zAAPs and zIIPs: Increasing the strategic value of System z

L. W. Wyman
J. Castaño
J. P. Kubala
R. J. Maddison
B. R. Pierce
R. R. Rogers

With the addition of IBM System z™ application assist processors (zAAPs) and integrated information processors (zIIPs) to the portfolio of special-purpose IBM System z processors, the reinvention of the IBM mainframe continues. Jointly, zAAPs and zIIPs provide significant IBM System z9™ integrated and cost-effective processing cycles for today's strategic Java™ and DB2® for z/OS® programming platforms which are increasingly fundamental to enterprise-class business environments. Overviews of zAAPs and zIIPs are presented that describe their functionality, design, and use by the z/OS operating system to achieve the execution of both Java and z/OS DB2 programming functions.

Introduction

In contemporary computer networks, it is a common practice to interconnect multiple computing systems, typically called *servers*, in a manner that segregates each of the interconnected servers into separate usage classes. The classes are based on the various functions each server is configured to execute. For example, one or more of the interconnected servers may be assigned the task of executing application programs, and others may be assigned the task of executing database programs that access and manage one or more of the network databases. Networks that are structured in this manner are typically described as distributed multitier server networks, as depicted in **Figure 1(a)**.

The structuring of distributed multitier networks and the assignment of work to each of the servers within the network is typically determined by a number of factors such as processing speed, functional capabilities, reliability, scalability, cost of acquisition, and cost to operate and maintain. Typically, the computing functions within these networks that require the highest degrees of reliability, availability, data integrity, and scalability are allocated to the database servers that control the management and access of the network user's data. Thus, the core computing elements of many networks—in terms of user data availability, integrity, and reliability—lie with data servers such as the IBM System z9*. Such database servers are responsible for accessing, updating, and maintaining the centralized and integrated databases upon which many companies rely for their associated business processes. As such, they are typically the most reliable and functionally robust elements within the network, but they can also represent the most expensive servers in terms of initial cost of ownership.

Within many distributed networks, the number of application servers has increased significantly because of the programming technologies being used to develop and implement timely and cost-effective web-based applications. This is especially true for applications designed to support the evolving strategic e-business and on-demand business models. Programming languages such as Java** and Extensible Markup Language (XML) for document processing play a major role in developing new strategic service-oriented architecture web-based applications. They provide significant advantages in programming development productivity, reduced development costs, and reduced time for application deployment. These application-based programming technologies present a high level of programming abstraction to the program developer, are independent of the specific computer architectures of the servers on which they are deployed, and provide open-standard programming interfaces to render application programs operable on all server platform implementations. However, these "application can operate anywhere" programming models have resulted in significantly

0018-8646/07/$5.00 © 2007 IBM

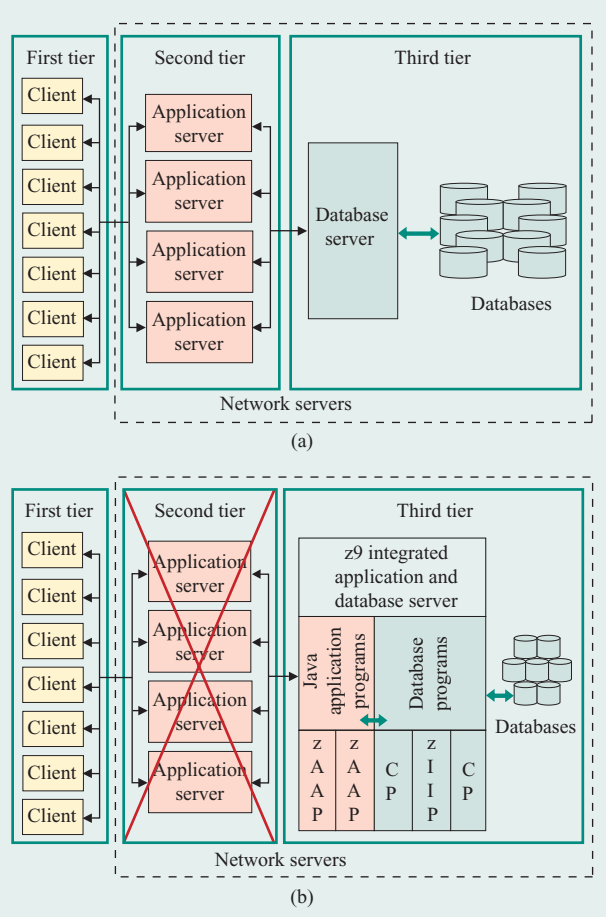

Figure 1

Multitiered networks: (a) distributed; (b) simplified by the use of integrated zAAPs and zIIPs.

increased processing requirements in terms of the total number of processor instructions that must be executed to support a given application. For example, Java- and XML-based web applications are emerging that can require the execution of millions of computer instructions to process a single application transaction. Consequently, such applications are often not cost-effective when deployed on more expensive server platforms within the information processing network.

Historically, attempts to simplify and reduce the overall cost of distributed multitier networks have met with varying degrees of success. A method often chosen for minimizing network sprawl and complexity is to reduce the number of application servers and integrate their application programming functions into more functionally rich database servers within the network. While such strategies have proven to significantly improve the overall efficiency and management of the network, the increased initial cost of ownership that is frequently associated with more robust database servers can be a barrier to entry. Consequently, the proliferation of dedicated application server farms is often the reality for many information technology environments. However, while the cost to purchase and deploy additional application-only servers may offer an attractive initial alternative, application server proliferation often results in unexpected increases in the total cost of network ownership. This typically becomes the reality for many mature yet ever-growing computing environments, as the total number of server elements within the environment increases to meet the growing demands of the business. This server proliferation often results in decreased reliability, additional complexity to manage, and inefficient utilization of the application servers within the information network.

