline and cfq (Completely Fair Queuing) [20]. On a running system a different scheduler can be changed dynamically for each disk device using the operator command sysX:~ # echo "noop" > /sys/devices/css0/0.0.0004/0.0.2b2b/block/dasda/queue/scheduler.

On a mainframe, I/O scheduling is performed by a control unit and/or the Enterprise Storage Subsystem. These tasks are offloaded from the CPUs. Without the offloading of I/O driver functions, a mainframe would never be able to handle its I/O traffic.

For a discussion of synchronous versus asnchronous I/O, see [21].

2.4.3 Connection Path Management

Mainframes connect to their control units by means of I/O adapter cards and glas fiber connections called channels. In place of Fibre Channel SCSI, the Fibre Channel High Performance FICON for System z (zHPF) Protocoll is used, providing superior bandwidth utilization [22]. Usually, each control unit is connected to the mainframe via multiple (up to 8) channels, as shown above. To improve availability, I/O devices (for example disks) usually connect to two different control units.

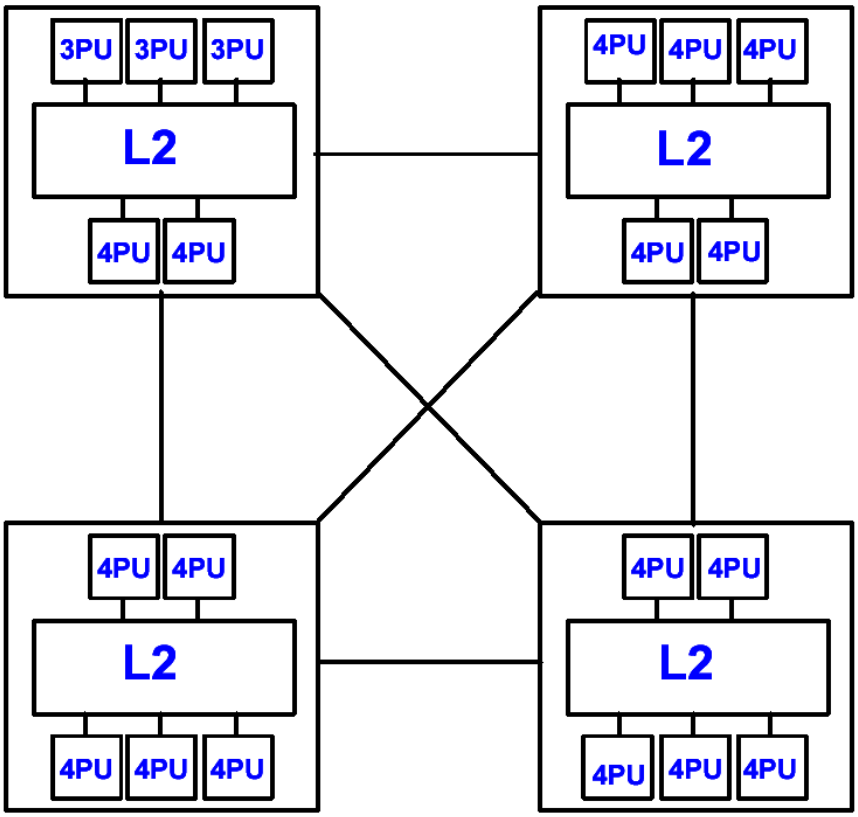
Thus, there exist multiple paths between a CPU and an I/O device. Over a particular channel, multiple I/O operations may be multiplexed. A particular I/O operation may finish on a different Channel than on the one on which it started, and it may switch channels inbetween. In a cluster configuration, independent mainframes may have channel connections to the same control unit. All these capabilities help to facilitate simultaneous access by thousands or ten-thousands of I/O devices to main memory and to provide significant higher I/O performance than available in large Unix systems.

Managing hundreds or thousands of channel connections requires significant processing power. To offload the CPUs, each mainframe has one (or multiple) Channel Subsystem(s) [23] . A channel subsystem optimises the utilisation of available channel paths between CPUs and I/O devices, offloading CPUs from this task. It consists of one or several processors, called System Assist Processors (SAP). SAPs run code outside the reach of the operating system. For this, physical memory contains an area, called the Hardware System Area (HSA), that is outside the addressing range used by the CPUs and their operating systems. The HSA may have a size of up to 16 GByte.

There is a 256-channel limit per operating system image, but z9 or z10 systems can have 1024 channels distributed in four Logical Channel SubSystems (LCSSs).

Since each Channel Subsystem can connect 64 K I/O devices (called Subchannels), this permits attachment of up to 256 000 I/O devices (e.g. disks) to a mainframe [24].

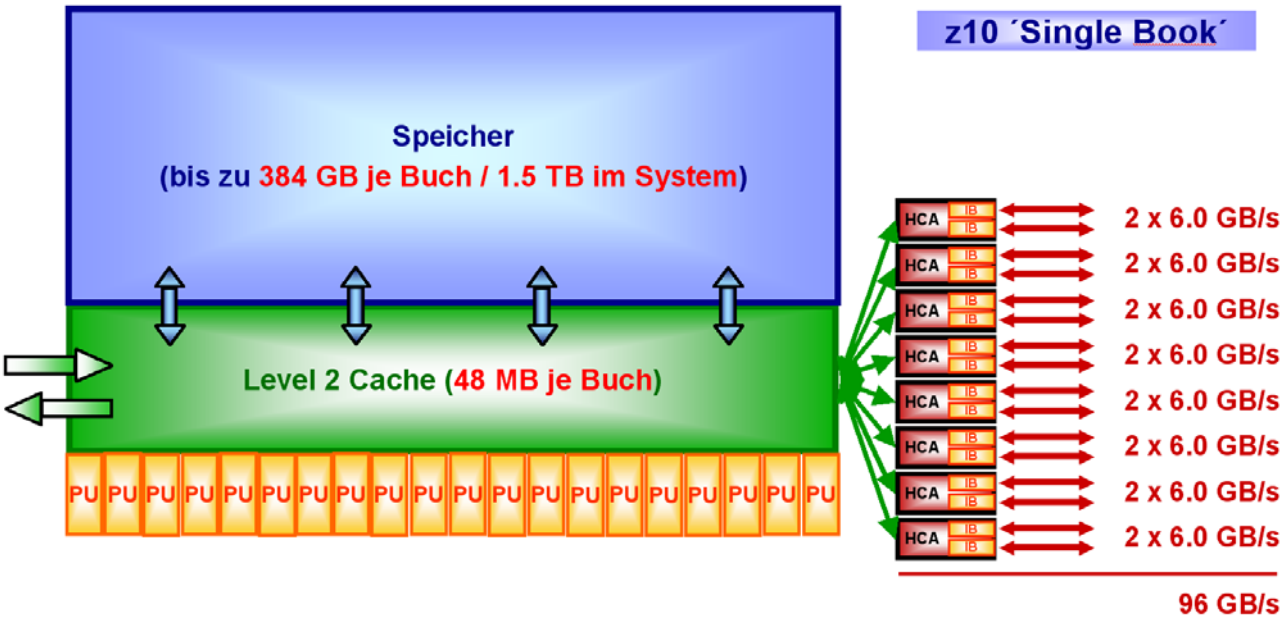
2.4.4 NUMA L2 cache



A maximum z10 system can have a total of 77 processing units (PU) in 4 books. Of these, 64 may be configured as CPUs; the remainder are SAPs and spares. PUs are packaged into "books", and a z9 or z10 mainframe may have up to 4 books. 3 of the 4 books may have 20 PUs each, the 4th book has 17 PUs.

The four L2 caches of the four books are interconnected by point-to-point links and form a single common and shared NUMA L2 cache that is used by the 77 PUs in all four books [25].

This is a unique System z feature. In other large systems, e.g from HP or Sun, the L2 cache is not shared, or is shared by a few CPUs at best.



2.4.5 DMA into L2 Cache

This is another unique z10 (and z9) feature. In all non-IBM systems, I/O adapter cards attach to main memory. In a z10 system, the Host Channel Adapter (HCA) attaches to the L2 cache, supporting a much higher bandwidth.

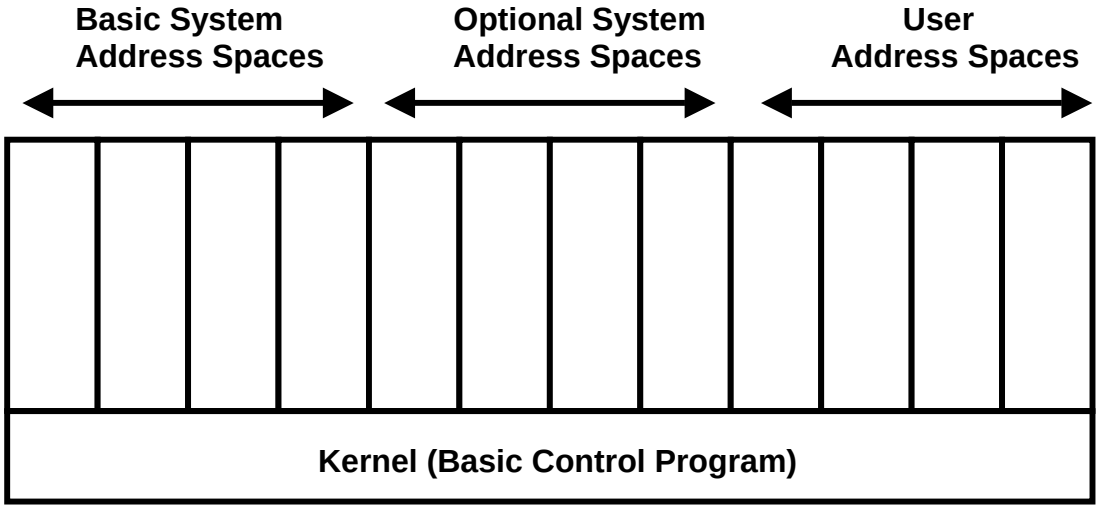
Each book has a maximum of 8 HCA adapter cards, and each HCA has 2 ports, each attaching a 6 GByte/s Infiniband link, for a total of 96 GByte/s I/O data rate per book. Four books permit a theoretical maximum data rate of 384 GByte/sec.

No other system but System z interfaces I/O connections to the L2 cache (instead of main storage). System z engineers were able to solve the resulting cache coherence problems. The cache coherence algorithms have not yet been published.

2.5 Supervisor

2.5.1 Supervisor Features

The z/OS Supervisor has a very clean structure. Its layout in main memory (called common) consists of a number of areas labeled Nucleus, Common Service Area, Link Pack Area, and System Queue area, which contain code and tables. They implement the basic supervisor (Kernel) functions like interrupt handler, scheduler/dispatcher, I/O supervisor, paging supervisor, file systems (z/OS calls them access methods), etc. These basic supervisor functions are separated from other operating system functions, which are implemented as subsystems, each running in its own virtual address space (system address spaces). Examples for subsystems are JES, TSO, Communication Server, Security Server, DFSMS, … There are well defined and stable interfaces between Kernel and subsystems, and inbetween subsystems. All interfaces are well documented.



Many of these interfaces and operating system components have remained essentially unchanged since the introduction of OS/360 in 1967. New functions are often implemented as additional subsystems.

The z/OS supervisor has unusual capabilities in terms of

- **reliability,**
- **availability,**
- **throughput**
- **maximum number of simultaneously executing processes**
- **maximum CPU utlization approaching 100 %.**
- **maximum number of CPUs in a symmetric multiprocessor configuration**

The very infrequent changes to basic operating system components result in something like 40 years of betatest. At the same time, 40 years of finetuning have improved throughput.

