A

Dissertation

On

Reducing the Latency of Interaction with the Power Servers

Submitted in partial Fulfillment of the requirement

For the award of the Degree of

Master of Technology

In

Computer Science & Engineering

Submitted By

Rishi Mathur

University Roll No. 2K11/CSE/11

Under the esteemed guidance of

Dr. Daya Gupta

Head of the Computer Engineering Department, DTU, Delhi

DELHI TECHNOLOGICAL UNIVERSITY

DELHI TECHNOLOGICAL UNIVERSITY

CERTIFICATE

This is to certify that the work contained in this dissertation entitled "REDUCING THE LATENCY OF INTERACTION WITH THE POWER SERVERS" submitted in the partial fulfillment, for the award for the degree of M.Tech in Computer Science and Engineering at DELHI TECHNOLOGICAL UNIVERSITY by Rishi Mathur, Roll No. 2k11/CSE/11, is carried out by him under my supervision. This matter embodied in this project work has not been submitted earlier for the award of any degree or diploma in any university/institution to the best of our knowledge and belief.

Date:

Dr.DAYA GUPTA

PROJECT GUIDE and HOD, Computer Science

Dept. of Computer Engineering Dept. of Computer Engineering

ACKNOWLEDGEMENT

I take this opportunity to express a deep sense of gratitude towards my guideDr. Daya Gupta and my mentor at IBM, India Mr. HariganeshMuralidharan(HMC Development, IBM) and Ms. RashmiNarasimhan (Architect-Service Processor Development, IBM), for providing excellent guidance, encouragement and inspiration throughout the project work. Without invaluable guidance of them, this work would never have been a successful one.

I would like to thank Mr. Ajay K. Mahajan (Power Firmware Architecture and Development, IBM) for guiding me throughout my tenure as an intern at IBM. I would like to thank Ms. KiranmaiKuchibhotla (Manager – Power HMC Systems Director Integration, IBM) for providing me a conducive atmosphere at IBM. I would also like to thanks Mr. Ratan K. Gupta (Service Processor Development, IBM) to guide me and to resolve any queries or doubts I had.

Rishi Mathur

College Roll no: 2K11/CSE/11

M.Tech (Computer Science and Engineering)

Department of Computer Engineering

Delhi Technological University

Delhi-11004

ABSTRACT

Virtualization is one of the most important aspects of server industry today as it helps to mask the limited amount of physical resources available. It also forms the basis of the trending concept of cloud computing which speaks volumes of its significance. The Power servers of IBM follow the classical architecture of Virtualization where the virtual machine resides in the form of Logical Partitions (LPARs). Each *active* logical partition has its own operating system running along with virtual device drivers to interact with virtual devices. But for maintaining this virtual environment without much overhead of indirection, a component called hypervisor provides many features. By the virtue of these features, it is capable of virtualizing almost each aspect .The main features of virtualization include – Virtualization of processors, Active Memory Sharing (Virtualization of Memory) and virtualization of input/output operations. The creation and management of these logical partitions is done using a Hardware Management Console (HMC).Through HMC we can create or dynamically change profiles of Logical Partitions where a profile denotes the amount of resources required and the capabilities of the logical partitions. The management process is internally done by issuing of commands from HMC. But these commands are not directly headed to the hypervisor. Instead, the commands are headed to the Flexible Service Processor (FSP) which acts as a processor of the power server and is responsible for booting up and maintenance of the server. In the existing architecture of Power Servers, there is only single channel of communication available and each command whether it is targeted to or not, is handled by FSP only. This puts a significant amount of load on FSP. IBM proposes a modification in the architecture in the process of off-loading FSP, whereby a new channel of communication is introduced and the terminating point of this channel is not FSP, but a special LPAR. This new channel not only requires the designing of a special

LPAR but also modifications on the HMC side. In our work, we studied the existing communication architecture with the prime focus on understanding of Hardware Management Console and its modular functionalities. We then developed a mechanism on the HMC side to interact with the special LPAR and modified the code base in a manner such that, on the HMC side only a decision is taken on which communication channel a command should use based on its target. A significant amount of improvement was observed on a carefully put experimental setup which was inline with the fact that two channels of communication allow simultaneous issuing of commands.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

The concept of virtualization dates back to 1960s when expensive mainframe hardware and general computing was the domain of large. At this point of time, the virtualization became popular as it provided a software abstraction layer for the underlying hardware resources. But with the emergence of multitasking operating systems and reduction in the cost of hardware resources in 1990s, the concept of virtualization started fading. But as the limitations of hardware and operating systems came into realization, the focus was shifted back to virtualization and it underpinned the basis of companies like VMware Inc. As it turns out, in the new millennium virtualization holds on a firm grip on current trends of market and it forms the underlying base for fast emerging technologies like Cloud Computing [1].

The process of virtualization basically involves the creation of Logical Partitions (LPAR) or Virtual machines (VM). (We will be using the terms LPAR and Virtual Machines interchangeably in the further sections).Virtual Machines are used in disciplines that range from operating systems to processor architecture. The creation of virtual machines which involves the virtualization of resources like processor (Micropartitioning), memory (Active Memory Sharing) and input/output devices liberates developers from traditional usage and resource constraints and thus enhances system impregnability, software interoperability and platform versatility [2].

IBM was one of the first companies to work on the development of Virtualization features and provided two kinds of server platforms – *System i* and *System p* up till the early 1990s targeted for different market segments. Later it consolidated its features of processor, server and software onto both the server platforms and eventually came up with *Powerhypervisor* that provided a common virtualization platform for both flavors of IBM servers and in 2008, System i and System p were unified into a single POWER System Servers [3].

Specifically, a management console called *Hardware Management Console(HMC)* is what a client uses to interact with the Power servers and in this process of interaction , all the commands issued pass through a single communication channel to a component called *Flexible Service Processor (FSP)* and this component based on the *target* field of the command takes a decision on whether to forward this command to the next component or not, since some of the commands are intended for the flexible service processor only.

The whole process of virtualization is in the hands of a component called *hypervisor (PHYP or Virtual Machine Monitor).* So, the Flexible Service Processor forwards the commands that are meant for creation and management of LPARs to hypervisor and hypervisor takes the appropriate action on the targeted virtual machine. The commands that are forwarded to the hypervisor are the focus of our attention. We try to reduce the effective amount of time it takes to interact with the server and thus reduce the workload of Flexible Service Processor by introducing a special Logical Partition. The special Logical partition will now interact with the target Logical Partition and in turn will reduce the effective latency of the interaction with servers.

1.1 Motivation

The motivation for this work comes from the realization of the growing importance of s virtualization technology which in turn also forms the basis of a trending trade of current market, cloud computing. In the process of striving for efficiency IBM took up the issue of interaction between clients and their proprietaryPower servers.

In the existing Power Server systems, there is only a single channel of communication present which puts a significant amount of load on a component called Flexible Service Processor (FSP) as it's the first interaction point on the server site.

So, keeping the two important issues (that is of off-loading FSP and a single channel of communication) in mind, IBM's STG Development team came up with an idea of developing a new channel of communication whose end point is not the FSP but a virtual machine itself.

1.2 Related Work

Virtualization has been widely studied over the years by the researchers and is till date a popular topic of discussion. Goldberg et al. [11] gave the criteria which a machine design should fulfill to be virtualizable and recursively virtualizable. The criteria included system hygiene, software simplicity and system performance.