To address these problems, IBM has developed the System z* application assist processor (zAAP) and the System z integrated information processor (zIIP). These are special-purpose processing engines integrated within the System z9 scalable symmetric multiprocessor (SMP) memory-coherent infrastructure for use by the z/OS* operating system [**Figure 1(b)**]. zAAPs execute strategic Java programming applications and their associated XML documents. zIIPs are designed to process portions of the z/OS IBM DB2* relational database management functions. Collectively, these two processor additions can execute a significant percentage of the total processing cycles consumed by these programming functions. Consequently, zAAPs and zIIPs free up much of the System z9 general-purpose central processor (CP) capacity for other uses. Also, because of the hardware and software cost advantages of zAAPs and zIIPs compared with that of System z9 CPs, they collectively provide a cost-competitive means to consolidate strategic Java and related application programming and a more cost-effective z/OS and DB2 database processing platform. As an integrated application and database server solution, they enable the IBM System z9 to provide greater performance, function, and cost-competitiveness when compared with typically less efficient and more complex distributed network solutions prevalent in today's information technology marketplace.

Integrated processing engines

Both zAAPs and zIIPs are integrated into the System z9 server platform central processor and memory attachment infrastructure to form a memory-coherent, yet heterogeneous, processor complex. The result is that both applications and associated database subsystem functions can be executed in a highly efficient manner. Also, by exploiting zAAPs and zIIPs with the z/OS operating system, the System z9 provides significant

functional benefits to both the application and database programs because both are now operating on the most reliable server within the network. Additionally, the Java application programs executed on zAAPs implicitly benefit from the System z9 and z/OS advantages of qualities of service, server scalability and reliability, and z/OS computing capabilities, such as dynamic workload balancing, multilevel security, logical partitioning, and other robust server virtualization capabilities.

The Java application programs and their associated database subsystem programs can interact with each other more efficiently by using System z9 SMP memory exchanges of data; that is, they can directly communicate and share data with each other. This means that they require neither physical network wires nor the associated communication programming stacks that would otherwise be necessary to interconnect physically separate application and database servers. This eliminates the requirement for such elements as Ethernet network adapters, the associated Transmission Control Protocol/Internet Protocol (TCP/IP) programming stacks, firewalls, data conversion, data compression/decompression, or data encryption/decryption overheads. System z and z/OS integrated interprogram communication using System z9 memory-based accesses and data-sharing capabilities can significantly reduce the overall complexity of the network, increase the total network reliability, and significantly reduce network transaction processing overhead.

Performance tests that demonstrate this increased efficiency were executed at the IBM Washington Systems Center, operating both Java application transactions and their associated database programs in the same LPAR on a System z platform. The result was an improvement by as much as 77% in transaction processing efficiency as measured by average central processing unit (CPU) time per transaction. Additionally, the total number of bytes transferred between the application and the database subsystem was reduced by as much as 99% when compared with operating the application and database subsystem on separate physical server platforms [1].

Using zAAPs and zIIPs

zAAPs and zIIPs render their use transparent to the Java programs that execute on zAAPs and the associated DB2 database programs that execute on zIIPs. The z/OS control program, operating in conjunction with the System z9 Processor Resource/Systems Manager* (PR/SM*) firmware [2], commonly called *the logical partition (LPAR) hypervisor*, transparently assigns and dispatches the execution of Java programs on the zAAP processors and the execution of appropriate DB2 database processing programs on zIIPs. In both cases, this is accomplished without the awareness or involvement of the Java programs or the DB2 programs.

Both zAAPs and zIIPs assist the System z9 CPs and are configured using the same LPAR definition and activation processes as provided for the CPs configured to an LPAR. Each z/OS LPAR may have one or more zAAPs and zIIPs configured to them along with their associated CPs. However, as zAAPs and zIIPs are designed to assist CPs, they have some unique characteristics. For example, neither zAAPs or zIIPs can be the target of an initial program load (IPL)[1], the maximum number of configured zAAPs and zIIPs cannot exceed the total number of CPs for a given system, and, depending on the System z model, zAAPs and zIIPs do not necessarily operate at the same speed as the CPs (see the section on zAAP and zIIP architecture characteristics below).

Program requirements

zAAPs are designed to execute Java programs, while zIIPs, are designed to execute z/OS DB2 database processing programs, which creates certain programming requirements. For zAAPs, the requirements are the following:

- z/OS 1.6 (or z/OS.e 1.6), or later.
- IBM Software Development Kit (SDK) for z/OS, Java 2 Technology Edition, V1.4, or later with program temporary fix (PTF) for authorized program analysis report (APAR) PQ86689 [the IBM Java Virtual Machine (JVM) that controls and executes Java programs].

Additionally, subsystems and applications that use the SDK 1.4 JVM and its subsequent releases automatically exploit zAAPs. These include the following:

- IBM WebSphere* Application Server (WAS) releases 5.1 and 6.0.1, or later ...
- IBM Customer Information Control System/Transaction Server (CICS*/TS) release 2.3, or later ...
- IBM DB2 for z/OS version 7, or later ...
- IBM Information Management System (IMS*) version 8, or later ...
- IBM WebSphere Business Integration (WBI) version 5 for z/OS Java batch programs.