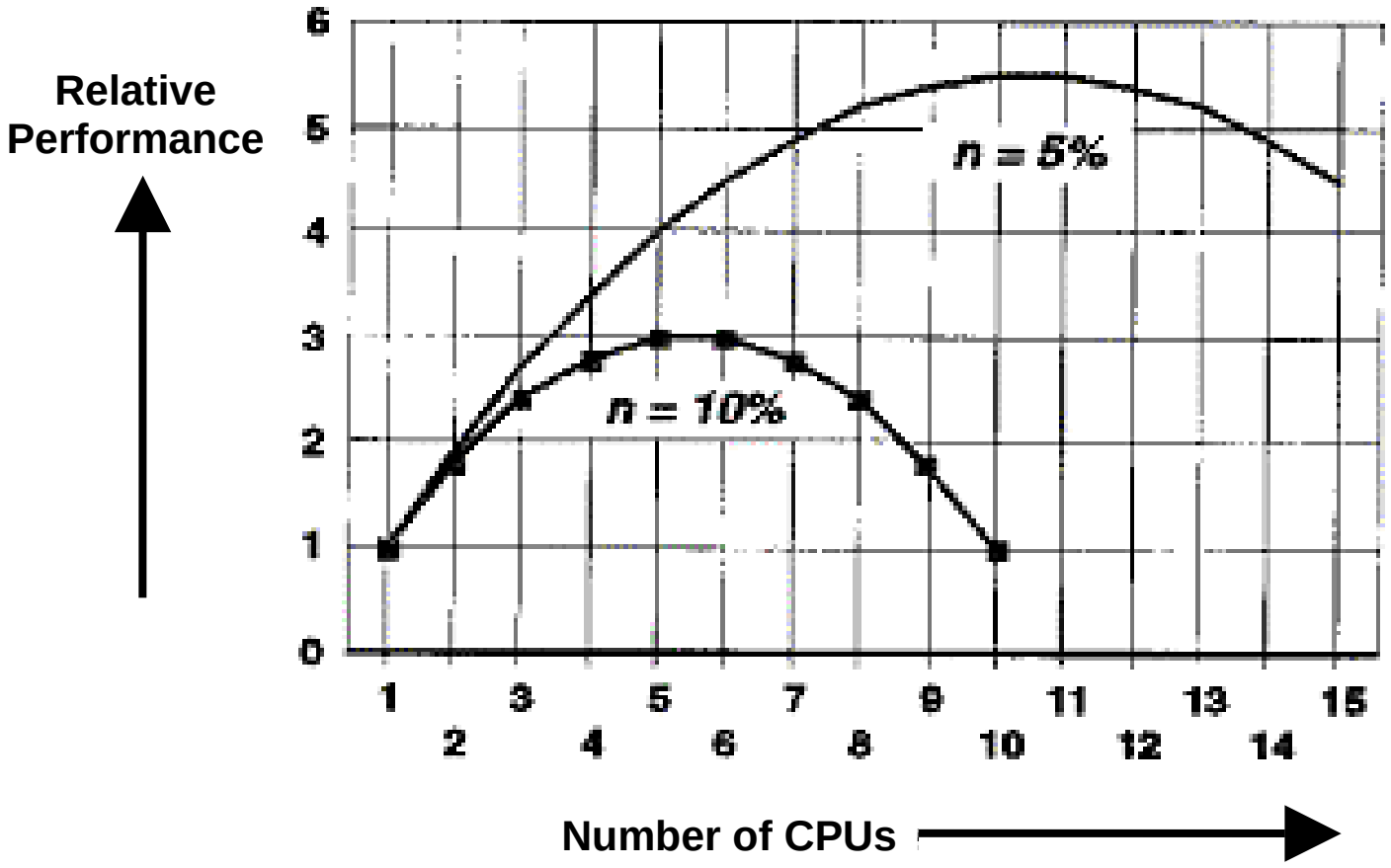
2.5.2 Supervisor Characteristics

z/OS supports a larger number of simultaneously executing processes than for example Linux. Among others this is due to facilities of the scheduler/dispatcher. As a consequence, when scheduling a new process, the scheduler/dispatcher pathlength is significantly longer then with Linux. For light workloads with uniform characteristics, the lean Linux scheduler/dispatcher is faster. For heavy and mixed workloads, typical for z/OS environments, the z/OS scheduler/dispatcher design is superior.

Mainframe installations frequently run at close to 100 % CPU utilisation. This is possible due to a number of features, one of them the z/OS Work Load Manager (WLM), see section 2.11. High CPU utilization means lower power consumption for each unit of work.

System z processors can be added dynamically without causing outages and processing disruptions.

2.5.3 Symmetric Multiptocessing



n = relative degredation of 2 versus 1 CPU

Symmetric Multiprocessors (SMP) can support only a limited number of CPUs. This number varies depending on the workload characteristics. Beyond a certain number, adding additionals processors will actually reduce performance. There are multiple reasons; the most important one are conflicts of multiple processes competing for the

same supervisor resourse. Unfortunately, transaction processing and data base operations have expecially poor SMP scaling characteristics.

For transaction processing and data base operations, z/OS supports in an SMP configuration a larger number of CPUs than any other operating system. Some experts believe the maximum number of usable CPUs is superior up to a factor 2 .

IBM announced the first dual processor mainframe in 1967 (S/360 model 67). Since then, 40 years of finetuning the z/OS kernel have continuously reduced the number of kernel access conflicts by independetly running processes. It appears, the multiprocessing finetuning of an operating system kernel requires many years of experience and cannot be accelerated. The superiority of z/OS is the result of having been doing this for more years than on any other system. In z/OS, there is still progress each year; the headstart of z/OS can be expected to continue for some time.

2.6 Job Entry – Automated Job Scheduling

Nothing comparable to the JES2 and JES3 systems exists on any other platform [26].

Run queues evolved from a literal queue of people at the door of a computing room waiting to enter their job into the computer, to a heap of media on a "jobs-waiting" table, or batches of punch-cards stacked one on top of the other in the card reader, until the machine itself was able to select and sequence which magnetic tape drives were online. At Cambridge University in England the job queue was at one time a washing line from which tapes were hung with different colored clothes-pegs to indicate job-priority. Automated job entry was a part of the first version of OS/360.

An overview of Microsoft Job Scheduling alternatives is available in [27]. The article states: "Microsoft Windows Server and the underlying multicore, multiprocessor architectures provide a platform that is capable of handling workloads equivalent to a 4000-MIPS mainframe computer", implying it does not scale to larger systems.

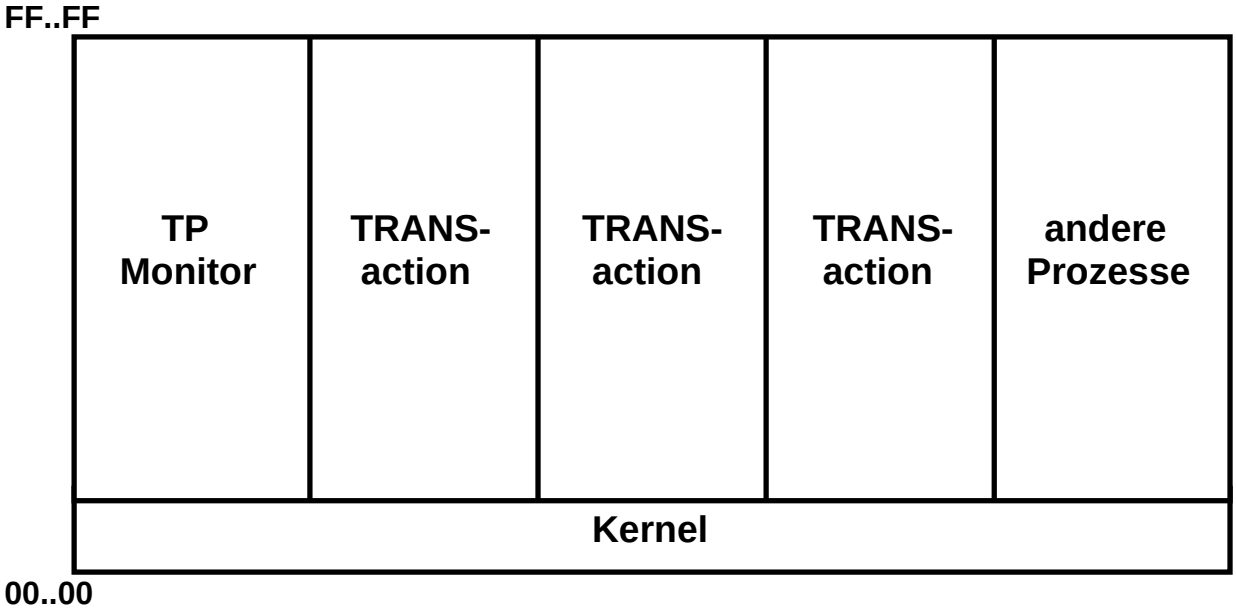
The cron program is the UNIX's answer to automated job scheduling. It is used to schedule jobs to run at a particular time or at a particular frequency. It is actually a background system process; -`crond` (the cron daemon) is started at boot time from `rc` scripts, that define the basic functions of each service, as well as each service's dependencies for start-up. Crontab files define jobs according to user name. Each user on the system can have a corresponding crontab file specifying their own automated job schedule – including root.

A large number of independent software vendors provide job scheduling solutions for Unix, Linux and Windows Server systems with functions that go beyond cron. Examples are:

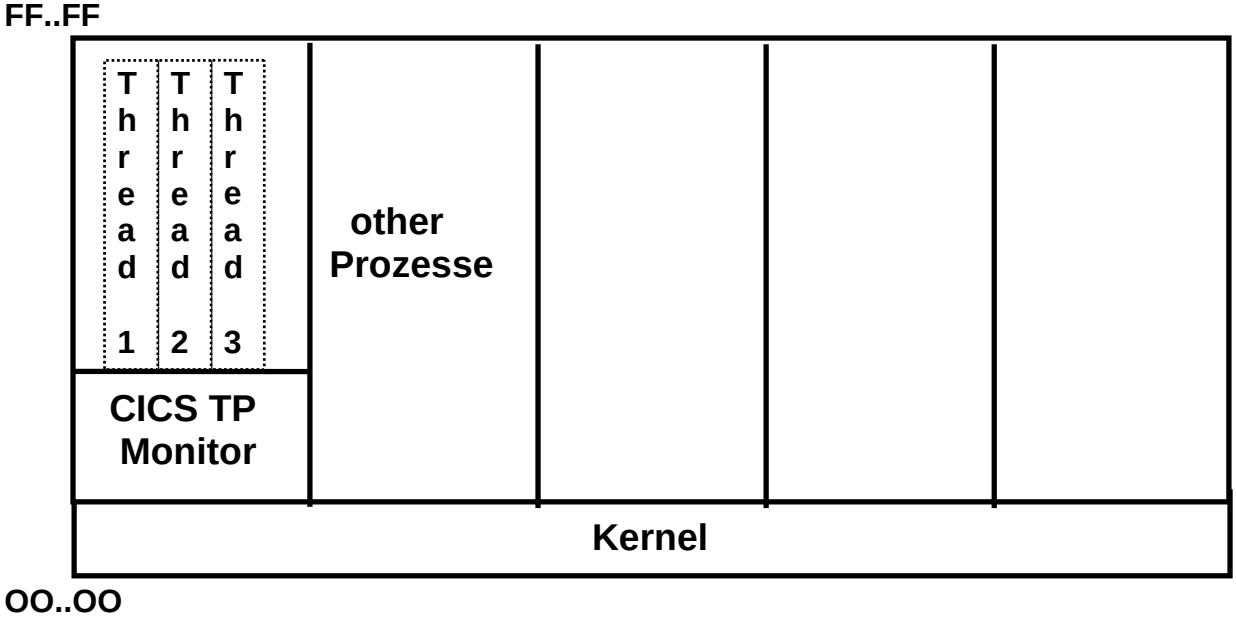
- **ActiveBatch der Firma Advanced –System Concepts, http://www.advsyscon.com/ ,**
- **BMC Control-M, http://www.bmc.com/products/offering/control-m.html,**
- **CA Unicenter AutoSys, http://www.ca.com/us/workload-management.aspx,**
- **IBM Tivoli Workload Scheduler (TWS),**
- **Dollar Universe der Firma Orsyp, http://www.orsyp.com/en.html ,**
- **OpenPBS (Portable Batch System), http://www.openpbs.org,·open source package originally developed for NASA in the early 1990s,**
- **PBS Pro der Firma PBS Grid Works, http://www.pbsgridworks.com, enhanced com mercial version von OpenPBS),**
- **Redwood Software, http://www.redwood.com/,**
- **SMA (Software and Management Associates) OpCon/xps, http://www.smausa.com/product.htm,**
- **Tidal Software Enterprise Scheduler, http://www.tidalsoftware.com/products/enterpriseJobScheduling.aspx,**
- **UC4 Software, http://www.uc4.com/,**
- **Global Event Control Server der Firma Vinzant Software, http://www.vinzantsoftware.com/.**

None of them comes close to the functionality of the z/OS Job Entry Subsystem (JES). For JES details, see [28], [29]

2.7 CICS



Process-Approach



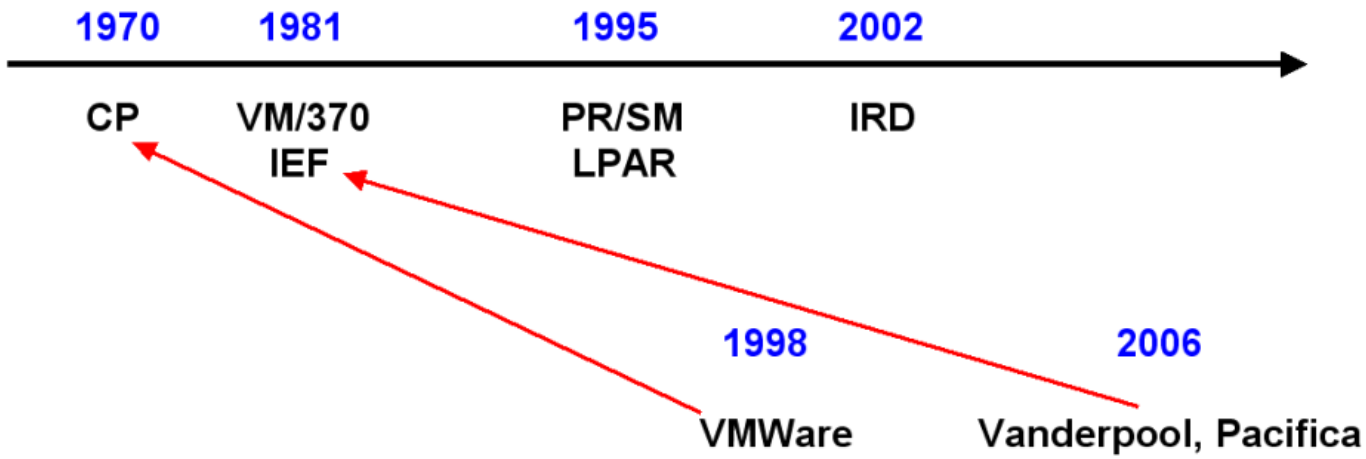
Thread-Approach

All existing transaction monitors, including z/OS IMS/DC and the multiplatform CICS version running on Unix, Linux and Windows systems, use separate address spaces for the execution of individual transactions. The same applies to all Java Transaction Service JVMs outside z/OS. CICS under z/OS runs multiple transactions and multiple JVMs within a single address space [30].