The Intel's X-86 architecture did not initially support Virtualization completely as John Scott Robin found out [12] that seventeen of the then existing instructions did

not meet the virtualization requirement as they were sensitive and unprivileged. But later both Intel [7] and AMD [8] introduced hardware support for virtualization.
In the process of thriving for efficiency more hardware based solutions were introduced to tackle different kinds of problems. Yazou Dong et al. [5] implemented hardware accelerations like Pause Loop Exit (PLE), ExtendedPage Table (EPT), and Single Root I/O Virtualization(SR-IOV) on Xen Hypervisor and analyzed the performance. In the series of hardware based solutions,Wing-Chi Poon and Aloysius K. Mok took up the issue of recursive virtualization [6] and argued that pure software solution falls short in improving the performance , thus they proposed a simple hardware extension for X-86 architecture.

DulyawitPrangchumpol et al. [9] proposed a preliminary idea of improving the performance of server virtualization by using data mining techniques to study the level of user access in virtual machines with heterogeneous workloads.

The above mentioned methodologies talk about performance improvement taking server side into consideration. We in our work take client's side into the picture and devise a strategy which provides a new channel of communication to interact with the server. We also consider the state of the intermediate components of the server, as the continuously increasing demands and modifications in the firmware put a significant load on some components. The new channel of communication that is introduced has a newly designed end point and more than command can be sent simultaneously.

1.3 Problem Statement

The Server virtualization forms an important aspect of the IT industry and it is essential to thrive for efficiency in whatever form it comes in. In this project we try to reduce the latency of the interaction with the servers and also try to balance the load of intermediate component called Flexible Service Processor (FSP). Our contribution in the project was to understand the interface architecture of the Hardware Management Console and the functionality of the different modules developing appropriate functionalities such that a communication path is established between the introduced special logical partition and the HMC. Also, we modified the hardware server code in order to make it capable of making a decision on which channel of communication to use for the commands. Thus we define our problem statement as

"**To modify Hardware server module to support new communication channel thereby reducing the latency of interaction with the Power Servers**"

1.4 Scope of Work

The project work is applicable to the areas of the technology which include the creation and management of virtual machines. In this project, no new component is introduced or platform specific modifications are done, only a dedicated virtual machine is introduced to manage the interaction with the targeted Logical Partitions , thus this technique is useful for all systems that support the creation and management of virtual machines.

In our work,we went through the process of understanding the architecture of the Hardware Management Console and functionalities of individual modules of the IBM Power Servers. We developed and modified Hardware Server module of HMC to enable it establishing a connection with the newly introduced special logical partition and also take a decision on which channel to use for the communication based on the target of the command.

We thus contributed on following levels in the project.

1. Understanding the basic architecture of Power Servers as well as the new architecture proposed by the IBM STG Development team.
2. Modify the Hardware Server module on HMC to support a new channel of communication.
3. Making Hardware Server capable of taking a decision on which channel of communication to use based on the TARGET of the command.
4. Establishing an experimental setup for the result analysis and selecting the best possible source (Hardware Server Logs and CIM Server Logs) to measure the improvements.

There are two advantages of the newly introduced channel –

- It helps in reducing load on Flexible service processor as it will have to deal only with the commands intended for the maintenance and surveillance of the server.
- It also allows more than command to be simultaneously issued from the Hardware Management Console, thus reducing the latency of interaction with the server.

1.5 Organization of Thesis

The rest of the thesis is organized as follows –

Chapter 2 discusses the concepts of Virtualization in detail where benefits and limitations are also described. Along with this it describes the architecture of Power Servers and discusses the role of important modules.

Chapter 3 details the existing architecture and the communication path followed by the issued commands. It then discusses the proposed architecture and describes that how proposed architecture reduces the latency of interaction with servers.

Chapter 4 describes our contribution in the project. It describes the modules we worked upon and also discusses the reasons for the modifications performed.

Chapter 5 discusses the Experimental Setup that was used to analyze the efficiency of the proposed architecture along with the result analysis.

Chapter 6 concludes the thesis and also provides a discussion on the future work.

Chapter 2

LITERATURE REVIEW

2.1 Virtualization

The computer systems, despite their complexity continue to evolve and the reasons for this evolution are –

- Well defined interfaces [2]
- Separate levels of abstraction [2]

But the well defined interfaces also come with their limitations as components and subsystems which are specific to a design will not work with the ones designed for another. This confining lack of interoperability is solved by the concept of virtualization [2].

Virtualization at a given abstraction layer maps the resources and interfaces of a component or a system onto a underlying real system. As a result, the real system appears to be a different system which is in fact a virtual one [2]. Goldberg et al. called it the *simulation* of one machine on a real machine and called the simulated machine as a virtual machine [11].

The server computer machines that host applications need to perform complicated tasks and thus need to utilize its resources well. A traditional approach used by computer network administrators is to dedicate a server for each special task. But this approach does not utilize the modern processing capabilities well. Underutilizing the resources and processing capabilities would also result in wastage of power. The issues of underutilization and wastage of power are well addressed by the concept of server virtualization. In the process of server virtualization, using specially designed software, one physical server can be utilized to run multiple virtual machines with each virtual machine having its own operating system. Theoretically, virtualization of servers can utilize all of the processing capabilities of the server [17][15][20].

In the coming sections we discuss the benefits of server virtualization, the limitations of server virtualization as well as the number of ways in which server virtualization can be done.

2.1.1 Benefits of Virtualization

Virtualization provides users with many benefits. Some of them are financially motivated and some are for technical reasons [20]. Uhlig et al. [13] classify the virtualization usages into three categories as shown in the Figure 1: -

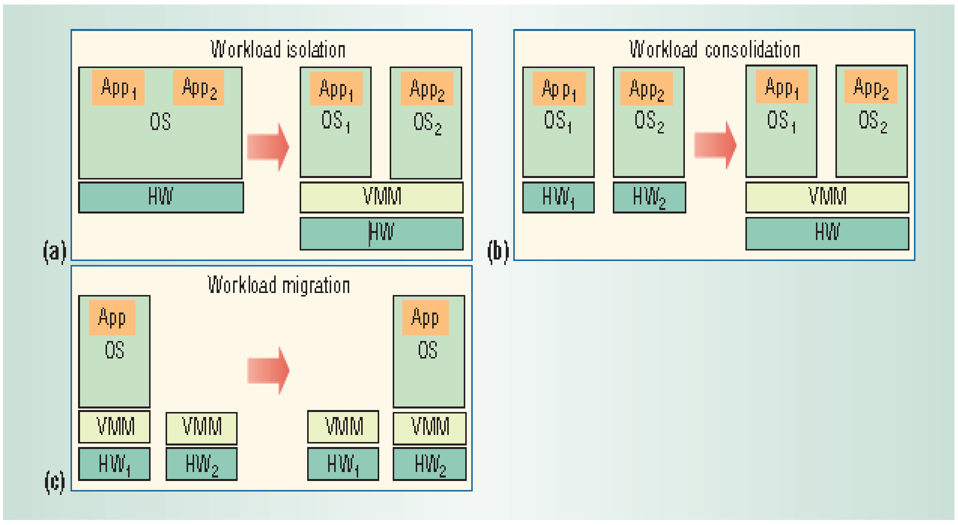


Figure 1 Virtualization Capabilities (a) Workload Isolation (b) Workload Consolidation (c) Workload Migration [13]

1. *Workload Isolation*

   Virtualization provides isolation of applications. The special mechanisms on the servers allow multiple applications to run simultaneously without interrupting the other applications [20]. The isolation ensures security and reliability for the virtual machines as intrusions would remain confined to that very virtual machine that is being attacked [13].

2. *Workload Consolidation*

   As mentioned before, the server virtualization serves to utilize the processing capabilities of the server perhaps in the best possible manner. So, if an application requires fewer amounts of resources, the resources can be allocated to the other applications [20].This reduces the total cost of ownership [13].