For zIIPs, the program requirements [3] are as follows:

- z/OS 1.6 (or z/OS.e 1.6), or later ...
- IBM DB2 for z/OS version 8 or later, with the following maintenance:

[1] An IPL is the System z method for booting an operating system into a logical partition and activating it.

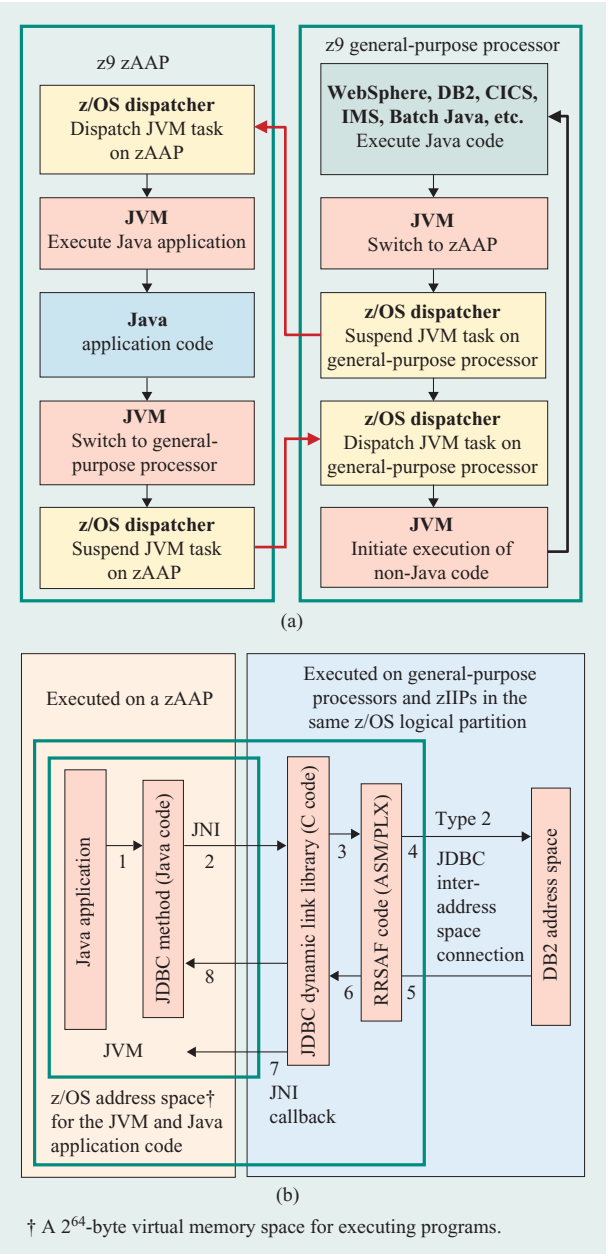

Figure 2

Programming flow: (a) z/OS switching between general-purpose logical processors and zAAP logical processors. (b) Java application interacting with the DB2 subsystem to access a DB2 database. (RRSAF: Resource recovery services attach facility; ASM/PLX: assembly and PLX code.)

- Distributed Rational Database Architecture (DRDA): PK18454.
- Utilities: PK19920.
- Star schema parallelism: PK19921.

Directing work to zAAPs

The process of assigning and executing zAAP-eligible programs to their corresponding physical zAAP processors is accomplished by a four-phase process. Phase 1 is the process by which the Java program is made eligible for execution on zAAP processors. Phase 2 is the process by which the z/OS dispatcher, operating on a CP, suspends execution of the zAAP-eligible programs. Phase 3 is the process by which the System z9 PR/SM selects and dispatches the zAAP logical processor to a corresponding zAAP physical processor (see the section below on controlling the execution of zAAPs and zIIPs). Phase 4 is the process by which the z/OS dispatcher, operating on a zAAP, selects and dispatches the zAAP-eligible programs on their respective zAAP logical processors operating on physical zAAP processors.

The JVM that controls and executes Java programs uses an internal secure "switch" programming interface to request the z/OS control program to make it eligible for execution on a zAAP. This z/OS switching service is available only to the appropriate IBM JVMs. Attempts by other programs to use this interface will not result in a switching action by z/OS.

The IBM JVM uses this interface prior to executing a Java program. When the JVM is subsequently dispatched on a zAAP, the JVM and its associated Java program execute on a zAAP until the Java program ends, abnormally terminates, or, more typically, requests service by use of a Java Native Interface (JNI) application programming interface (API) for a non-Java program function. For example, if the Java program requests data from the z/OS DB2 subsystem through the use of a Java database connectivity (JDBC) JNI, the JVM again uses the z/OS switch interface service to request that z/OS be redispatched to request its execution on a CP in order to activate the non-Java programming components of the JDCB function on the CP. Subsequently, when the non-Java functions have completed and control is returned to the JVM, it again requests that z/OS execute it on a zAAP in order to resume execution of the Java program. This programming flow is depicted in **Figure 2(a)**. **Figure 2(b)** depicts the program flow between a Java application program, operating on a zAAP in its own z/OS address space, and the DB2 subsystem, operating on CPs and zIIPs in a DB2 address space, in order to access or modify data in a DB2 database.