The z/OS version of the CICS Transaction monitor has superior performance characteristics. System z CICS runs thousands of transactions in a single address space, and z/OS solves the isolation problem using the Enclave mechanism. The enclave mechanism performs isolation for hundreds or thousands of parallel executing CICS transactions within a single CICS address space [31].

CICS can execute more than 10 000 transactions each second.

2.8 Virtualization



2.8.1 History of virtualization

Virtualization started in the 1960s at ICL (International Computers Limited) Corporation (now Fujitsu) in England and their VME (Virtual Machine Environment) operating system [32], and simultaneously at the IBM Scientific center in Cambridge, Mass., with the implementation of the CP-67/CMS operating system for the IBM System/360-67 [33].

Originally, the CP hypervisor (host operating system, later renamed VM/370) supported the CMS guest operating system. CMS was single user, and non-virtual. It was originally able to run on naked S/360 hardware.

Running a virtual operating system like OS/370 as a guest under CP caused a problem, because the guest, running in a host virtual address space, wants to do its own address translation. Shadow page tables were introduced to solve this problem.

2.8.2 IEF, Vanderpool, Pacifica

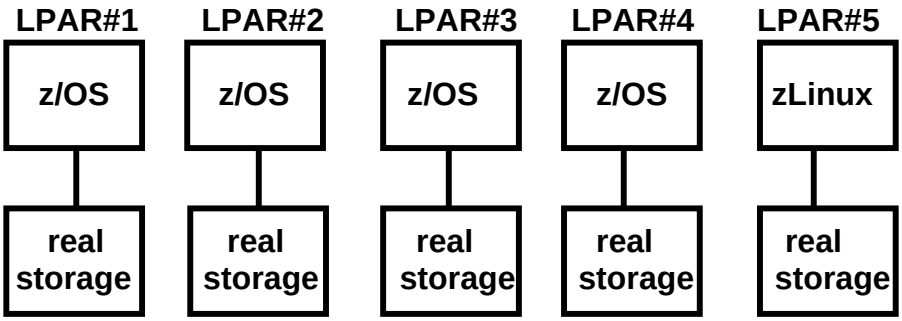
Each process switch requires reloading the shadow page tables, which reduces performance. As a solution, VM/370 introduced in 1981 the Interpretive Execution Facility (IEF) [34]. IEF allows a CPU to run either in Host or in Guest Mode, and when running in Guest Mode, to perform virtual address translation in the regular way, using segment and page table just like in a non-virtualization environent.

In the 90s, a small group of VM/370 developers left IBM to start VMWare. Their aim was to introduce virtualization for the IA32 (Pentium) platform. Unfortunately, the IA32 architecture lacks facilities which were avalable on S/370 [35]. It was a major achievenment to get virtualization running on IA32 despite the shortcomings of the architecture. A major performance impact and some compatibility problems remained.

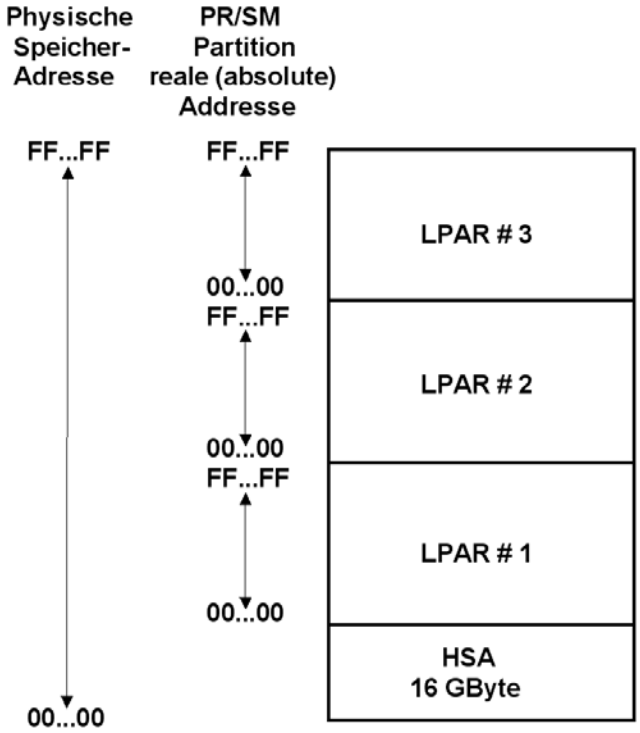
To overcome these problems, and to eliminate shadow page tables, Intel and AMD introduced the Vanderpool and Pacifica technologies in 2006. This is essentially the same approach used in the VM/370 Interpretive Execution Facility in 1981.

2.8.3 Logical Partitions

Virtualization permits a maximum of flexibility, but suffers some performance degredation. In 1994 Amdahl introduced the Multiple Domain Feature (MDF) [36] , and IBM a little later the Logical Partition (LPAR) [37]. Different from virtual machines, LPARs run in real and not in virtual storage.



This looks like shown above. LPARs are the equivalent of a separate mainframe for most practical purposes. Each LPAR contains its own operating system and maps its virtual storage into its own real storage. A Hypervisor named PR/SM controls the LPARs.



For practical reasons, the real storage of all LPARs is mapped into a single physical storage. Each LPAR is assigned a part of physical storage, using address relocation, with a starting address of hex 00..00 . The size of real storage assigned to each LPAR is defined by a system initialization parameter, and may differ between LPARS.

A separate part of physical memory (Hardware System Area, HSA) is used für system internal purposes, e.g. the hypervisor.

These are some of the characteristics:

- **Each LPAR is independent.**
- **Each LPAR runs its own operating system.**
- **Each partition owns a defined amount of real storage.**
- **Strictly no storage shared across partitions.**
- **Separation of logical partitions is considered as perfect as having each logical partition on a separate server.**
- **Zone relocation lets each partition start real storage at address 00..00.**
- **No virtual storage / paging is done by the LPAR hypervisor.**
- **The LPAR hypervisor (control program) is considered part of firmware.**
- **CPUs may be dedicated to a logical partition or may be shared by multiple partitions.**
- **When shared, each LPAR is assigned a number of logical processors (up to the maximum number of physical processors) and a weighting.**
- **I/O channels may be dedicated to a partition or may be shared by multiple partitions (Multiple image facility, MIF).**
- **I/O Devices can be dedicated or shared across several LPARs.**
- **Each LPAR has its own architecture mode (S/390 or z/Architecture).**

As an LPAR extension, IBM introduced in 2003 the IRD (Intelligent Resource Director) technology. Intelligent Resource Director uses facilities of the z/OS Workload Manager (WLM), Parallel Sysplex, and PR/SM. It includes:

- **Dynamic administration of real storage**
- **LPAR CPU management**
- **Dynamic Channel Path Management**
- **Channel Subsystem Priority Queuing [38]**

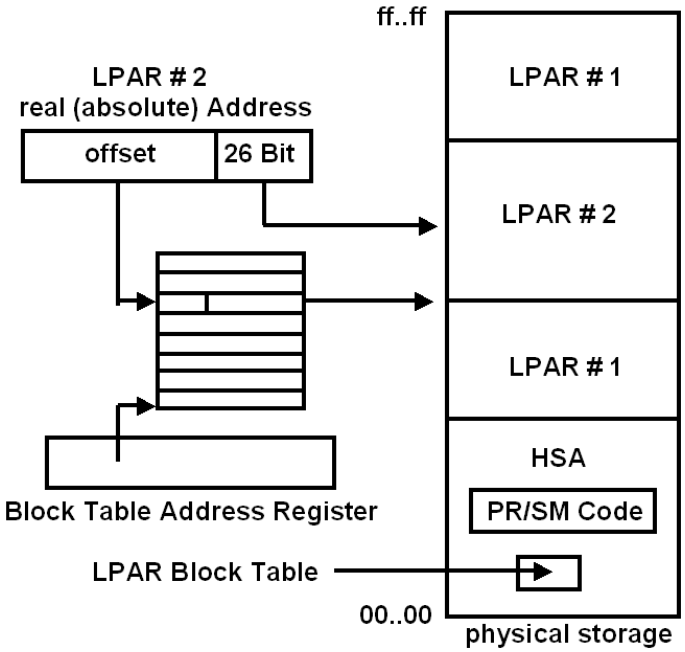
The system z LPAR implementation is unique in comparison to the other partitioning implementations available from IBM and other hardware vendors. It is the most mature and dynamic partitioning technology in the industry [39]. A subset of the functions has been implemented in IBM's System p.

Until now (2010), the x86, Itanium, and Sparc architectures feature no equivalent for LPAR and IRD. It is reasonable to assume they will close this gap at some time in the future.

2.8.4 Dynamic administration of real storage

The basic LPAR approach assumes that available physical storage is partitioned into multiple real storage areas at system initialization time. One real storage area is assigned to each LPAR and its size is fixed as a system initialization parameter. When it is desired to change the partitioning parameters later on (e.g. real storage size), a shutdown of the running systems is required.

The dynamic administration of real storage permits on a running system the dynamic change of real storage size assigned to individual LPARs in 64 MByte blocks. Unfortunately each LPAR requires a contiguous set of real addresses starting at address 00..00.



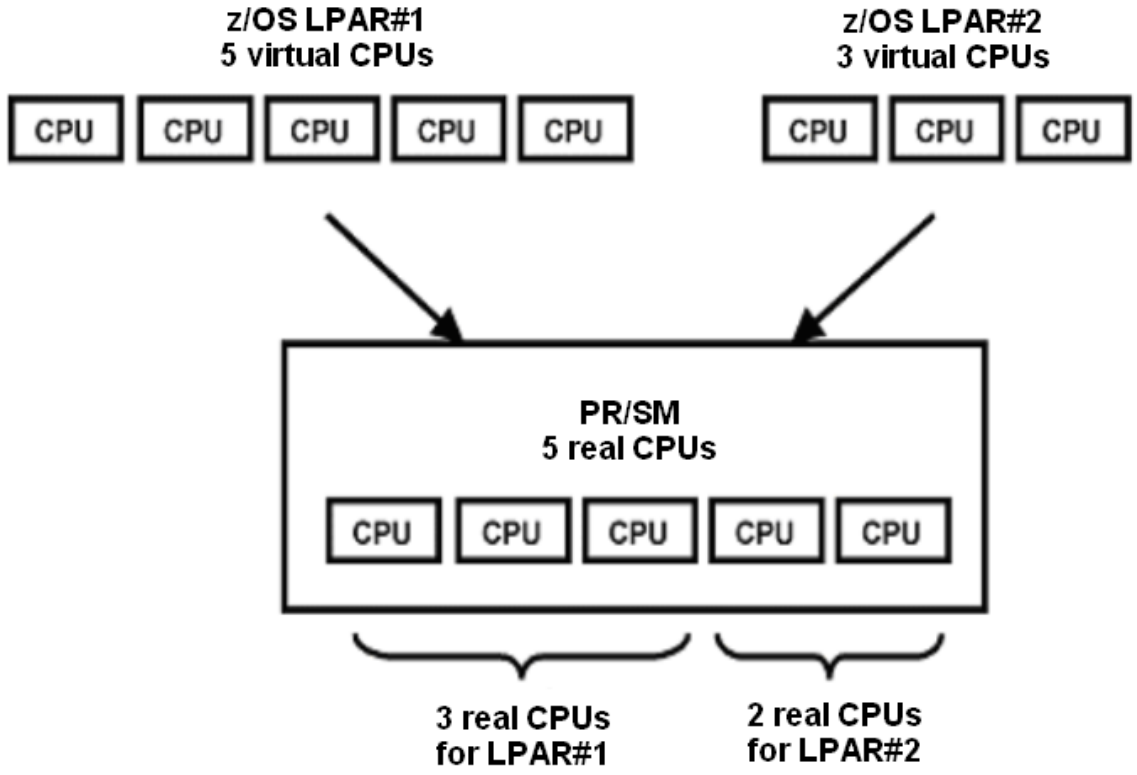
To assign real storage in sizes of 64 MByte blocks, and maintain a contiguous range of addresses, an additional level of address translation is used as shown above (no page fault mechanism is required). This is independent of (and invisible to) the virtual address translation performed by the operating system in each LPAR. It permits system administration functions (especially the Work Load Manager) to dynamically adjust the amount of real storage available to each LPAR without interrupting the running systems and applications.