3. *Migration of Virtual Machines*

   The server virtualization provides a mechanism in which, the virtual machine can be shifted from one physical machine to another. The migration is achieved by encapsulating the state of the guest within the virtual machine and decoupling it from the hardware. The migration can also be automatically triggered by the load balancing agents and failure prediction agents. This feature reduces the total operation cost and also improves the quality of service [13].

   One of the many scenarios in which migration is required is the maintenance of physical servers [20].

2.1.2 Classification of Virtualization

There are three ways in which virtualization can be done.

1. Full Virtualization
2. Para – Virtualization
3. Operating System Level Virtualization

Full Virtualization

VMware was the first to introduce the concept of Full Virtualization [14]. In full virtualization unprivileged instructions execute normally while a binary translator is used to convert the privileged instructions into a block of unprivileged instructions. The translated block can now execute directly on the CPU [16].

Paravirtualization

The concept of paravirtualization was first introduced by Denali [18]. In paravirtualization, hypervisor aware operating systems are present. The operating systems use a hypervisor call known as *hcall*to perform logical operations like updating the virtual page tables [4]. The Paravirtualization proves to more efficient than full virtualization as it provides reduction in context-switching and parameter checking overload [20] [2]. But paravirtualization comes with the demerit that operating systems should be modified to be hypervisor aware [18].

Operating System Level Virtualization

The operating system level virtualization is completely different from the two types of virtualizations described above as it doesn’t use a hypervisor at all for the virtualization. The host operating system itself is responsible for providing this environment. But the biggest disadvantage of Operating System Level Virtualization is that all the virtual machines should be running the same operating system [20].

A logical partition is assigned with non-overlapping set of resources which includes virtual processors, regions of system memory and I/O adapter bus. In Power Systems for instance, the resource configuration information is communicated to the operating system by the platform firmware. Rest of the platform resources like memory controllers, interrupt controllers and almost all of the I/O infrastructures are controlled by the hypervisor. The operating system communicates with hypervisor using *hcall*to access these resources.

2.1.3 Limitations of Virtualization

Despite looking like a flawless concept, like most technologies server virtualization also comes with its limitations [20]. Some of those limitations are discussed below -

1. There are applications which require high processing power and performing virtualization on the physical machines that run these critical applications, can hamper the performance these critical applications .So, in such cases it's better to provide dedicated servers to these tasks.
2. The more the virtual machines created on the physical server, lesser is the amount of resources available to the virtual machines. So, running too many virtual machines on the same physical server is also not always a good idea.
3. The migration support in which one virtual machine can be migrated to another is possible between the physical machines running on the same processor.

2.2 Introduction to Power Servers

2.2.1 Overview

Power systems follow the classical server virtualization architecture in which hypervisor acts as the controller between the virtual machine requests and the physical available resources. The Power Servers allows the operating system to be hypervisor aware that helps to improve the utilization of the system resources as the operating system can now directly interact with the hypervisor. To give control to the hypervisor, there is a special instruction called 'hcall' which is a context switching instruction to give control to the hypervisor [4].

The Power implementation takes the approach of *paravirtualization* [4][16]. A system is paravirtualized if it has got a hypervisor aware version of operating system which utilizes 'hcall' to interact with hypervisor. A combination of hardware and firmware design can prevent the operating system to access the resources from other partitions or hypervisor. Paravirtualization acts a performance middle ground at the cost of relatively few Operating System changes [4].

The Power processor provides the mechanism to instantaneously reassign an idle processor to a LPAR that allows a LPAR to have more processors than there are actually (physically) present.

2.2.2 Architecture of the Power Platform

Power Server is a system assembly that is created by interconnecting various subsystems with each subsystem having its own resources [1]. The resources include memory, CPU cycles and I/O capability .A uniform access time is provided by an interconnect fabric which connects the subsystems. The three most important building blocks of the Power Server architecture are as shown in the FIG 2 and they are -

A) Hardware Management Console (HMC)

B) Flexible Service Processor (FSP)

C) Hypervisor Layer

Hardware Management Console forms the *Virtualization Control Point* for the users [4]. In other words, HMC is the console via which user can configure and control the virtual machines (LPAR).

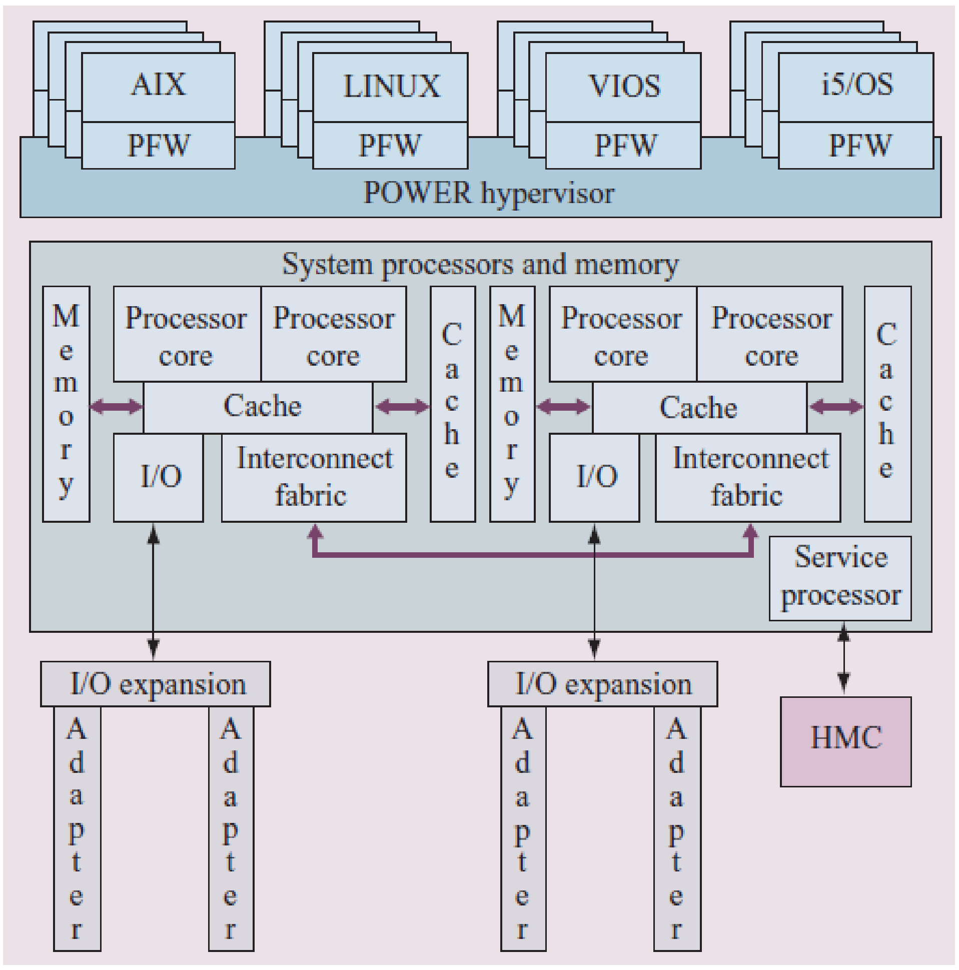


Figure 2 Architecture of Power Server [4]

The service processor is basically used for booting up and maintenance of the power servers [4]. It is the first interaction point for the user as the user should interact with the service processor in order to Power ON the system. The Hypervisor layer ensures integrity and isolation of the Logical partitions by validating the instructions issued by the operating system to utilize the resources [4]. The Hypervisor is the heart of virtualization and it is responsible for sharing of the available resources among the virtual machines. In the further sections, we will be discussing each of those 3 components in detail.