In addition to Java application programs, generally available XML parsing programs that are coded in Java implicitly benefit from executing on zAAPs. Such XLM parsing programs derive the same functional and price/performance benefits as all other Java-coded programs that execute on zAAPs.

Controlling zAAP-eligible work

To ensure that Java program execution meets the desired transaction response time and to provide user control of the overall Java workload for capacity planning and management purposes, z/OS provides two Java execution options, as specified by the z/OS `SYS1.PARMLIB` dataset parameter `IFAHONORPRIORITY`[2] = `YES/NO`.

When `IFAHONORPRIORITY = YES` is specified, z/OS executes both Java and non-Java programs as follows:

- All Java-eligible programs are executed on zAAPs in priority order.
- All non-Java programs are executed on CPs in priority order. Java-eligible programs may also be executed on CPs in priority order, but only when the zAAP processors are overcommitted and require assistance to meet the processing demands of the overall Java programming workload.

When `IFAHONORPRIORITY = NO` is specified,

- All Java programs are executed on zAAPs in priority order.
- All non-Java programs are executed on CPs in priority order. Java programs may also be executed on CPs, but only when there are no non-Java programs to dispatch on the CPs; that is, prior to the processor entering an enabled wait state[3].

In the unlikely event that the zAAP processors become unavailable for program execution (that is, they enter the not-operational state or they are varied offline), z/OS dispatches the Java-eligible programs to CPs, just as when no zAAP processors are configured to the LPAR.

Directing work to zIIPs

zIIPs execute those z/OS programs that are structured to operate under control of z/OS-preemptable enclave service request blocks (SRBs)[4] [4]. An enclave is a z/OS construct that allows a unit of work or transaction, such as a distributed relational database architecture (DRDA) [5] request for DB2 to be assigned a goal by the z/OS Workload Manager (WLM) [4] on the basis of customer-provided rules. The execution threads, such as SRBs, that perform the transaction are assigned priority and are actively managed by WLM to achieve the assigned goal. The process of assigning and executing zIIP-eligible SRB programs is similar to the four-phase process used for zAAP-eligible programs, as described in the preceding subsection. However, unlike zAAPs, which execute only Java programs, zIIPs can be exploited to execute any programming function that is structured to operate under control of enclave SRBs[5]; that is, zIIPs are not restricted to specific programming languages, but rather to a specific type of z/OS-dispatchable unit. The z/OS interface used to schedule enclave SRB programs as eligible for execution on zIIPs is proprietary and is therefore not described. However, like the proprietary zAAP switching interface, this zIIP-enabling interface results in the eligible programs executing on zIIPs either wholly or in part as determined by two elements: the requirements of the program that scheduled the zIIP SRB enclave and the overall zIIP processing workload as controlled by the z/OS WLM.

ZIIPs are not subject to the zAAP `IFAHONORPRIORITY` parameter. However, they are managed in a manner similar to zAAPs when `IFAHONORPRIORITY = YES` is specified; that is, non-zIIP-eligible programs execute only on CPs in priority order. They may also execute zIIP-eligible programs in priority order under the following conditions:

- zIIPs are overcommitted and require CP assistance.
- zIIP-eligible programs have met or exceeded their zIIP processing requirements as specified by their associated zIIP scheduling program.
- zIIP processors become unavailable for program execution.

Which DB2 functions execute on zIIPs

z/OS DB2 Version 8 is the initial IBM subsystem to exploit zIIPs, and this exploitation can significantly enhance the cost and performance benefits of DB2 for System z9 users. In the future, it is expected that other strategic System z program offerings will be enhanced to take advantage of the benefits that zIIPs provide.

Currently, DB2 provides zIIP enablement for the following functions:

- *Network-connected applications:* For applications that execute on other server platforms but access a DB2 database hosted on a System z9, a portion of the DB2 subsystem functions that accept and process Structured Query Language (SQL) calls from the remote application over a TCP/IP DRDA connection are enabled for zIIPs. Additionally, the System z HiperSockets*[6] facility may also be used in conjunction with the DRDA connector by application

[2]Integrated facility for applications (IFA) is the programming name assigned to zAAPs.

[3]An enabled wait state is the processor state in which no instructions are executed and the processor is enabled to accept an interrupt in order to place it back in the instruction execution state—for example, when an I/O or timer interrupt condition is detected that causes the processor to again execute an instruction stream appropriate for the type of interrupt condition accepted by the processor.

[4]An SRB is the z/OS data structure used to dispatch and control the execution of a high-performance program.

[5]Enclave SRBs are collections of SRBs representing a business unit of work or transaction that are managed as a group.

[6]HiperSockets is the System z9 facility that provides a virtual LAN connection for TCP/IP communications between programs operating in separate logical partitions.

programs operating in other LPARs on the same System z9 database server to access DB2. Examples of the application workloads that may be running on the remote server include business integration (BI), enterprise resource planning, and customer relations management applications.

- *Data warehousing applications:* Portions of database queries that utilize DB2 star schema parallel SQL queries [6] are zIIP-eligible. For example, BI applications may use such queries.

Some DB2 utility functions [7] that are used to maintain index maintenance structures (e.g., `Load`, `ReOrg`, and `ReBuild Index`) are zIIP-eligible—specifically, those portions of these utility functions that execute as enclave SRBs.