The addresses of the relocated 64 Mbyte blocks are contained in an LPAR Block Table, which is part of the Hardware System Area (HSA) mentioned above. Each LPAR has its own Block Table Address Register. The lower 26 bit of an LPAR real address point to a data item within a 64 MByte block. The higher bits are translated in an entry of the LPAR Block Table to point at a particular 64 MByte block in physical storage.

2.8.5 LPAR CPU management

In a system with a single CPU, the CPU resources are used (timeslized) by multiple LPARs. In a system with multiple CPUs, these may be either dedicated to an LPAR, or may be shared by multiple LPARs. A mix is possible: Some LPARs use dedicated CPUs; the remaining CPUs are shared by the other LPARS.

Sharing CPUs allows optimal resource utilization. Assume for example a system with 5 real CPUs and 2 LPARs.



In this example you may assign a weight of 3 to LPAR#1 and a weight of 2 to LPAR#2. This means, LPAR#1 is assured a minimum capacity of 3 CPUs, and LPAR#2 a minimum capacity of 2 CPUs.

If one of the LPARS does not need the assured capacity, IRD may assign the unused CPU capacity to the other LPAR. However. the operating system in this LPAR has to be configured to be able to use the additional CPU capacity.

Thus, in this example, the z/OS in LPAR#1 is configured for using 5 CPUs and the z/OS in LPAR#2 is configured for using 3 CPUs. These are Logical CPUs; PR/SM and IRD assign (executable) logical CPUs to physical CPUs if and when they are available. Mapping of logical CPUs to physical CPUs can occur with arbitrarily fine granularity. The Work Load Manager (WLM) is used to optimze overall system performance.

This permits to run CPUs at near 100 % capacity. For a given workload this minimizes the number of CPUs. Mainframes feature superior energy efficiency (green computing), because there are fewer idle hardware resources.

2.8.6 Dynamic Channel Path Management

I/O devices like disks are attached via channels to a mainframe system. Without IRD, channels are statically assigned to individual LPARs. Multiple channels are needed to support a maximum of I/O traffic. System definition parameters specify, which channels belong to a particular LPAR. When a system is initialized, the channels are assigned to their specific LPARs.

An LPAR may have to handle different types of workloads during the run of a day. Typically, there are different requirements during daytime and the night shift.

Dynamic Channel Path Management is designed to dynamically adjust the channel configuration in response to shifting workload patterns. It can provide improved performance by dynamically moving the available channel bandwidth to where it is most needed.

2.8.7 Channel Subsystem Priority Queuing

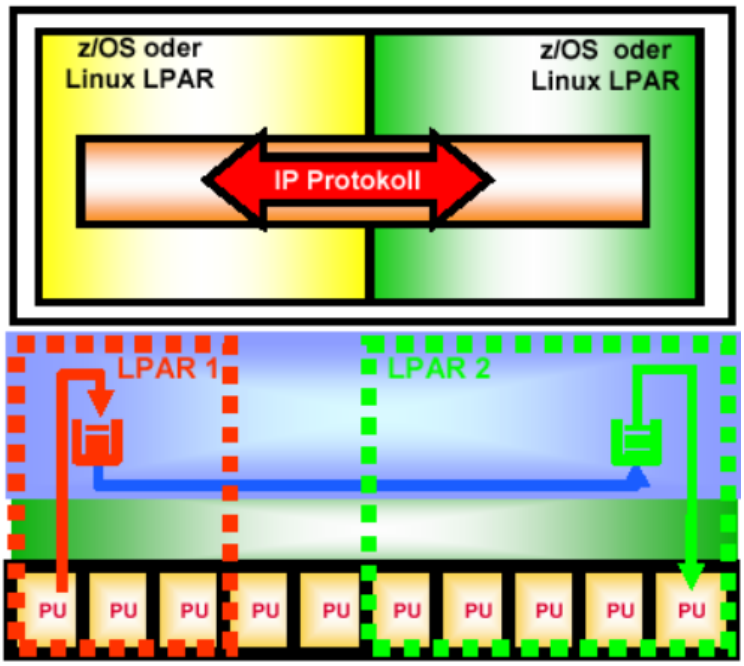
Without IRD, I/O requests are handled by the channel subsystem on a first-in, first-out basis. This can, at times, cause high priority work to be delayed behind low priority work.

With Channel Subsystem Priority Queuing, if important work is missing its goals due to I/O contention on channels shared with other work, it will be given a higher channel subsystem I/O priority than the less important work.

This function goes hand in hand with the Dynamic Channel Path Management, as additional channel paths are moved to control units to help an important workload meet its goals.

Assume workloads that happens to be using the same channel. Channel Subsystem Priority Queuing ensures that an important workload receives additional channel bandwidth in front of a less important workload [40].

2.8.8 Hipersockets



Internal Queued Direct I/O and Hipersockets permit high speed communication between LPARs. It is possible to configure up to 16 internal (virtual) Ethernets within a single mainframe system [41]. This permits LPARs to communicate with each other over what looks like a socket/TCP/IP/Ethernet connection with each other, which in reality is implemented with code and queues in the Hardware System Area.

A typical application may have multiple zLinux instances communicating with each other and with one or several z/OS DB2, or CICS LPARs [42].

2.9 DB2 for z/OS

Most data base systems are designed to be portable from one operating system to another.

For DB2 there exist two implementations: The DB/2 for z/OS implementation is independent from the DB2 multiplatform implementation for Unix, Linux, and Windows. The DB2 multiplatform implementation (like any multiplatform implementation) cannot utilize platform specific functions.

While the compatibility with the z/OS DB2 implementation is the same as for example between a Linux DB2 and a Windows DB2 installation, the z/OS implementation of DB2 can utilise z/OS functions not available on other operating systems. Examples are: WLM, Sysplex, GDPS, Coupling Facility, hardware encryption, XML parsing, I/O subsystem, System Managed Storage, and security facilities.

DB/2 for z/OS features high availability and scalability for very large data bases [43], [44].

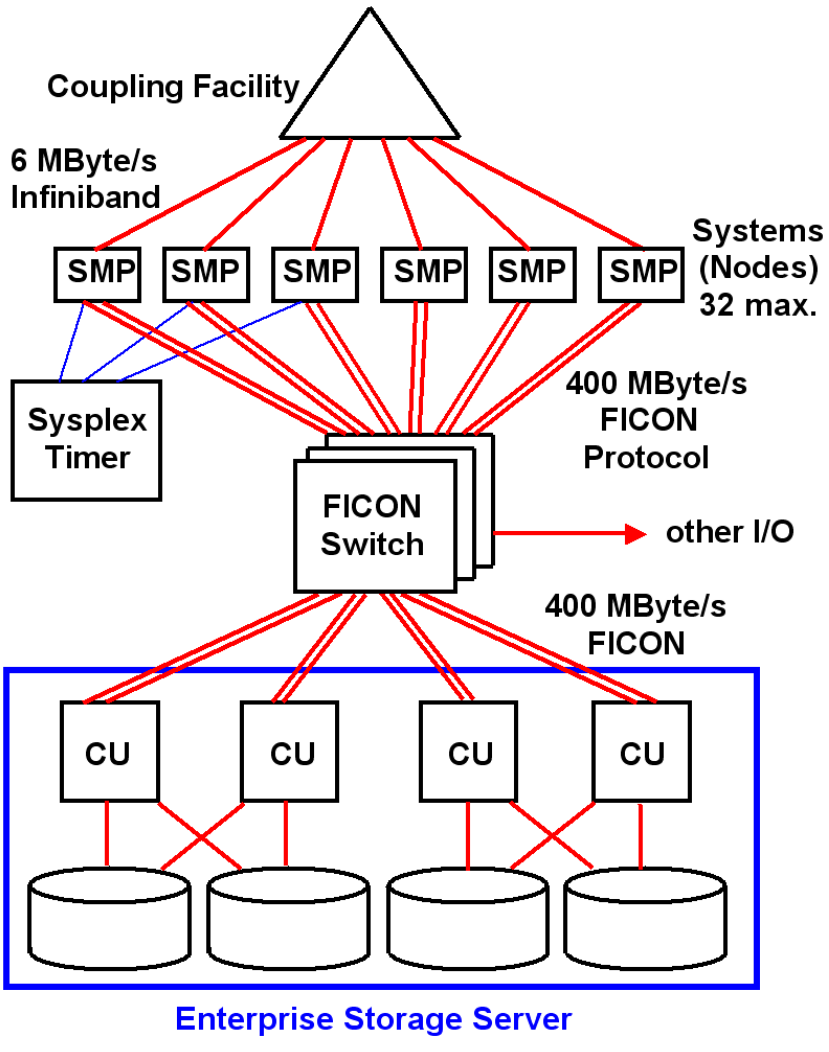
2.10 Coupling Facility

2.10.1 Parallel Sysplex

A Parallel Sysplex, commonly called a Sysplex, is a cluster of multiple mainframe systems. A Sysplex looks like a single logical system running on multiple physical systems. In addition, multiple LPARs on a single system have all the characteristics of independent systems, and can thus also be configured as a Sysplex.

Components of a Sysplex include:

- **Multiple Mainframes, called Systems or Nodes (maybe implementd as LPARs);**
- **Connections to multiple Enterprise Storage Servers;**
- **A Sysplex Timer which synchronizes all member system clocks;**
- **Coupling Facility (CF) hardware, allowing multiple processors to share, cache, update, and balance data access;**
- **Cross System Coupling Facility (XCF), which allows systems to communicate peer to peer;**
- **Couple Data Sets (CDS);**
- **Global Resource Serialization (GRS), which allows multiple systems to access the same resources concurrently, serializing where necessary, to ensure exclusive access.**



Multiple Symmetric Multiprocessors (SMPs) use the FICON Version of the Fibre Channel Protocol to connect to Control Units (CU), implemented today by an Enterprise Storage Server.

All I/O devices can be reached by multiple channel paths, and a connection may dynamically change during the execution of an I/O operation.

A central component is the Coupling facility (CF). CFs are unique to System z mainframes. Equivalent functions are not available on other platforms.

2.10.2 Coupling Facility Characteristics

A coupling facility is a mainframe processor, with memory, special connections (CF links), and a specialised operating system called Coupling Facility Control Code (CFCC). Originally, the Coupling Facility was always a dedicated external mainframe system, specially configured, and using only coupling facility processors. In this case, the CF is connected by dedicated glas fibre links (CF links) point to point to all systems of a sysplex. The CF link connection has DMA capabilities supported by spezialized hardware at each end. Cross-system extended services (XES) is a z/OS component that enables communications over coupling links with the coupling facility.

As mentioned above, multiple LPARs on a single system have all the characteristics of independent systems, and thus can be configured as a sysplex. Therefore, an LPAR can also be configured as an "Internal Coupling Facility" (ICF). A specialized protocol connects an ICF to LPARs on the same system, emulating CF links.

A CF has no I/O devices, other than the CF links. The information in the CF resides entirely in memory, and CFCC is not a virtual memory operating system. A CF typically has a large memory - of the order of multiple gigabytes. In principle, any IBM mainframe can serve as a coupling facility. The CF runs no application software.

Supported by CFs, a Sysplex cluster scales very well up to several hundreds of CPUs (up to 32 systems, each with up to 64 CPUs) running transaction and data base applications. Using the CF links, data can be directly exchanged between the CF memory and the memory of the attached systems, using a direct memory access like mechanism, without interrupting a running program. Systems in a Sysplex cluster store CF information in local memory in areas called bit vectors. This enables them to locally query critical state information of other systems in the Sysplex without the need for issuing requests to the CF. The System z Architecture includes 18 special machine instructions and additional hardware features supporting CF operation.

A CF is used to store data arrays (called structures) for three purposes:

- **Locking information that is shared among all attached systems,**
- **Cache information (such as for a data base) that is shared among all attached systems, maintaining coherency between local buffer pools in each system,**
- **Data list information that is shared among all attached systems.**

These three purposes are catered for by three types of structures:

- **Lock structure**
- **Cache structure**
- **List structure**

A structure is a dedicated portion of CF memory. It is said to be connected to specific CF-exploiting applications on the coupled z/OS systems. A typical Parallel Sysplex contains several structures of each type; each software exploiter may use several structures of each type. For example each DB2 Data Sharing Group uses one Lock structure, one List structure and several cache structures (one for each Group Buffer Pool) [45].