2.3 Hardware Management Console (HMC)

2.3.1 Introduction

HMC is a set of firmware tools that manages resources on the server via messages to the hypervisor and the operating system on the partition that is used for operational management of the platform [18]. The HMC is the control point for dynamic reconfiguration of resources on the partition, platform hardware operations and deferred and concurrent maintenance of both hardware and firmware [4]. The messages sent by HMC to hypervisor are actually commands and the path these commands follow depends on the OPCODE and TARGET specified in the command.

2.3.2 Basic Operations

The Hardware management console –

I. Creates initial configuration definition.

II. Controls boot and termination of partitions.

III. Provides Virtual Console Support.

2.3.3 Types of HMC

The HMC runs as an embedded OS on an Intel® based workstation that can be *Desktop or Rack Mounted* [18]**.** The embedded OS and applications take over the whole PC, and no other applications are allowed to be loaded. There are some models of HMC that were available for pSeries. These models can be upgraded to run the HMC code and manage System i5 systems. But they cannot manage both Power4 and System i5. The upgrade is simply a scratch install of the System i5 HMC code level.

2.3.4 HMC Implementation

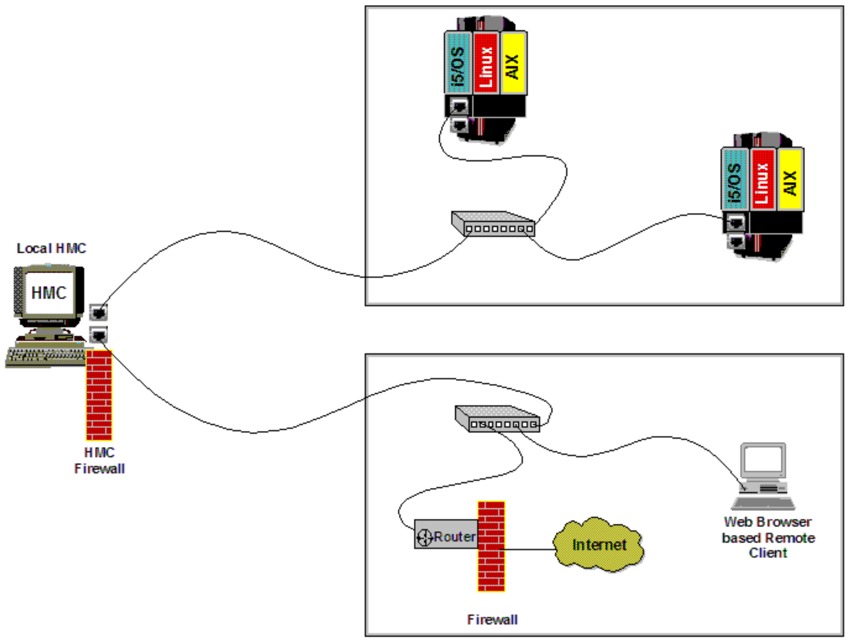


Figure 3 Implementation of HMC[18]

To provide flexibility and availability, HMC can be implemented as local and redundant HMC

A) Local HMC

A *local HMC* is one that is located physically close to the system that it manage and that is connected by either a private or public network [18]. An HMC in a *private network* is a DHCP server for the service processors of the systems it manages.

An HMC can also manage a system over a *public network* where the managed system's service processor IP address has been assigned manually using the Advanced System Management Interface (ASMI) or assigned by a DHCP server on the public network. For convenience of service personnel, an HMC is typically kept in close proximity to the servers that it manages [18].

B) Redundant HMC

A *redundant HMC* manages a system that is already managed by another HMC. When two HMCs manage one system, they are peers, and each can be used to control the managed system [18]. If both HMCs are connected to the server using private networks, each HMC must be a DHCP server set up to provide IP addresses on two unique, non-routing IP ranges. For best redundancy, redundant HMCs is kept on separate sub networks and attach to different server support network ports.

2.4 Flexible Service Processor (FSP)

Flexible Service Processor is not the main processor but a microprocessor within the system with its own hardware and firmware. FSP is responsible for booting up the system and providing various surveillance and monitoring functions.

The primary functions of FSP are –

1) POWER ON/OFF

FSP performs the booting operation for the system during which it loads the PHYP code from Flash into the system memory and starts the system's main processors. During the booting process, FSP also initializes various parameters that hold the state of the system.

2) Run time management

FSP recovers the system from recoverable errors and captures the failure state data in case of unrecoverable errors.

3) Updation of Firmware

The firmware update on the system is done via FSP.

4) Interfacing point

FSP forms the first interaction point on the server side for the commands issued form the HMC and in the existing architecture, all the commands issued from HMC pass through FSP. Since the management commands are to be passed to the Hypervisor (PHYP), FSP forms an interaction point with Hypervisor in the form of a mailbox. A similar mailbox is present on the Hypervisor’s side and the commands are passed through communication between these two mailboxes.

As discussed above, Flexible Service processor is the component that takes the responsibility of gathering all the commands that are issued from the Hardware Management Console. Besides that also, it has numerable tasks to perform. One of the primary aims of our work is to offload this component as we try to develop an environment in which not all the commands would be passing through the FSP. We will discuss this on detail in the coming chapters.

2.5 Hypervisor (PHYP)

A Hypervisor is the manager of the virtual machines running on a single system. The hypervisor allows multiple operating systems to shard the available physical resources among themselves. Each operating system thinks that it has a dedicated memory, processor and input/output adapters available [17][7].

But, instead they are sharing the available physical resources and the hypervisor manages these resources keeping a measure of the fact that one virtual machine does not disrupt with another.

2.5.1 Classification

The hypervisors can be classified into two major categories -

1. Bare Metal Hypervisors

Bare metal hypervisors run directly on the hardware to manage the virtual machines. No separate operating system is there for these hypervisors [17]. Thus these types of hypervisors work on the level just below the guest operating system and above the hardware.

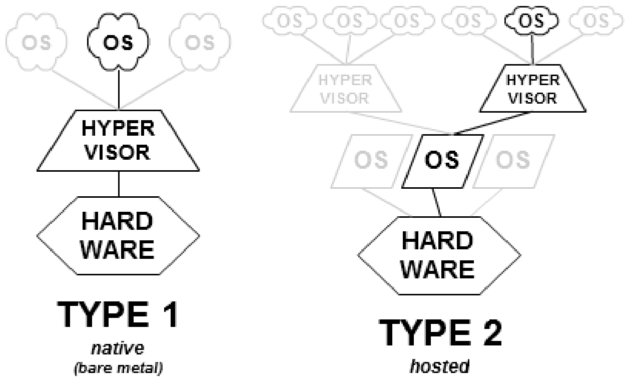


Figure 4 Types of Hypervisors [17]

2. Hosted Hypervisors

   The Hosted Hypervisors run on a separate operating system. So a hosted hypervisor works on the third level not directly above the hardware but above an operating system [17]. This is depicted in the figure shown above (FIG 3).

2.5.2 Power Hypervisor (PHYP)

Power hypervisor is the IBM version of the hypervisor that offers the functionality, the modern virtualization demands [19]. Power hypervisor forms the basis of virtualization by providing an environment via which the virtual machines (LPARS) can share resources among themselves.

Through a combination of hardware and software, Power Systems prevent hypervisor to access the resources of the other partitions. Without much hardware support, hypervisors must emulate hardware of the guest operating system, this will affect the performance. So, by the concept of paravirtualization hypervisor aware operating systems are designed that use a Memory Management Unit (MMU) system call *hcall* to interact with the hypervisor. The basic function of this system *hcall*is to perform the entire logical operation of virtual address translation [4].