Controlling zAAP and zIIP execution

As with previous System z mainframes, the PR/SM LPAR hypervisor controls the creation, activation, and execution of LPARs. This includes the assignment of processors, memory, and I/O to each partition. As with System z9 CPs, zAAPs and zIIPs may be configured only to Extended System Architecture (ESA)-mode [8] LPARs[7]. Each ESA-mode partition must have at least one CP and may have one or more zAAPs and zIIPs, up to a maximum of 54 total processors. However, the number of zAAPs or the number of zIIPs, individually, cannot exceed the number of CPs for a given system configuration. For example, if the configuration has five installed CPs, it can have a maximum of five zAAPs and five zIIPs.

zAAPs and zIIPs may be defined as either dedicated or shared, just as with the CPs assigned to a z/OS LPAR. However, regardless of processor type, dedicated and shared processors may not be concurrently assigned to the same LPAR. Dedicated zAAPs and zIIPs and their corresponding CPs are used exclusively by the z/OS operating in the LPAR to which they are assigned. More typically, as with shared CPs, shared zAAPs and zIIPs are available for use by all LPARs for which shared zAAPs and zIIPs are defined.

Just as it does for CPs, the LPAR hypervisor creates zAAP and zIIP logical processors and dispatches them to the corresponding zAAP and zIIP physical processors in order to execute the programs residing in the memory of each LPAR to which they are configured (**Figure 3**). Each zAAP and zIIP logical processor comprises processor hardware, millicode, and logical partitioning firmware controls, which collectively represent the physical processor. Just as with CPs, each zAAP and zIIP logical processor contains a complete set of physical processor controls and associated operating states necessary to execute the logical processor on the hypervisor-selected physical processor. For example, if a zAAP is configured to two LPARs, each partition has a separate zAAP logical processor configured to it. In turn, the z/OS control program provides independent zAAP and zIIP logical processor controls in order to dispatch the appropriate programming tasks and SRB programs to their associated zAAP and zIIP logical processors.

The hypervisor creates and maintains separate physical processor pools for each processor type—CPs, zAAPs, zIIPs, Integrated Facility for Linux** processors (IFLs), and Integrated Coupling Facility processors (ICFs)—and uses the proprietary z/Architecture* start interpretive execution (SIE) virtualization technology provided by System z to define, initiate, and control the execution of zAAP and zIIP logical processors on their corresponding physical processors. This architecture provides a logical processor state description that is loaded into the physical zAAP and zIIP instruction processing controls when the SIE instruction is executed by the hypervisor. In turn, the SIE instruction activates the physical processor on behalf of its associated LPAR. This process is depicted in **Figure 4**. Once activated, zAAP and zIIP logical processors typically remain active on behalf of their associated partitions until one of the following conditions is encountered:

- It is placed in a wait state by the OS; for example, when there are no zAAP- or zIIP-eligible programs to be executed.
- A preemptive interrupt condition—such as a time-slice end interruption condition or, in the case of zIIPs, an I/O interruption condition associated with another LPAR—is recognized by the hypervisor. In both of these cases, the hypervisor is given control and typically dispatches the physical processor to another sharing partition.
- z/OS executes an instruction for which the hypervisor must take a special action in order to assist in the execution of the instruction. For example, the OS executes a low-frequency privileged instruction for which the processor SIE controls do not interpretively execute.
- The physical processor encounters an error condition that requires assistance from the hypervisor to recover.
- An operator action is recognized that requires hypervisor assistance: For example, the logical processor is varied offline.
- The LPAR is placed in a nonexecutable state: For example, the operator deactivates the LPAR.

To control the percentage of zAAP and zIIP processing capacity for each LPAR that shares zAAPs and zIIPs, the

[7] z/OS operates only in ESA-mode LPARs.