2.10.3 Coupling Facility Requests

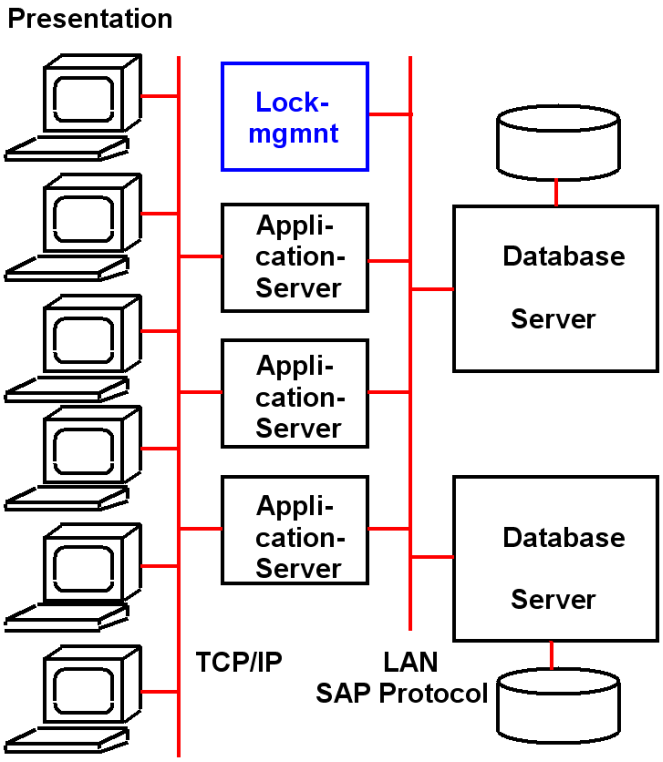
A request by a Sysplex system to a CF structure is of one of two kinds:

- **Synchronous (Sync) requests. When a z/OS system issues a request it waits for the request to complete, actively "spinning" on one of its own processors. Sync requests are quick but the response time depends on the CF link transmission time and the CF response time. So Sync requests are relatively expensive in CPU terms.**
- **Asynchronous (Async) requests. When a z/OS system issues a request it does not wait for the request to complete. Async requests are slower than Sync requests (as they have a lower priority in the CF).**

Exploiting z/OS applications explicitly issue CF requests as Synch or Asynch. The "Dynamic Request Conversion" heuristic algorithm uses sampled response times to decide whether to convert Sync requests to Async or not. These decisions are based on such criteria as coupled processor speed and geographical distance. The greater the distance between the coupled z/OS system and the CF, the greater the likelihood requests will be converted to Async from Sync.

2.10.4 Lock Structure

Modern multi-system transaction processing systems use a central Lock management server to synchronise parallel transaction processing.



Using a typical SAP R/3 ERP configuration as an example, a group of application servers is supported by an additional Lock management server (Sperr-Server or Enqueue Server in SAP terminology). Whenever an application server wants to acquire or change a lock, it interrogates the lock server as to the lock status.

Application servers, lock server, and data base server(s) are interconnected through a high speed network, for example a gigabit or 10 gigabit ethernet. To interrogate the lock server, the requesting application server has so send a message traversing the complete TCP/IP stack.

In a Sysplex, the coupling facility lock structure is interrogated through the low latency coupling facility protocol. In the majority of cases, even this is not necessary, since the needed locking information is available in the local bit vector.

The net effect is much superior scaling. To quote Jim Gray and Adreas Reuter: “You cannot scale a transaction processing system, if you do not solve the locking problem” [46].

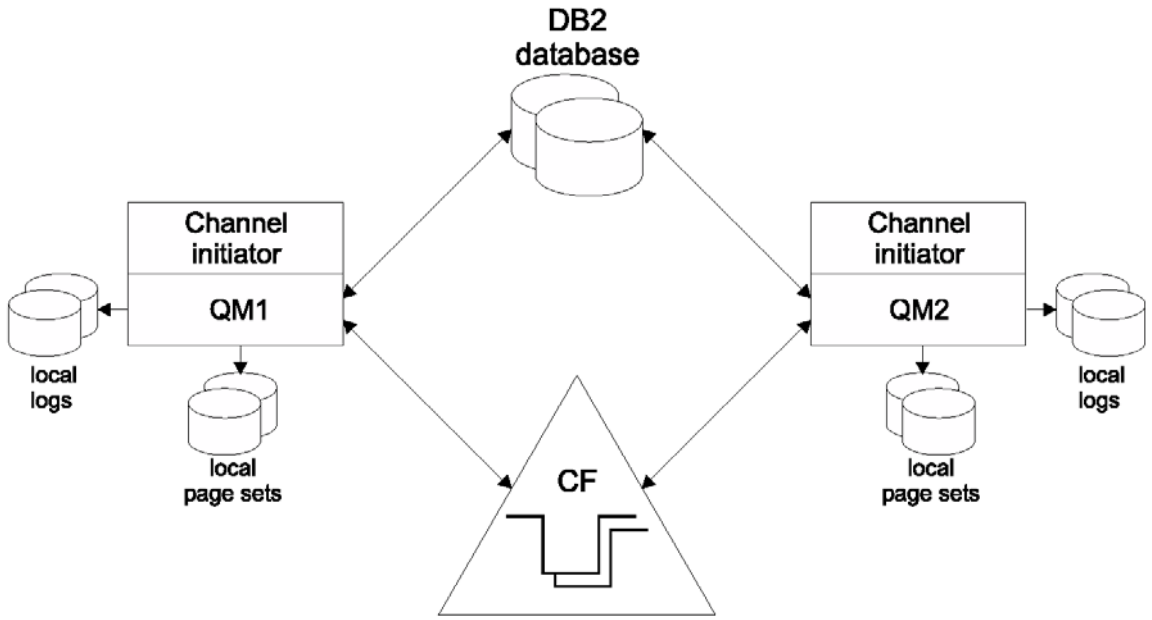
2.10.5 Cache Structure

Using Coupling Facility Cache Structures it is possible to avoid Buffer Pool invalidation when committing a transaction [47]. Frequently a following transaction may want to use data already available in the buffer pool. Cache structures maintain cache directories (directories of all buffer pool entries) to gurantee data integrity and thus avoid unnecessary I/O operations. The bit vector facility is used for this purpose.

Some Data Bases, e.g. DB2 and Adabas, additionally use cache structures to cache data themselves.

2.10.6 List Structure

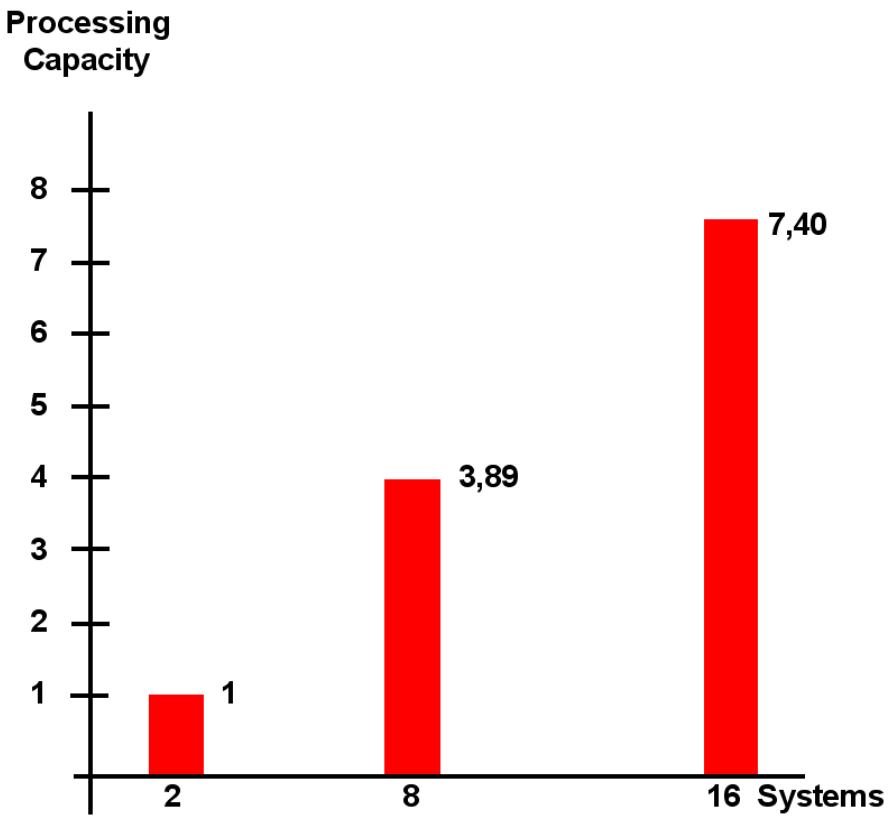
The WLM (see section 2.11) in a Sysplex configuration provides an effective and proven Cloud Computing infrastructure (Cloud Hosting). One example of using the Coupling Facility List structure is MQSeries Cloud processing [48].



In a Message Based Queuing Application an MQSeries message is sent to a sysplex. Each system in the sysplex contains a Queue Manager (QM) and associated Channel Initiator, able to receive an MQmessage. The group of queue managers that can access the same shared queues is called a queue-sharing group. The Queue Managers are running on separate systems. Each member of the queue-sharing group has access to the same set of shared queues. A coupling facility list structure manages the queues.

The figure above illustrates a queue-sharing group that contains two queue managers.

2.10.7 Performance



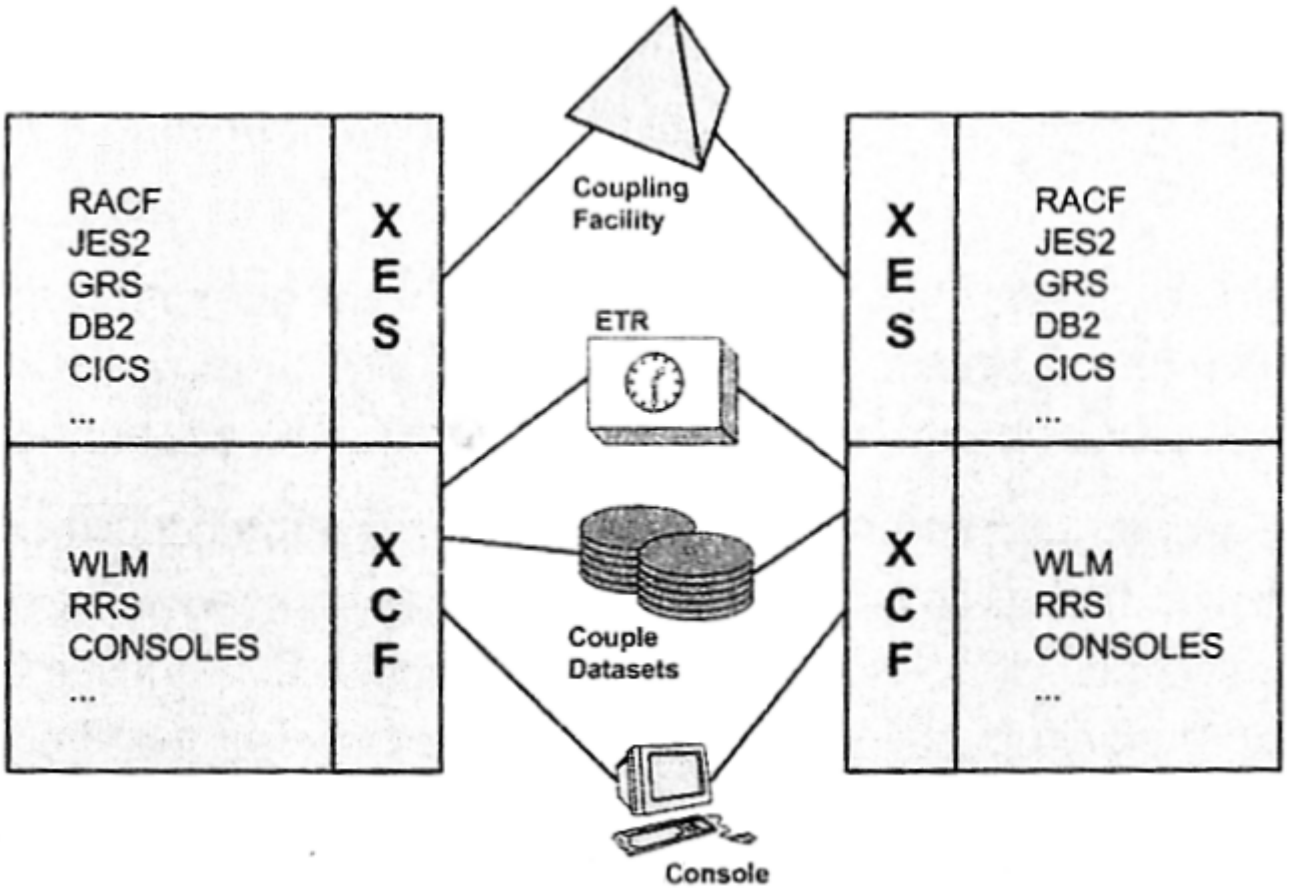
The above graph shows Parallel Sysplex performance degredation. It has been copied from [49]. The benchmark used includes a real life mixture of on-line transaction processing, reservation systems, data warehousing, and banking applications.