But the concept of paravirtualization comes with reduced portability as guest operating system needs to be modified adequately. The Power architecture comes with 3 modes of privilege levels – User / Supervisor / Hypervisor mode. The guest operating system on the Logical Partition runs in supervisor mode, giving it access to the privileged instructions like system calls while the applications runs in user level mode. And finally there is a hypervisor mode which provides universal access to the physical resources [19][4].The hypervisor mode enables the access to the hypervisor to enable the access to the hardware resources by the guest operating system in hypervisor mediated fashion. So, when a guest operating system wants to perform a privileged operation such as modification of TLB (Translation Look Aside Buffer) , the processor is configured to trap this interrupt and direct it to the hypervisor which decides whether to allow / disallow or abort the request.

Hypervisor can also control the interrupts it wants to bypass. For example, if a guest operating system wants to access a dedicated I/O, it can allow the guest operating system to interact with the dedicated resource [4]. On the other hand, if the guest operating system on a shared LPAR wants to interact with a shared I/O, hypervisor must intercept the interrupt and take control and based on the availability of the requested resource, it takes the appropriate action. One of the major roles of PHYP is to manage and protect virtual machines. The hypervisor must prevent unauthorized access to a logical partition and also allows a Logical Partition to use its allocated amount of resources [19].

2.5.3 PHYP Architecture

PHYP’s architecture can be explained by dividing it to two main components

1. Platform Licensed Internal Code (PLIC)
2. Dispatchable PHYP

We now discuss each of these components

Platform Licensed Internal Code (PLIC)

PLIC is a non blocking interrupt driven layer which performs the time critical operations required for virtualization. PLIC is responsible for maintaining hardware page tables to translate an LPAR memory address into a physical address [7]. This prevents one LPAR to access the memory of any other logical partition. Similarly, PLIC maintains a Translation Control Entry (TCE) table and an I/O memory unit which is used to translate the address generated by I/O devices to physical memory assigned to the LPAR [7].

An ownership of physical I/O device must be assigned to a LPAR before a LPAR is permitted to map a portion of its memory to TCE entry associated with device [7].

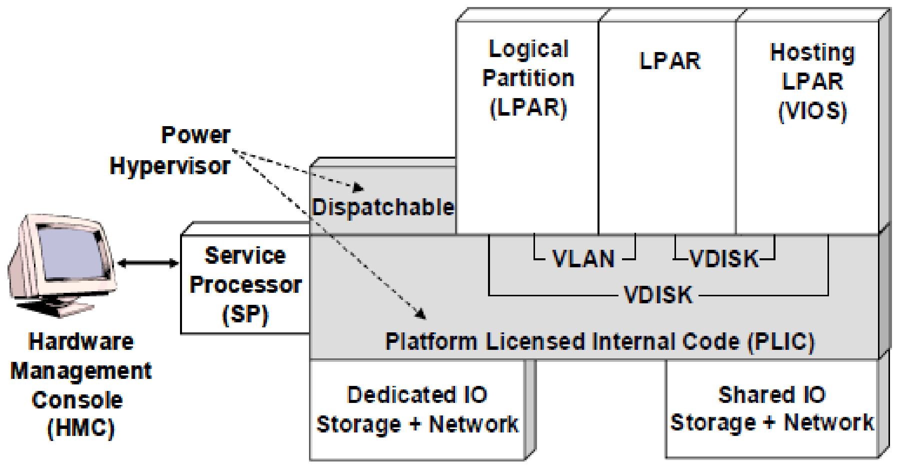

Logical Partition (LPAR)
LPAR
Hosting LPAR (VIOS)
Power Hypervisor
Dispatchable
Service Processor (SP)
VLAN
VDISK
VDISK
Platform Licensed Internal Code (PLIC)
Hardware Management Console (HMC)
Dedicated IO Storage + Network
Shared IO Storage + Network

Figure 5 : PHYP Architecture [19]

Dispatchable PHYP

Dispatchable PHYP executes as a hidden partition and provides the platform for non-critical services. The HMC is the one that provides configuration data of the LPAR to the Dispatchable PHYP [19]. So, Dispatchable PHYP is the one which is responsible for maintaining and processing the configuration data. Even in the operations related to the Dynamic Logical Partitions where the resources are allocated or deallocated for a LPAR dynamically, Dispatchable PHYP updates the configuration data.

Dispatchable PHYP is also involved in activities related to start up and termination of a logical partition. Besides that, it also mirrors processors which are used to maintain LPAR state [19].

The design of PHYP follows an overriding design principle in which PLIC and dispatchable PHYP are used to perform only those operations that cannot be performed by any other component of the software stack (which includes application, Operating System or VIOS) [19][4].

2.6 Virtualization of Memory (Active Memory Sharing)

2.6.1 Introduction to AMS

Memory Virtualization is an important feature of power systems. Power Servers implement the memory virtualization using Active Memory Sharing (AMS). So, AMS basically allows sharing of memory among the partitions and thus allowing many partitions to be a part of a shared memory pool. The Active Memory Sharing (AMS) not only allows more than one partition to be the part of shared memory pool but also allows the flow of memory among those partitions.AMS helps in -

A) Dynamically allocating memory to the partitions based on the workload demands

B) Improving the memory utilization or in other words allows over commitment.

C) Allowing the contents of the memory to be extended to the paging device.

The comparison of classical virtual memory and Active memory sharing virtual memory is depicted in the figures shown below.

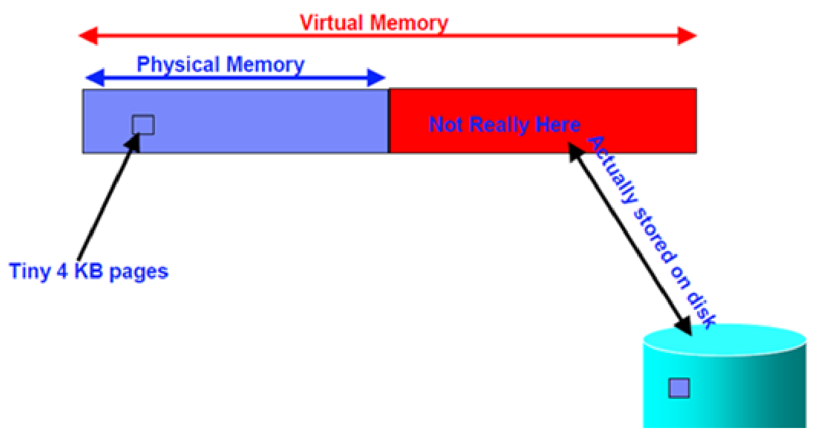


Figure 6Classical Virtual Memory

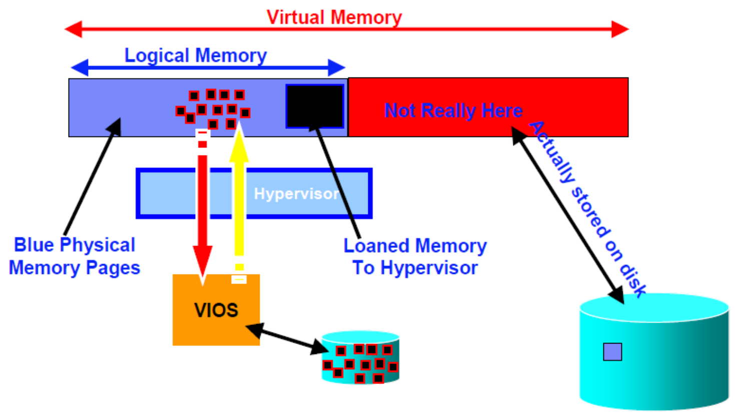


Figure 7 Active Memory Sharing Virtual Memory

In the classical virtual memory scenario, there is only one level of paging done via which the logical memory appears to be much more than the actual physical memory available. The mapping of the logical memory is directly done to the physical memory. While in the Active Memory Sharing scenario, two levels of paging are performed. Two levels of paging are required as Virtualization allows multiple virtual machines to run simultaneously [10]. Thus at the higher level, the operating system level paging is done. The operating system level paging is done by operating system mainly to perform page loaning while the paging at the Hypervisor level is done when page loaning is not possible and a page fault has occurred. In case of a page fault, the hypervisor pages out the contents of a page to the AMS paging device and utilizes this page.