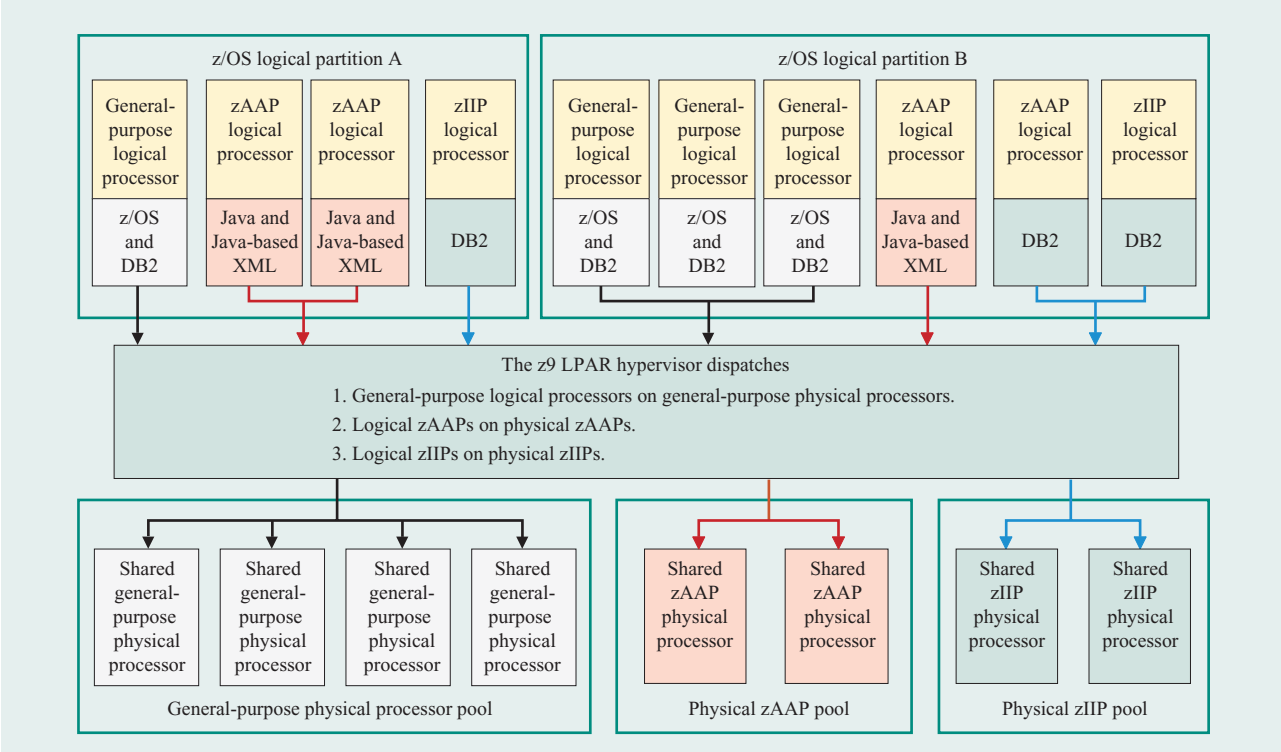

Figure 3

Dispatching CP, zAAP, and zIIP logical processors to their respective physical processors.

LPAR activation profile [2] provides individual zAAP and zIIP processing weights, like those provided for CPs; that is, CPs, zAAPs, and zIIPs each have separate initial, minimum, and maximum processing weights assigned to them for each sharing partition. The hypervisor maintains separate physical CP, zAAP, and zIIP processor pools and uses these weight specifications to control the percentage of physical processor capacity from each processor pool for each sharing partition [2]. However, unlike CPs, zAAPs and zIIPs are not subject to the z/OS WLM soft-capping function[8]. While the z/OS WLM monitors and provides usage statistics for both zAAPs and zIIPs, it does not dynamically adjust zAAP and zIIP shared processor weights, as is possible for shared CPs.

Allocating physical zAAPs and zIIPs to System z9 books

The System z9 physical packaging structure provides from one to four processor books[9]. Depending on the model, each book provides either 12 or 16 processing units and a 40-MB shared level 2 (L2) cache. A high-speed communications ring interconnects the L2 caches of each book with all other configured books to provide a shared, memory-coherent, SMP L2 cache structure for all assigned processing units.

To maximize the performance of CP, zAAP, and zIIP memory accesses, System z9 millicode assigns each of these different processor types in a manner that is designed to minimize multibook L2 cache coherency overhead; that is, to increase the probability that a processor memory access can be completed without having to communicate with the L2 cache in either the L2 cache of an adjacent book or the L2 cache of the book that is two books away on the L2 cache communications ring. To accomplish this, the installed CPs, zAAPs, and zIIPs are assigned in a manner that minimizes their placement to as few books as possible; that is, they are clustered together when possible. Additionally, they are segregated from the books containing ICF and IFL processors when possible. This is accomplished as follows:

- CPUs, zAAPs, and zIIPs are clustered together, starting with book 0, then book 2, then book 3, and

[8]Soft-capping is the process in which the z/OS WLM interacts with the logical partition hypervisor to dynamically adjust the percentage of shared processor capacity that a logical partition may use [2].
[9]A processor book is the physical packaging method used by System z9 to contain such elements as system processors, cache, memory cards, and I/O connectors.

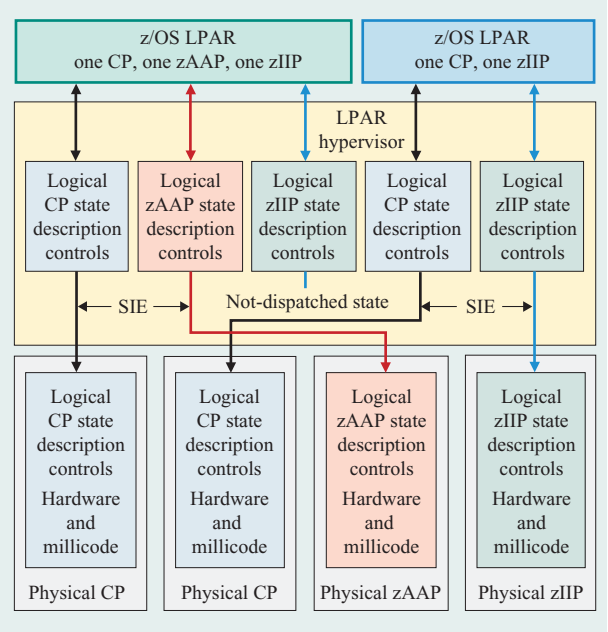

Figure 4

PR/SM activation of logical CPs, zAAPs, and zIIPs on their associated physical processors using the start interpretive execution (SIE) instruction.

finally book 1 when necessary. Within each book, CPs are assigned first, zAAPs are assigned next, and zIIPs are assigned last.

- ICF and IFL processors are clustered together starting with book 1, then book 3, then book 2, and finally book 0 when necessary. Within each book, ICFs are assigned first, and IFLs are then assigned.