A sysplex with 2 systems and 16 CPUs on each system was used as a base. Increasing the number of systems to 8 (128 CPUs) resulted in a 3.89 times better processing capacity (theoretical maximum = 4).

Using 16 systems and 256 CPUs resulted in a 7.40 times better processing capacity (theoretical maximum = 8).

A comparable scaling for a mix of transaction processing applications should be very difficult or impossibe to achieve on any other platform [50].

2.10.8 Geographically Dispersed Parallel Sysplex



A Geographically Dispersed Parallel Sysplex (GDPS) is an extension of a Parallel Sysplex of mainframes, supporting multiple site configurations, located potentially in different cities. GDPS helps automate recovery procedures for planned and unplanned outages to provide near-continuous availability and disaster recovery capability. Assume for example two sites, one for production, and the other for disaster recovery.

There are two flavors of GDPS:

- **Synchronous GDPS/PPRC (peer-to-peer remote copy) that uses Metro Mirror technology;**
- **Asynchronous GDPS/XRC that uses z/OS Global Mirror technology.**

Common to both synchronous and asynchronous GDPS solutions is the ability to provide consistent data in the recovery site for z/OS and non-z/OS data. Without this ability to maintain dependent write sequences, an installation cannot, for example, guarantee data integrity where log records need to match the database content. To avoid data integrity problems, you might need to restore from the last image copy and apply the log records, a process that can take several hours or even days. Instead, with GDPS, all you might need to do is restart the database managers, which takes a matter of minutes.

Metro Mirror functions offer a synchronous long-distance copy option that constantly updates a secondary copy of a volume to match changes made to a source volume.

With Metro Mirror copying, the source and target volumes can be at different sites some distance apart. Synchronous mirroring means that each update to the source storage unit must also be updated in the target storage unit before another update can process. When Metro Mirror receives a host update to the source volume, it completes the corresponding update to the target volume. This guarantees data consistency by ensuring that a write operation that completes is received by the host application after the update has been committed to the target storage unit and acknowledged by both the source and target storage units. This results in near perfect data consistency but can result in lag time between transactions.

With Metro Mirror, consistency is guaranteed across all volumes on which an application does write operations. Metro Mirror copying supports a maximum distance of 300 km (186 miles). Delays in response times for Metro Mirror are proportional to the distance between the volumes. However, 100% of the source data is available at the recovery site when the copy operation ends.

Global Mirror (XRC) for z/Series is an asynchronous disk mirroring methodology. Most other large volume disk mirroring approaches are controller or storage subsystem based. XRC utilizes application code running on a z/Series processor to perform the data replication and maintain data consistency.

The host-based application that supports and controls the replication process is called the System Data Mover (SDM). Since the SDM is a z/OS application, it provides a greater level of control and reporting capabilities with fewer constraints than a controller based solution.

The z/Series implementation consists of two complementary asynchronous processes:

- **The first process issues special Read commands across the network to each of the primary disk subsystems. Any changed data is read across the network and written to journal files at the secondary site.**
- **Meanwhile, another task under the control of the SDM is constantly checking the status of the journaled data. Once the SDM determines that there is a time consistent copy of data from all of the primary disk subsystems, this data becomes a consistency group and is written to the secondary volumes.**

In this fashion, the SDM ensures that no dependent writes are made out of sequence and data residing on the secondary volumes will always provide a time consistent copy of the primary volumes being mirrored. Because it is time consistent, the data on the secondary volumes can be used for recovery.

The Global Mirror for z/Series is the most mature of any of the alternatives that provide true data consistency across any distance. The software has been available for over 10 years and offers growth potential and scalability that exceeds the other disk mirroring methodologies.

Since running as z/OS application code, many of the constraints inherent in other types of implementations can be mitigated simply by adding additional SDMs. The current version supports 182 clustered SDMs with each SDM mirroring up to 2000 volumes. This provides the ability to provide data consistency to over 350,000 volumes.

2.11 Work Load Management.

On a large Unix system, applications with differen characteristics are often assigned to independet Unix instances, freqently independent computers. This makes installation planning very easy: If you want to add a new application, you buy a new server – computers like Blade Servers are inexpensive. However, this results in poor CPU utilization, high energy consumption, and - most important – high administration cost.

Workload management funktions are part of many modern operating systems. and used in all large commercial UNIX- systems. Examples are the HP-UX Workload Manager [51] oder der Solaris 10 Resource Manager [52].

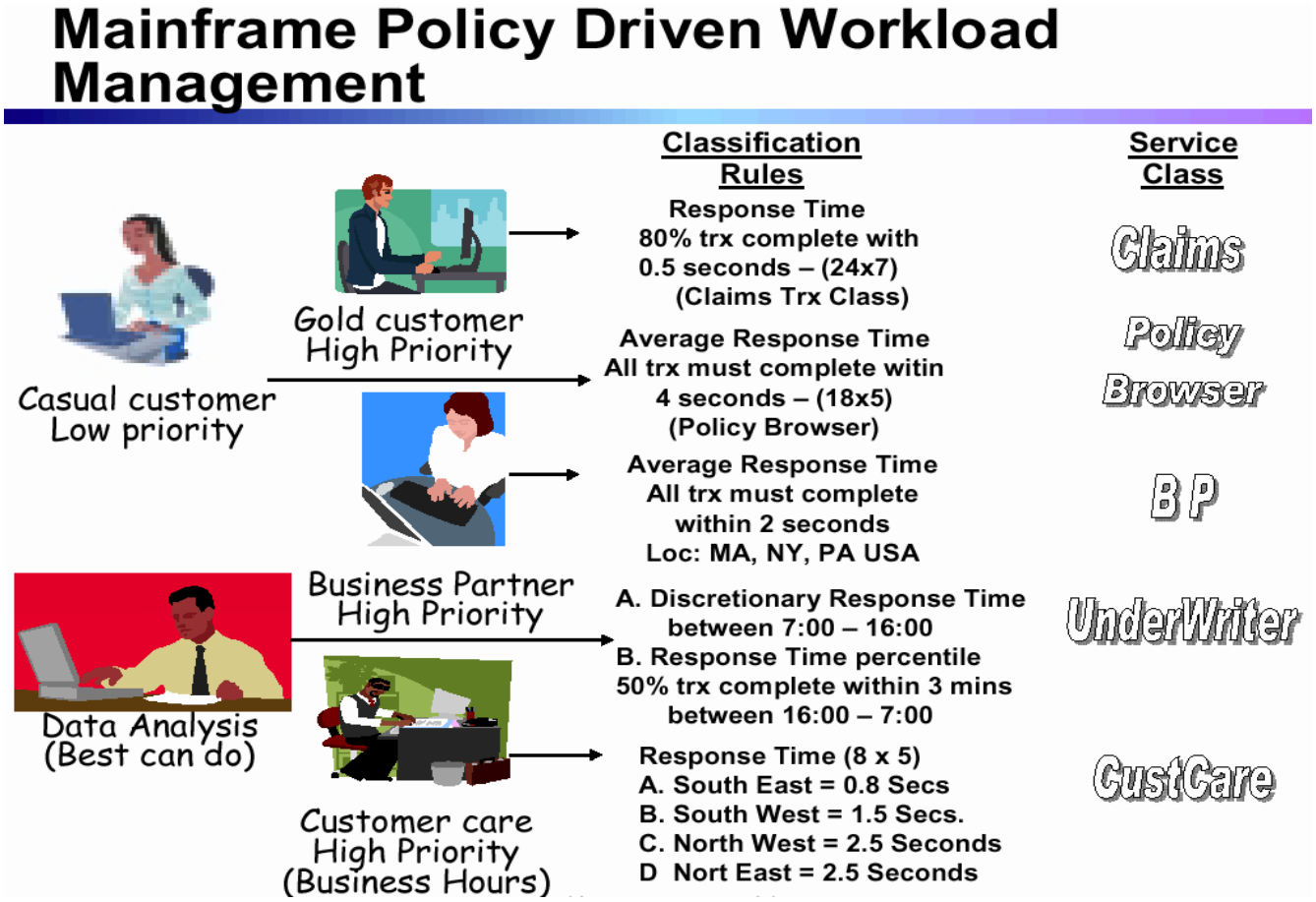
On a mainframe system, many different workloads of different types compete for the available resources. The types of workload cover things like online transaction processing, database queries or batch jobs as well as online timesharing users. The resources that are needed by these workloads are hard resources like CPU capacities, main memory, or I/O channels, as well as soft resources, like available server processes that serve transactions.

Every installation wants to make the best use of its resources and maintain the highest possible throughput. To make this possible the z/OS Workload Manager (WLM) was introduced [53] . With the Workload Manager, one defines performance goals and assigns a business importance to each goal. The user defines the goals for work in business terms, and the system decides how many resources, such as CPU and storage, should be given to it in order to meet the predefined goal. The Workload Manager will constantly monitor the system and adapt processing to meet the goals.

The z/OS Work Load Manager differs fron other workload manager implementations by using highly optimized and advanced methods and algorithms to control resources and workloads in a processor cluster [54]. It uses comprehensive adaptive algorithms to automatically meet goals established by the system administrator in an optimzed fashion [55]. For this, no prefabricated rules and/or fixed parameters are used. Required adaptions to a constantly changing work load environment occur automatically with a cadence of for example 10 seconds. [56].

The work classification and WLM observation of how the work uses the resources provide the base to manage the system. This management is done automatically and without administrator interference, based on the goals that the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class; the goal determines how much service the work in the class is able to receive. These classes of work are named "*service classes*".

Beside the goal, it is also necessary to define which work is more important than other work. In the case where several service classes do not achieve their target goals, the "*importance*" helps WLM decide which service class it should help get resources.

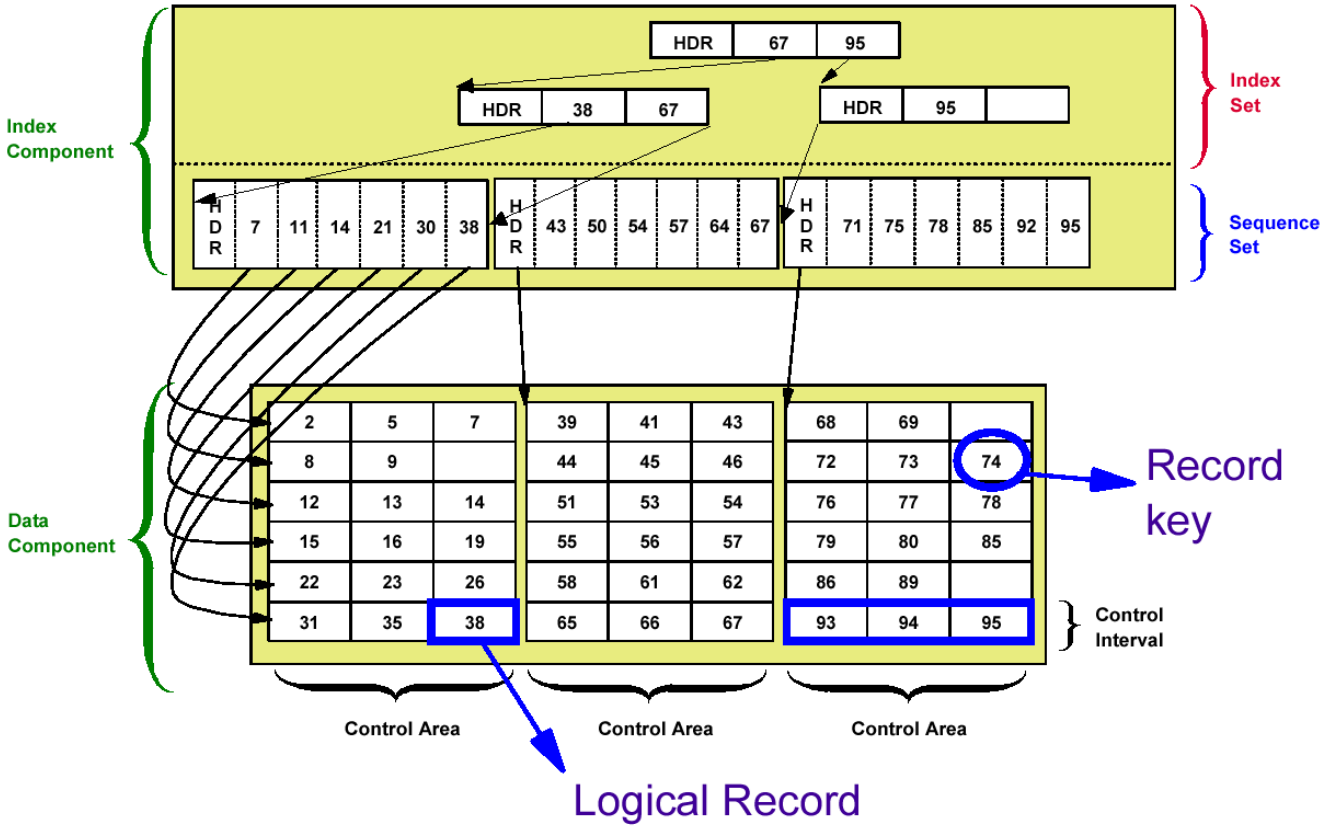


Goal oriented Work Load Manager

z/OS provides the capability to run processors at high utilization. By design, many System z customers consistently run their processors at 90+% busy for extended periods, and some will literally run at 100%.[57]. Processor utilization is usually much lower on Unix, Windows and Linux platforms running transaction and data base applications. High Processor utilizationutilization contributes significantly to the lower energy consumpton of mainframe installations.