2.6.2 Active Memory Sharing Components

1. Virtualization Control Point

Hardware Management Console (HMC) forms the virtualization control point for the power servers. Through this management console, the shared memory pool can be configured. Depending upon the number of partitions of the shared memory pool, the size of shared memory pool can be changed. The job of modification of the size of the pool is in the hands of system administrator. The system administrator determines the amount of physical memory to be assigned to the shared memory pool and this physical memory is assigned in the multiples of Logical Memory Blocks (LMB).Besides that, the system administrator can also select the number of paging devices for the shared memory pool [10].

The Management Console allows the user to select the desired and maximum shared memory pool size, the VIOS servers to be used for paging and the number of paging devices for shared memory pool.

2. Active Memory Sharing Manager (AMSM)

Active Memory Sharing Manager is a part of hypervisor responsible for managing the shared memory pool and managing the memory of partitions that are the part of shared memory pool. AMSM works on to keep pages needed by the workloads on the partitions inside the shared memory pool [10]. Some of the operating systems have a

Collaborative *Memory Manager* (CMM) as shown in figure helps AMSM to free up the partition pages upon request.

AMSM maintains a list of free physical pages. This list contains the free pages from partitions .When a page fault occurs for a workload on the partition; AMSM uses a free page to handle that page fault. This kind of handling keeps on happening until there is a low point hit. After the occurrence of this low point, AMSM starts taking pages from other partitions through *page stealing*. While doing *page stealing*, Hypervisor has to keep following factors into consideration

A) The Hypervisor perception of the relative memory loads.

B) The partitions shared memory weight.

AMSM provides a guaranteed mechanism that the contents of stolen pages are not reachable to any other partition by zeroing the contents of the page.

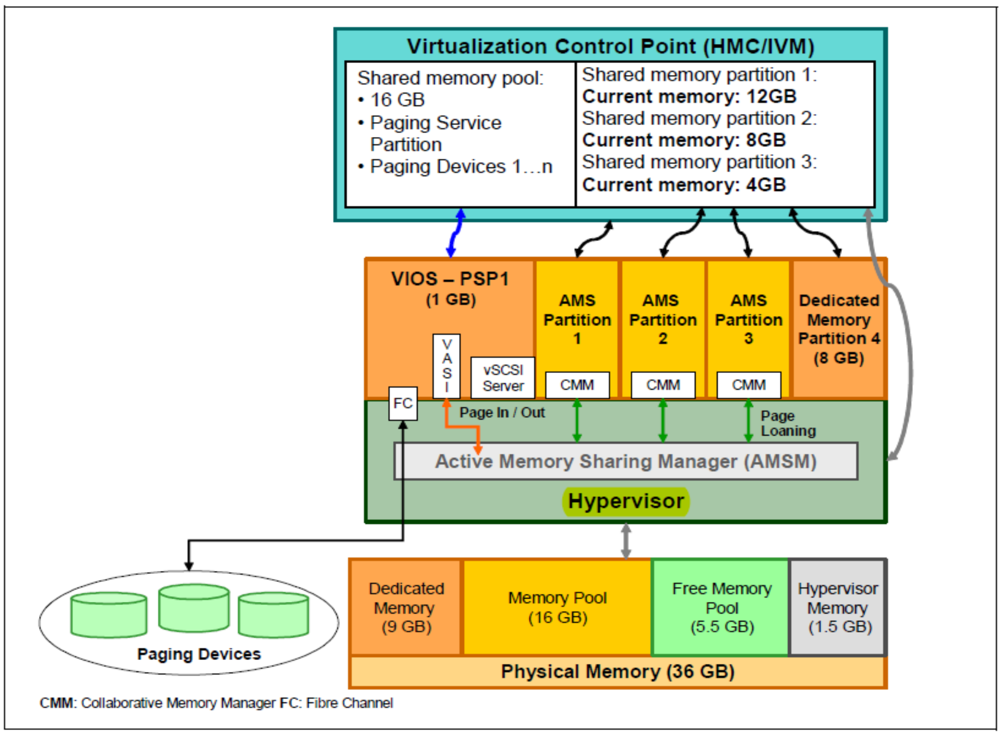

Virtualization Control Point (HMC/IVM)
Shared memory pool:
• 16 GB
• Paging Service Partition
• Paging Devices 1…n
Shared memory partition 1:
Current memory: 12GB
Shared memory partition 2:
Current memory: 8GB
Shared memory partition 3:
Current memory: 4GB
VIOS – PSP1 (1 GB)
VASI
vSCSI Server
FC
AMS Partition 1
AMS Partition 2
AMS Partition 3
Dedicated Memory Partition 4 (8 GB)
CMM
CMM
CMM
Page In / Out
Page Loaning
Active Memory Sharing Manager (AMSM)
Hypervisor
Paging Devices
Dedicated Memory (9 GB)
Memory Pool (16 GB)
Free Memory Pool (5.5 GB)
Hypervisor Memory (1.5 GB)
Physical Memory (36 GB)
CMM: Collaborative Memory Manager FC: Fibre Channel

Figure 8 Active Memory Sharing Architecture [10]

3. Paging VIOS

The purpose of paging VIOS is to allow the hypervisor to back up the excess pages onto the paging device. Single paging VIOS is associated with a shared memory pool but through the management console, a redundant paging VIOS can also provided to function in a redundant manner [10]. A VIOS can be dynamically enabled to work as a paging VIOS and can also be disabled when not in use as a paging VIOS through the management console.

4. PagingDevices

Before we discuss the role of paging devices in Active Memory Sharing Architecture, it’s important to discuss the concept of Page loaning as the major role of operating system paging is to perform Page Loaning which in turn will reduce the considerable amount of paging overhead [10].

Page Loaning

Page Loaning is the technique by which the operating system can loan special page frames called “Loaned Page Frames”.

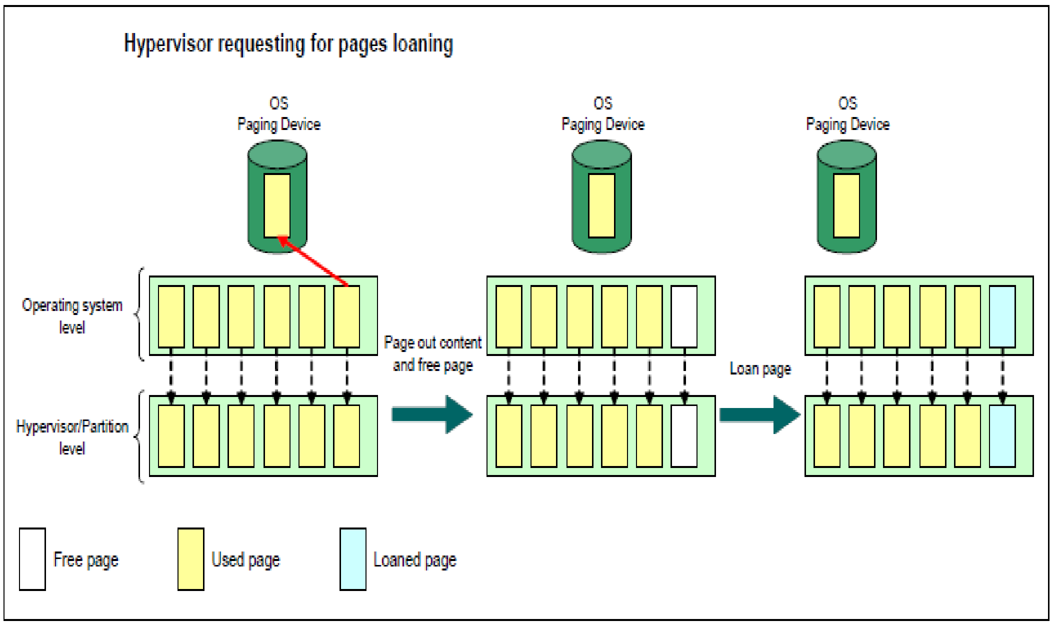


Figure 9 Page Loaning [10]