Monitoring and reporting zAAP and zIIP usage

Collectively, z/OS, WLM, and the resource measurement facility (RMF) [9] monitor and report zAAP and zIIP consumption metrics using the same recording and reporting facilities as provided for general-purpose CPs. The appropriate z/OS service management facility (SMF) resource utilization records are used to record zAAP and zIIP activity numbers such as individual zAAP and zIIP utilization and collective averages for both. These statistics are then used by RMF and DB2 instrumentation to generate various processor resource reports, such as processor workload, activity, and enclave reports.

zAAP and zIIP architecture characteristics

zAAP and zIIP processors are capable of executing all z/Architecture-mode instructions necessary to operate z/OS and the zAAP and zIIP associated subsystem and application programs. Additionally, several privileged supervisory instructions used by z/OS to control execution of zAAPs and zIIPs (such as the signal processor instruction, store system information instruction, and read CPU information instruction) are enhanced to accommodate zAAP and zIIP processor types. As with general-purpose CPs, zAAPs and zIIPs may be added or removed by the System z9 dynamic capacity on/off on demand facility [10], assuming that zAAP and zIIP dynamic additions do not exceed the maximum number allowed.

However, zAAPs and zIIPs differ from general-purpose CPs in several ways:

- zAAPs are not enabled to process I/O interrupts or time-of-day clock interrupts.

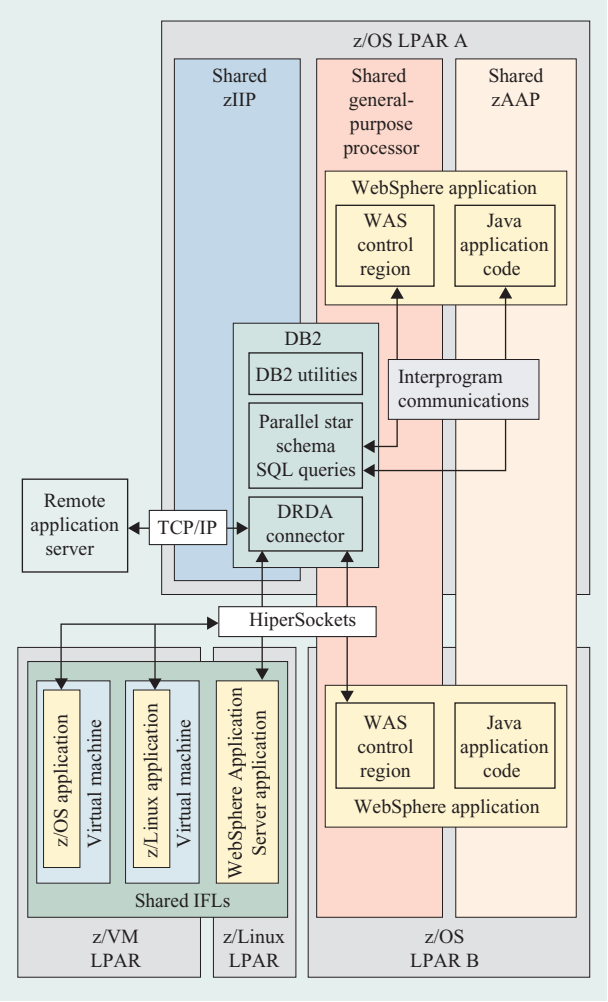

Figure 5

Using zAAPs and zIIPs in a typical System z operating environment.

- zAAPs and zIIPs are not subject to initial program loading (IPL[10]); that is, neither can be specified as the target processor for an operator-initiated LOAD PROCESSOR function, often referred to as an LPAR boot, or any of the System z9 LOAD derivatives.
- A few other operator-initiated functions, such as the PSW[11] RESTART and the STOP function, and their derivates, are either not available for zAAPs or zIIPs or do not affect zAAP and zIIP operating states [2].

Summary

zAAPs and zIIPs can individually provide significant benefits in terms of increased network integration, workload efficiency, and reduced information network costs. When viewed collectively with other System z9 capabilities, these advantages become even more apparent. Benefits of zAAPs and zIIPs can be derived indirectly by application programs operating in other LPARs on the same System z9 server and by application programs operating on remote application servers, as depicted in **Figure 5**.

For example, application programs operating in separate z/OS, z/VM*, or z/Linux LPARs on the same System z9 server as their associated DB2 partition may all communicate with the z/OS LPAR DB2 subsystem operating on zIIPs and CPs by use of the System z9 HiperSockets facility. In addition to the reliability, scalability, integrity and other z9 advantages these applications implicitly accrue by operating on the same System z9 platform, they can also have integration benefits similar to those that would be available if they were operating in the same LPAR as the DB2 subsystem. By using HiperSockets, all TCP/IP communications among such applications and their associated DB2 subsystem are performed at memory speeds faster than that of a remote network adapter operating at Ethernet speeds. Additionally, the associated remote server TCP/IP communication overheads such as data conversions, data compression and decompression, and data encryption and decryption can be eliminated. Applications configured to operate in either z/VM or z/Linux IFL partitions benefit from the advantage of executing on System z9 IFL processors, which provide cost benefits similar to those of zAAPs. Finally, remote applications operating on different physical servers can benefit indirectly from the reduced cost per database transaction when zIIPs are used to assist processing of the DB2 requests.

With the advantages provided by the IBM System z9 zAAPs and zIIPs, the mainframe evolution continues. Its balanced capabilities support the integration of both traditional enterprise-class workloads and the ever-increasing web-based workloads onto an enterprise-class business server that is both world-class and renowned for its reliability, scalability, and robust functionality.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Sun Microsystems, Inc. or Linus Torvalds in the United States, other countries, or both.