It is possible, to anticipate dynamic load changes, and to adjust WLM parameters ahead of time. In cooperation with the IBM development laboratory in Boeblingen, Germany, the Computer Science Department of Tuebingen University developed WLM algorithms using neural networks. It was possible to predict probable load changes for a time period of up to 7 hours [58].

2.12 VSAM



VSAM stands for Virtual Storage Access Method. It is a method of managing files that is used mainly on mainframes. Specifically, VSAM can speed up access to file data by using a reverse index of records appended to files.

VSAM is a record-oriented file system. VSAM files are called datasets. In this kind of dataset, information is stored as a collection of records. VSAM records can be of any length. They are, however, organized into blocks called Control Intervals, which are measured in bytes. These Control Intervals are further organized into Control Areas, which are measured in much larger units.

VSAM can be organized into three types of dataset: Entry Sequenced Data Set (ESDS), Key Sequenced Data Set (KSDS), and Relative Record Data Set (RRDS). ESDS items are listed in the order in which they were input, regardless of any other consideration. Each item in a KSDS is assigned a unique numerical key, according to which the data set is indexed. RRDS items are organized by related record numbers, which can be set by users.

VSAM speeds up access to data in files by using an inverted index (called a B+tree) of all records added to each file.

Unix, Linux and Windows Betriebssysteme frequently use a relational database for record I/O, even when the application requires only a single table. z/OS VSAM provides this function with much superior performance [59]. No data base can match VSAM in performance if only simple functions are used.

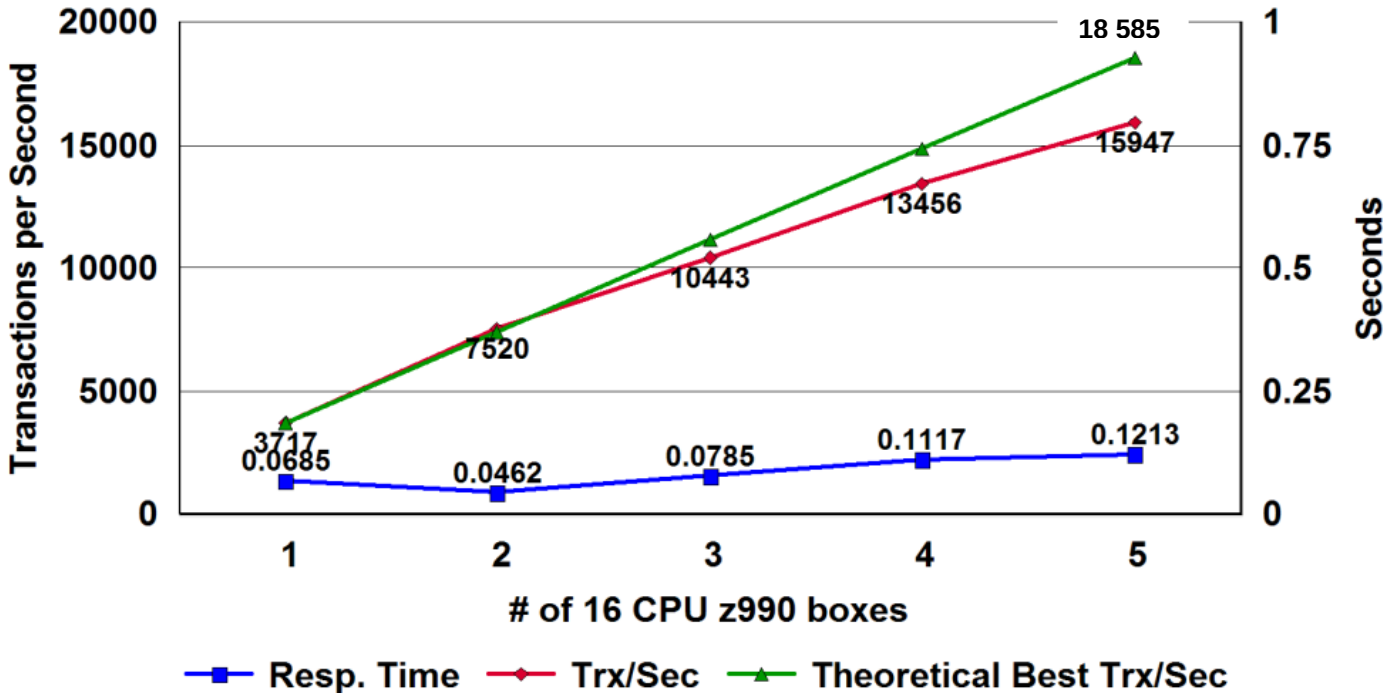
2.13 USS

Just like Windows, z/OS includes a Unix funktionality. Different from Windows, the z/OS Unix System Services are closely integrated into the z/OS operating system and heavily used. An example is the z/OS version of the WebSphere Web Application Servers, or running SAP System R/3, using all available z/OS functions. [60].

2.14 WebSphere for z/OS

Although the z/OS WebSphere version uses the same code base as the WebSphere distributed (multiplatform) version running on Linux, Unix, or Windows, many WebSphere functions are only available under z/OS. Examples are the sysplex support, or using the z/OS Work Load Manager.

Look for example at the Controller/Servant concept, the HTTP internal Transport of the controller region, the WAS cluster, the WebSphere optimized local Adapters (WOLA), or the Java Record I/O (JRIO) on z/OS. [61].



This results in outstanding z/OS WebSphere scalability characteristics. z/OS uses for many tests a standard client/server benchmark called the “Trader” application. This benchmark has been repeatedly updated over the years. Shown are results with the Trader Benchmark with up to 5 systems. A single system with 16 CPUs is able to handle 3 717 Trader transactions/s. The theoretical maximum for 5 systems and 80 CPUs is 5 x 3717 tx/s = 18 585 tx/s. The actual measured throughput is 15 947 tx/s, or 15 947 / 18 585 = 86 % of the theoretical maximum [62].

2.15 Transaction Processing Facility - z/TPF

The System z Transaction Processing Facility (z/TPF) operating system uses a unique approach to maximize transaction processing throughput. In z/TPF, the transaction processing monitor is part of the supervisor, and some, if not all, transactional applications also run in supervisor mode. No operating system outside System z has these characteristics.

In October 2006 Worldspan L.P., a global travel technology leader and the leading supplier of Internet-based travel offerings, has selected six new System z9 Enterprise Class (EC) mainframe servers with z/TPF to provide electronic data services that link approximately 700 travel suppliers around the world to millions of travelers. The mainframe platform is used for core transaction processing that is closely integrated with Intel server technology for shopping functions, such as Worldspan's industry leading e-Pricing low-fare shopping technology.

Worldspan is using the new System z9 EC to provide travel agents as well as online travel reservation sites with the ability to shop and book travel products from airlines, hotels, car rental companies and other travel service providers globally, via the Worldspan global distribution system. Additionally, the company is taking advantage of a unique Capacity On-Demand Offering to ramp up computing power on a daily basis and to accommodate spikes in reservations and ticketing [63].

2.16 Hybrid Architectures

Mainframes often serve as a platform for the development of new advanced technologies. An example are hybrid systems, consisting of a mainframe plus accelerators, for example on a cell processor or a PowerPC basis.

Mainframe applications often contain compute intensive parts that do not require typical mainframe characteristics like I/O performance or availability. In this case it is possible to offload parts of the application to a processor with a different architecture, for example Cell or PowerPC. This may be attractive whenever specific mainframe characteristics are not needed. Let us assume for example, we want to build a petaflop system using mainframe microprocessors. This would be possible, but significant parts of the microprocessor chips would never be used.

Demand for lower latency will drive co-location of hybrid transaction processing elements. A trail blazer development is Taikodom [64]. This is a Massively Multiplayer Online Role-Playing Game (MMORPG), which runs on the „Gameframe" server of a company named Hoplon. The Gameframe server consists of a regular z10 mainframe, to which a large number of cell processor blade centers are connected using Infiniband.

Data Base accelerators use a similar approch. The Tuebingen University Computer Science Department presently cooperates with the IBM Boeblingen development laboratory in the design of new hybrid architecture mathematical models and algorithms for use in the finance industry [65].

3. Leading Edge Technology

After many discussions we are unable to identify any transaction and data base server technological capabilities not available on mainframes.

We assume that present technologies like logical partitions (LPAR), Intelligent Resource Director, Hardware Storage Key Protection, or JVM Reset will become available on other server platforms within the next 10 years. For example, introducing a Coupling Facility would improve the scaling characteristics of a 16 core Intel chip.

We also assume mainframes will have introduced new not yet identified technologies at this time, and that the size of the technology gap will remain roughly the same. During the last 30 – 40 years this has been the case, and the driving forces have not changed. IBM continues to invest 1.2 Billion $ annually in the development of new mainframe hardware- and software technologies [66].

Mainframes appear to have a bright future. We predict, they will continue to be a technology leader, and will be around for the next 50 years.

4. Acronyms

AES	**Advanced Encryption Standard**
APAR	**Authorized Program Analysis Report**
APF	**Authorized Program Facility**
ARM	**Automated Restart Manager**
CCA	**Common Cryptographic Architecture**
CDS	**Couple Data Set**
CF	**Coupling Facilitx**
CFQ	**Completely Fair Queuing**
CICS	**Customer Information Control System (Transaction Server)**
CMS	**Conversational Monitor System**
CP	**Central Processor**
CU	**Control Unit**
CFCC	**Coupling Facility Control Code**
CPACF	**Co-Processor Assist for Cryptographic Functions**
CEX2	**Crypto Express 2**
DASD	**Direct Access Storage Device (IBM term for “disk”)**
DES	**Digital Encryption Standard**
DFSMS	**Data Facility Storage Management Subsystem**
DMA	**Direct Memory Access**
ECC	**(Hamming) Error Correction Code**
ERP	**Enterprise Resource Planning**
ESDS	**Entry Sequenced Data Set**
FICON	**Fibre Channel Connectivity**
GBP	**Group Buffer Pool**

GDPS	**Geographically Dispersed Parallel Sysplex**
GRS	**Global resource serialization**
HCA	**Host Channel Adapter**
HMC	**Hardware Management Console**
HSA	**Hardwase System Area**
IA32	**Intel Architecture 32 Bit**
ICF	**Internal Coupling Facility**
ICSF	**Integrated Cryptographic Services Facility**
IDE	**Integrated Development Environment**
IEF	**Interpretive Execution Facility**
IRD	**Intelligent Resource Director**
IPL	**Initial System Load (booting an operating system)**
JES	**Job Entry Subsystem**
JRIO	**Java Record I/O**
JVM	**Java Virtual Machine**
KSDS	**Key Sequenced Data Set**
LCSS	**Logical Channel SubSystem**
LPAR	**Logical Partition**
MMORPG	**Massively Multiplayer Online Role-Playing Game**
NUMA	**Non-Uniform Memory Architecture**
PFA	**Predictive Failure Analysis**
PR/SM	**Processor Resource / System Manager (LPAR Hypervisor)**
PTF	**Program Temporary Fix**
QM	**Queue Manager**
RACF	**Resource Access Control Facility**
RSA	**Rivest, Shamir, Aldeman asymetric encryption algorithm**
RU	**Recovery Unit**
RRDS	**Relative Record Data Set**
SAF	**Security Authorisation Facility**
SAP	**System Assist Processor**
SDM	**System Data Mover**
SE	**Service Element**
SMP	**Symmetric Multiprocessor**
SMP/E	**System Modification Program/Extended**
SSL	**Secure Socket Layer**
STI	**Self timed Interface (IBM alternative to the PCI bus)**
TDES	**Triple Digital Encryption Standard**
TLS	**Transport Layer Security**
TPF	**Transaction Processing Facility**
TRSM	**Tamper Resistant Security Module**
TSO	**Time Sharing Option (z/OS shell)**
USS	**z/OS Unix System Services**
VSAM	**Virtual Storage Access Method**
WLM	**Work Load Manager**
WOLA	**WebSphere Optimized Local Adapter**
XCF	**Cross System Coupling Facility**
XES	**Cross-System Extended Services**
XRC	**eXtended Remote Copy**
zHPF	**Fibre Channel High Performance FICON for System z**

5. Literature

[001] TOP500 Supercomputing List. http://www.top500.org/list/2009/11/100/.