As shown in figure above, the operating system pages in the data of a loaned page to the paging device and frees one of the loaned page frames, so that the page frame can now be used by the hypervisor. When operating system cannot loan pages to the hypervisor, the hypervisor has to use its paging devices to get free pages for the partitions. There are two levels of paging done in Active Memory Sharing, since AMS provides a dynamic memory management among partitions. The two levels of paging are explained below –

A) Operating System Paging:

Operating System level paging is used in Active memory sharing to *loan* (Page Loaning) pages from hypervisor. Operating System Level paging is specific to an operating system .For instance, in IBMi the paging is not done on a single paging device but is spread across all around the disk storage network [10].

B) Hypervisor Paging Devices:

AMS ensures over commitment of memory to the shared partitions. There can be two scenarios based on the fact whether loaning is enabled or not. If we consider the scenario where *loaning (Page Loaning)* is disabled among partitions, then there would be considerable amount of paging [10]. Since, in this case the hypervisor would have to page in the contents of a page before it can allocate that page to another partition. In case where 'loaning' of pages is allowed, the paging overhead would be significantly decreased.

5. Virtual Asynchronous Service Interface(VASI)

VASI is a virtual device that allows paging VIOS to interact with the hypervisor [10].

6. I/O Entitled Memory

I/O entitled memory is the minimum amount of memory required by the Logical Partitions to perform I/O operations. The unavailability of physical memory during I/O operations can lead to failure of I/O operations. For example, network requires a minimum amount of physical memory for receiving packets. If the required amount of physical memory is not available, it may lead to latency or network time outs [10]. Similar is the case while doing a disk read operation as a minimum amount of amount of memory is required to hold the data that is to be read. So, for a partition a set of physical memory should remain unmoved during these set of operations.

6. Collaborative Memory Manager (CMM)

The AMSM does not know about the active and aged pages of each partition. The Hypervisor does not select the logical pages which are not a part of working set of partition [10]. The job of CMM residing in the operating System is to give hints to the hypervisor about what pages to select. This is needed when the system is overcommitted.

2.6.3 Active Memory Expansion

Active Memory Expansion (AME) is a feature that allows partitions to expand their memory by using in-memory data compression. Using Active Memory Sharing, the amount of logical memory is defined. An 'AME factor' can be applied on the logical amount to expand this memory. For example, as shown in figure two LPARs are shown in the figure. The amount of memory assigned to each LPAR is 4 GB and the AME factor is 2. Thus the logical memory of a LPAR is expandable up to 8 GB [10].

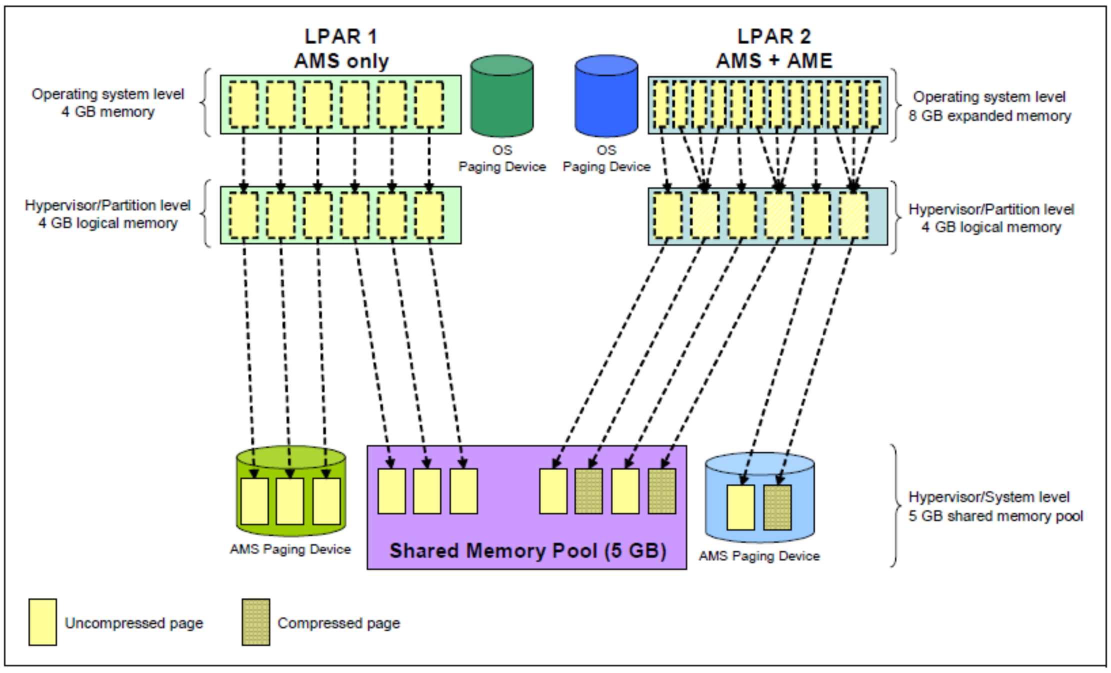


Figure 10 Active Memory Sharing [10]

The memory of 4 GB is provided by Active Memory Sharing which includes the physical memory on the shared memory pool plus the memory on paging devices. The extra 4 GB is provided through Active Memory Expansion by the virtue of which in memory data compression done at operating system level.

The basic difference between AMS and AME are -

A) Without AME, page loaning to the hypervisors could be done by paging out the contents of the page to be loaned to the paging devices.

B) With AME, page loaning would be done by first trying an in memory data compression. When compression is not possible, then only the contents of the page would be paged out.

This scenario is depicted in the figure shown below.