References

1. IBM Corporation, "Optimizing WebSphere for z/OS Performance," White Paper; see *http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100558*.
2. IBM Corporation, *System z9 109 Processor Resource/Systems Manager Planning Guide* (SB10-7041-01a), October 10, 2006; see *http://www-1.ibm.com/support/docview.wss?uid=isg22526a41d800842b98525701c007281db*.
3. zIIP program requirements; see *http://www-03.ibm.com/systems/z/ziip/gettingstarted/prereqs.html*.
4. P. Bari, P. Cassier, A. Defendi, J. Hutchinson, A. Maneville, G. Membrini, and C. Ong, IBM Corporation, "System Programmer's Guide to: Workload Manager," IBM Redbooks, January 19, 2006; see *http://www.redbooks.ibm.com/abstracts/sg246472.html?Open*.
5. B. Steegmans, N. Armstrong, C. Cemiloglu, S. V. Kumar, and S. Todokoro, IBM Corporation, "Distributed Functions for DB2 for z/OS and OS/390*," IBM Redbooks, June 30, 2003; see *http://www.redbooks.ibm.com/abstracts/sg246952.html?Open*.
6. S. Podcameni, V. Anavi-Chaput, V. L. Hicks, and P. Bruns, IBM Corporation, "Data Warehousing with DB2 for OS/390," IBM Redbooks, December 17, 1997; see *http://www.redbooks.ibm.com/abstracts/sg242249.html?Open*.
7. B. Steegmans, R. Garcia, S. Kaschta, R. Kumar, and M. Parbs, IBM Corporation, "DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More," IBM Redbooks, May 2004; see *http://www.redbooks.ibm.com/redbooks/pdfs/sg246079.pdf*.
8. IBM Corporation, *z/Architecture Principles of Operation*; see *http://www-03.ibm.com/servers/eserver/zseries/zos/bkserv/r3pdf/zarchpops.html*.
9. IBM Corporation, *IBM z/OS Resource Measurement Facility Programmer's Guide* (SC33-7994-04); see *http://publibz.boulder.ibm.com/epubs/pdf/erbzpg30.pdf*.
10. F. Injey, G. Chambers, M. Gasparovic, P. Hamid, B. Hatfield, K. Hewitt, D. Jorna, and P. Kappeler, IBM Corporation, "IBM System z9 Enterprise Class Technical Guide," IBM Redbooks, December 2006; see *http://www.redbooks.ibm.com/redbooks/pdfs/sg247124.pdf*.

Received March 21, 2006; accepted for publication June 15, 2006; Internet publication January 9, 2007

[10]IPL is the z/Architecture term for the processor program loading and program activation function. The operator LOAD PROCESSOR function and its derivatives are the application of the z/Architecture IPL function on System z systems.

[11]PSW, or Program Status Word, is the processor hardware logic that contains processor state information, such as the current instuction execution address, condition code, and other state information used to control instruction sequencing.

Les W. Wyman *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (lwyman@us.ibm.com)*. Mr. Wyman retired in 1993 as a Senior Technical Staff Member and rejoined IBM in 1999. He has held numerous technical and technical leadership positions in programming systems, channel engineering, and systems architecture, including the mainframe zSeries* logical partitioning and multiple high-performance virtual machine conceptualizations and architectures, the multiple logical channel subsystem conceptualization and architecture, the queued direct I/O (QDIO) architecture, System z zAAP conceptualization, and others. Mr.Wyman has achieved the IBM Ninth Plateau Invention Achievement Award and has received numerous IBM division and corporate technical achievement awards.

José Castaño *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (castano@us.ibm.com)*. Mr. Castaño received a B.B.A. degree from Pace University. He is the program director and manager for System z strategy and technology for z/OS and the strategy manager for the zSeries on demand initiative. He is the technical chairman of the System z e-Business Leadership Council and the System z System Design Council.

Jeffrey P. Kubala *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (kubes@us.ibm.com)*. Mr. Kubala received a B.S.E. degree in computer engineering from the University of Connecticut. He is an IBM Distinguished Engineer and Master Inventor in the z/OS development department. He is currently the technical team leader for the System z LPAR hypervisor. Mr. Kubala, in addition, is actively engaged with the iSeries* and pSeries* hypervisor teams as a technical consultant.

Robert J. (Bob) Maddison *IBM Software Group, Hursley Park, Winchester, England SO21 2JN (bob_maddison@uk.ibm.com)*. Mr. Maddison was involved in the development of Java virtual machines specializing in z/OS-specific issues, including the design and implementation of JVM support for zAAPs. He currently works with the recently formed common infrastructure development team in the field of information management.

Bernard R. Pierce *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (brpierce@us.ibm.com)*. Mr. Pierce has been a leader and innovator in performance analysis and design for thirty years. He is a Senior Technical Staff Member with responsibility for performance analysis and performance design of the z/OS operating system. He has been instrumental in the z/OS design to support zAAPs and zIIPs as well as research in the management of high-end SMP systems. Mr. Pierce is the inventor or coinventor of five patents relating to z/OS resource management.

Robert R. Rogers *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (rrrogers@us.ibm.com)*. Mr. Rogers received a B.A. degree in mathematics from Marist College. He is an IBM Distinguished Engineer working on System z software system design. He has received several IBM Outstanding Technical Achievement and IBM Outstanding Innovation Awards for his design work. Mr. Rogers was a recipient of the 2000 IBM Chairman's Award.