**[02] Peter Baston: Unsafe At Any Speed?
http://www.sparcproductdirectory.com/artic-2002-jan-pb.html.**

[003] http://www.thebrainhouse.ch/gse/doku/67_GSE/GSE67_Silvio_Sassos_Corner.pdf, p.42.

**[04] Uno Bengtsson: System z Trends And Directions.
http://www-05.ibm.com/no/news/events/lsu2009/pdf/2_1_LSU_System_z_TD_Final.pdf.**

[05] T. A. Wise: I.B.M.'s $ 5,000,000,000 Gamble. FORTUNE Magazine, Sept. 1966, p.118.

[06] T. A. Wise: The rocky Road to the Marketplace. FORTUNE Oct. 1966, p 138.

[07] G. M. Amdahl, G. A. Blaauw, F. P. Brooks: Architecture of the IBM system/360. IBM Journal of Research and Development, Volume 8 , Issue 2 (April 1964), Pages: 87-101.

[08] Richard L. Sites: Alpha Architecture Reference Manual. Digital Press, Digital Equipment Corporation, 1992, ISBN 1-55558-098-X, p. IX.

**[09] IBM Redbooks: IBM System z9 109 Technical Introduction.
SG24-6669-00, July 2005.**

**[10] Charles F. Webb: "IBM z10: The Next-Generation Mainframe Microprocessor". IEEE Micro, vol. 28, no. 2, pp. 19-29, March/April, 2008.
http://www.computer.org/portal/web/csdl/doi/10.1109/M.**

**[11] Helge Lehmann, Wilhelm G. Spruth: Eigenschaften einer modernen Ein-/Ausgabe Architektur.
it - Information Technology (vormals it+ti), 2003, Volume 45, Issue 01, S. 20-29.**

**[12] Karla Arndt, James Caffrey : Detecting Soft Failures Using z/OS PFA. SHARE March 2010, Session: 2240,
http://shareew.prod.web.sba.com/client_files/callpapers/attach/SHARE_in__Seattle/S2240XX154725.pdf.**

**[13] z/Architecture Principles of Operation, p 3-8,Form No. SA22-7832-05.
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/download/A2278325.pdf?DT=20070807125005&XKS=DZ9ZBK07.**

**[14] IBM Redbooks: z9-109 Crypto and TKE V5 Update. SG24-7123-00, December 2005.
http://www.redbooks.ibm.com/abstracts/sg247123.html?Open.**

[15] Patrick Kappeler: A Brief Illustrated Overview Of System z Hardware Cryptography. March 2008.

**[16] IBM Redbooks: z/OS 1.6 Security Services Update, SG24-6448-00, July 2005.
http://www.redbooks.ibm.com/redbooks/pdfs/sg246448.pdf.**

[17] W. S. McPhee: Operating system integrity in OS/VS2. IBM Systems Journal, Volume 13, Issue 3 (September 1974), Pages: 230-252, ISSN:0018-8670.

[18] Martin King: The Development of z/OS System Integrity. Z/Journal, Aug./Sept. 2006, p. 84, http://www.zjournal.com/pdfIssue/pdfArticle/king.zJ.Aug-Sept06.pdf.

**[19] ABCs of z/OS System Programming Volume 10
SG24-6990-03, September 2008
http://www.redbooks.ibm.com/redbooks/pdfs/sg246990.pdf.**

**[20] Jeffrey B. Layton: Tuning CFQ - What Station is That? Linux Magazine, Wednesday, October 14th, 2009
http://www.linux-mag.com/cache/7572/1.html.**

**[21] M. Tim Jones: Boost application performance using asynchronous I/O.
29 Aug 2006, http://www.ibm.com/developerworks/linux/library/l-async/.**

**[22] Iain Neville:High Performance FICON for System z Technical summary for customer planning. IBM White Paper, Jan. 2009,
ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/zsw03058usen/ZSW03058USEN.PDF.**

[23] R. Cormier, R. Dugan, and R. Guyette. System/370 Extended Architecture: The Channel Subsystem. IBM J. of Research and Development, Vol. 27(3), 1983.

[24] L. W. Wyman, H. M. Yudenfriend, J. S. Trotter, K. J. Oakes: Multiple-logical-channel subsystems. IBM J. of Research and Development, Vol. 48(3/4), 2004.

[25] IBM Redbooks: System z10 Enterprise Class Technical Introduction. , SG24-7515-02, November 2009. http://www.redbooks.ibm.com/redbooks/pdfs/sg247515.pdf.

**[26] Gary DeWard Brown: zOS JCL. Wiley, 5th Edition, June 2002,
ISBN-10: 0471236357, # ISBN-13: 978-0471236351.**

[27] Mike Gilbert, Michael Dee Hester: Job Scheduling on Windows. Microsoft White Paper, Dec. 2006, http://www.microsoft.com/windowsserver/mainframe/papers.mspx.

**[28] Srinivas Devadas, Ed Suh, Larry Rudolph: Effects of Memory Performance on Parallel Job Scheduling. Proceedings of the 7th Workshop on Job Scheduling Strategies for Parallel Processing, SIGMETRICS 2001, Cambridge, MA, USA, June 2001.
http://csg.csail.mit.edu/pubs/memos/Memo-441/memo-441.pdf**

[29] Norman Bobroff, Gargi Dasgupta, Liana Fong, Yanbin Liu, Balaji Viswanathan, Fabio Benedetti, Jonathan Wagner: A distributed job scheduling and flow management system. ACM SIGOPS Operating Systems Review, Volume 42 , Issue 1 (January 2008).

[30] J. Horswill: „Designing & Programming CICS Applications“. O´Reilly, 2000. ISBN 1-56592-676-5.

**[31] Tuning Language Environment enclave storage for JVMs.
http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfht3/dfht3rr.htm.**

[32] Martin Campbell-Kelly:
ICL: A Business and Technical History: The Official History of Britain's Leading Information Systems Company. Oxford University Press, USA (March 22, 1990), ISBN-10: 0198539185, ISBN-13: 978-0198539186.

[33] Yonathan Bard: An analytic model of CP-67 - VM/370. Proceedings of the workshop on virtual computer systems. Cambridge, Massachusetts, 1973, p.170-176.

[34] D. L. Osisek, K. M. Jackson, P. H. Gum: ES/390 interpretive execution architecture, foundation for VM/ESA.
IBM Systems Journal. Vol 30 (1), 1991, p. 34-51.

[35] Gerald J. Popek, Robert P. Goldberg : "Formal Requirements for Virtualizable Third Generation Architectures". Communications of the ACM 17 (7), 1974, p. 412 –421.

[36] Robert W. Doran: Amdahl Multiple-Domain Architecture.
Computer, Volume 21 , Issue 10 (October 1988), Pages: 20 - 28
http://www.princeton.edu/~yctwo/files/readings/amdahl_multiple_domain_doran88.pdf.

[37] I. G. Siegel, B. A. Glendening, J. P. Kubala:
Logical partition mode physical resource management on the IBM eServer z990.
IBM Journal of Research and Development, Vol. 48, Issue 3-4, May 2004, p. 535 – 541.

[38] IBM Redbooks: z/OS Intelligent Resource Director.
SG24-5952-00, August 2001.
http://www.redbooks.ibm.com/abstracts/sg245952.html.

[39] IBM Redbooks: pSeries 670 and pSeries 690 System Handbook, p- 8,
SG24-7040-02, May 2003
http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg247040.pdf.

[40] Robert Vaupel, Michael Teuffel: Das Betriebssystem z/OS und die zSeries.
Oldenbourg, 2004, ISBN-10: 3486275283, ISBN-13: 978-3486275285.

[41] IBM Redbooks: System z Connectivity Handbook. SG24-5444-10, October 2009,
http://www.redbooks.ibm.com/redbooks/pdfs/sg245444.pdf.

[42] Jay Aiken, Jim Goethals, Bob Perrone: zSeries and z/OS HiperSockets Overview.
IBM System Magazine, Jan./Feb. 2003,
http://www.ibmsystemsmag.com/mainframe/januaryfebruary03/technicalcorner/9920p1.aspx.

[43] Lee Siegmund: DB2 for z/OS Features Contribute to High Availability
IBM Systems Magazine, Nov./Dec. 2005.
http://www.ibmsystemsmag.com/mainframe/septemberoctober05/administrator/10021p5.aspx.

[44] Lee Siegmund: DB2 for z/OS High Availabilty and Unplanned Outage Recovery. IBM Systems Magazine, Nov./Dec. 2005.
http://www.ibmsystemsmag.com/mainframe/novemberdecember05/administrator/9995p1.aspx?ht=.

[45] Martin Packer: DB2 Data Sharing Performance for Beginners. http://lsx.gse.org.uk/Downloads/2007%20Jan%20DB2%20Data%20Sharing%20for%20beginners.pdf.

[46] Jim Gray, Andreas Reuter: Transaction Processing: Concepts and Techniques. Morgan Kaufmann 1993, ISBN 1-55860-190-2.

[47] IBM Boulder: Coupling facility structures. http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.dshare/db2z_cfstructures.htm.

[48] IBM Redpaper: WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment. January 14, 2004, http://www.redbooks.ibm.com/redpieces/pdfs/redp3636.pdf.

[49] IBM Redbooks:System/390 Parallel Sysplex Performance. SG24-4356-03 December 1998, http://www.redbooks.ibm.com/redbooks/pdfs/sg244356.pdf.

[50] IBM Redpaper: Coupling Facility Performance: A Real World Perspective. March 2006, http://www.redbooks.ibm.com/redpapers/pdfs/redp4014.pdf.

[51] HP-UX Workload Manager User's Guide. http://docs.hp.com/en/B8844-90014/B8844-90014.pdf.

[52] Solaris 10 Resource Manager. http://dlc.sun.com/pdf/817-1592/817-1592.pdf.

[53] L. Aman, J., C. K. Eilert, D. Emmes, P. Yocom, D. Dillenberger : Adaptive algorithms for managing a distributed data processing workload. IBM Systems Journal, 36(2), 1997, 242-283.

[54] Robert Vaupel: Managing Workloads in z/OS, An overview of WLM and its major enhancements. IBM Systems Magazine, Jan./Feb.2004, http://www.ibmsystemsmag.com/mainframe/januaryfebruary04/administrator/10079p1.aspx.

[55] Michael Teuffel, Robert Vaupel: Das Betriebssystem z/OS und die zSeries. Oldenbourg Verlag München, 2004, ISBN 3-486-27528-3.

[56] IBM Redbooks: System Programmer's Guide to Workload Manager. Form No. SG24-6472-03, March 2008. http://www.redbooks.ibm.com/redbooks/pdfs/sg246472.pdf.

[57] Gary King: System z Performance. IBM White Paper, November 2, 2007, http://www-03.ibm.com/support/techdocs/atsmastr.nsf/84279f6ed9ffde6f86256ccf00653ad3/1e54eb16aae59150862573e600727399/$FILE/gking_high_util_v1.pdf.

[58] M. Bensch, D. Brugger, P. Baeuerle, W. Rosenstiel, M. Bogdan, W. Spruth: Self-Learning Prediciton System for Optimisation of Workload Managememt in a Mainframe Operating System.International Conference on Enterprise Information Systems, Funchal, 2007, p. 212-218.

[59] IBM Redbooks: VSAM Demystified. SG24-6105-01, September 2003, http://www.redbooks.ibm.com/redbooks/pdfs/sg246105.pdf.

[60] IBM Redbooks: ABCs of z/OS System Programming Volume 9 SG24-6989-01, November 2005. http://www.redbooks.ibm.com/redbooks/pdfs/sg246989.pdf.

[61] IBM Redbooks: WebSphere for z/OS V6 Connectivity Handbook. SG24-7064-02, December 2005, http://www.redbooks.ibm.com/redbooks/pdfs/sg247064.pdf.

[62] Mike Cox et al.: IBM WebSphere Application Server for z/OS Version 6, A performance report. ftp://ftp.software.ibm.com/software/zseries/pdf/WASforzOSv6PerformanceReport.pdf.

[63] IBM Redbooks: z/TPF and WebSphere Application Server in a Service Oriented Architecture. April 2007, SG24-7309-00, http://www.redbooks.ibm.com/redbooks/pdfs/sg247309.pdf.

[64] The whole Universe is running on Gameframe. http://www.taikodom.com/content/view/109.

[65] Thomas Grosser, Alexander C. Schmid, Markus Deuling, Hoang-Nam Nguyen, Wolfgang Rosenstiel: Off-loading Compute Intensive Tasks for Insurance Products Using a Just-in-Time Compiler on a Hybrid System. CASCON'09, Proceedings of the 2009 conference of the center for advanced studies on collaborative research, Nov. 2009.

[66] Timothy Prickett Morgan: The IBM Mainframe Base. http://www.itjungle.com/big/big071007-story01.html.

All Internet References listed here are reproduced on a mirror under http://www-ti.informatik.uni-tuebingen.de/~spruth/edumirror/report.html .