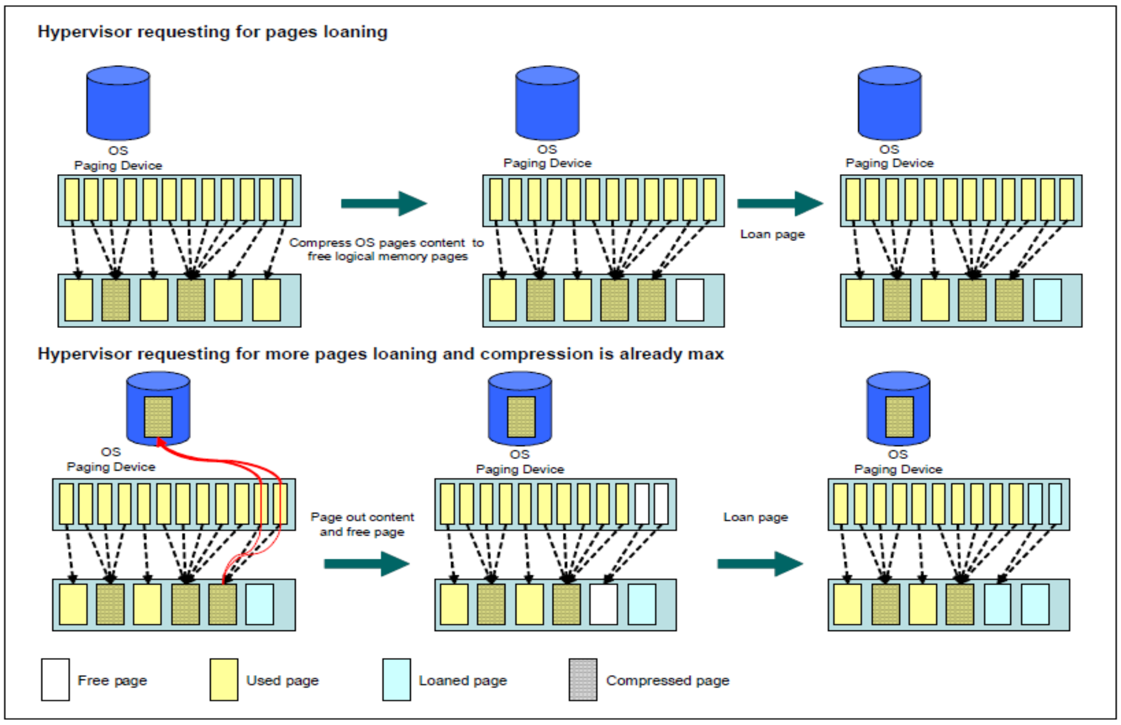

Hypervisor requesting for pages loaning
OS Paging Device
Compress OS pages content to free logical memory pages
OS Paging Device
Loan page
OS Paging Device
Hypervisor requesting for more pages loaning and compression is already max
OS Paging Device
Page out content and free page
OS Paging Device
Loan page
OS Paging Device
Free page
Used page
Loaned page
Compressed page

Figure 11 Page Loaning in Active Memory Sharing [10]

2.7 Virtualization of Processors (Micropartitioning)

2.7.1 Introduction

One of the most important features of Power virtualization is the virtualization of the processors. The virtualization of processors, also called *micropartitioning*is possible because of hypervisor as now a logical partition(if it is a part of shared processor pool) thinks, it has dedicated processor available with it but internally the processor cycles are being managed by the hypervisor.

The Virtualization of the Processors is depicted in the figure shown below –

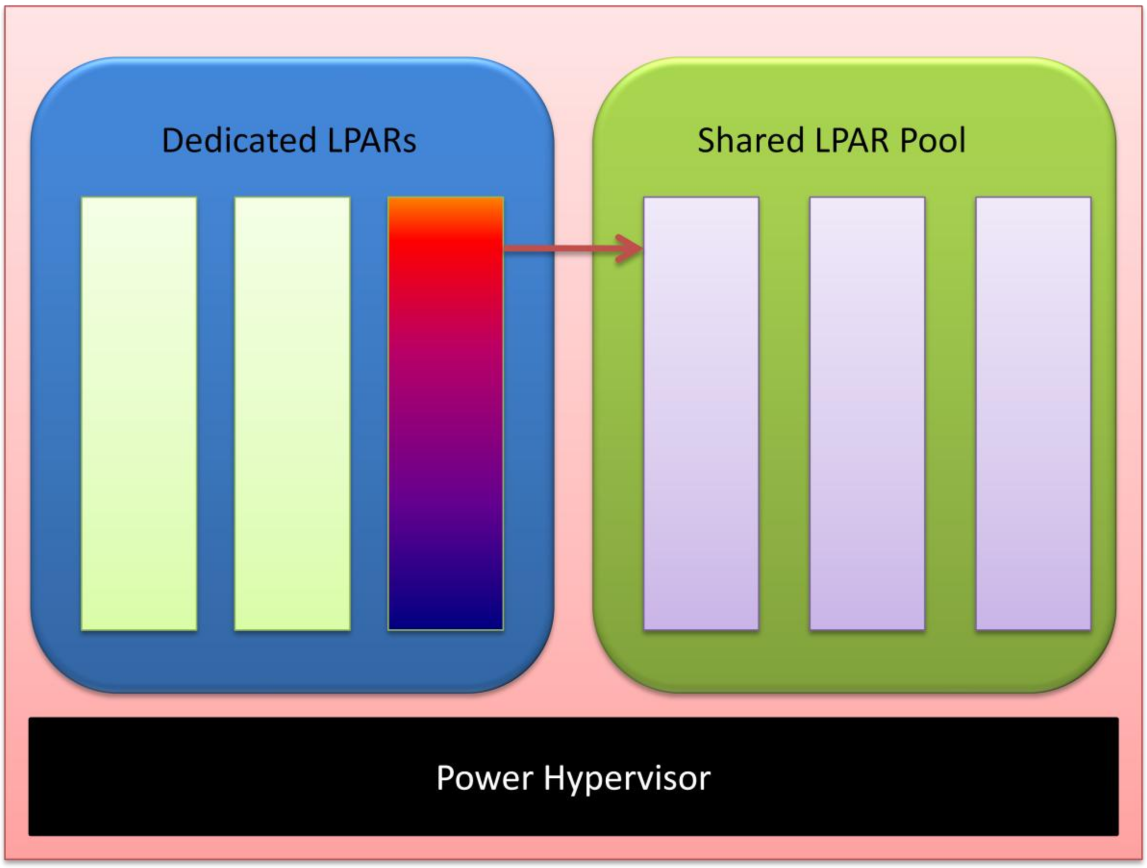


Figure 12 Dedicated and Shared Processor LPAR

The Logical Partition can be configured to have a dedicated processor resource (then it will be called as a shared partition) or it can be configured to have a shared processor resource in which case it will be a part of a shared processor pool. (Shared Logical Partition)

The shared logical partition can further be classified as *capped* or *uncapped* partition. A logical partition is called an uncapped partition if it can receive more processor cycles then its *entitled capacity* allows [4]. Here, the entitled capacity is the amount of physical processor resources allocated to the shared partition .Using the Hardware Management Console; we can define uncapped weight for a partition which is the priority share of the unused capacity of the processor relative to the other partitions sharing the same processor pool. Dynamic Logical Partitions (DLPAR) can have their entitled capacity and uncapped weight of a partition dynamically changed.

When a shared partition is configured, it is given with a definite amount of share of processor cycles. The logical partitions can be assigned with virtual CPU units that range from 0.1 to 64 CPUs. We can assign 0.1 CPUs because the unit of measurement for virtualization of processors is the number of processor cycles. Thus Virtualization of Processors implies the amount of processor cycles a logical partition is assigned with [4].

2.7.2 Methodology of Micropartitioning

The Hypervisor is equipped with special capabilities to ensure fine grain micropartitioning for the shared partitions. One of those special capabilities is in the form of *Hypervisor Decrementor (HDECR).* HDECR provides a guaranteed timer interrupt regardless of the partition execution state. HDECR interrupt is directly routed to the hypervisor and uses only hypervisor resources to capture the partition state [4].

Power servers also support Simultaneous Multithreading (SMT). Between SMT and shared processor, a special register called PURR (Performance Utilization Resource Register) is present to exactly count the number of processor cycles consumed by the partition thread on the physical processor. Also there is concept of *dormant thread*available[4]. A thread becomes dormant when it is not performing any job (idle), in that case the partition can invoke an *hcall*to let the other active threads to use the available processor cycles.

As an when an interrupt occurs to time out occurs for a dormant thread, the hardware reactivates the thread and its state is restored and control returns to the partition.

Similar to the operating system level optimization in Active Sharing Memory, operating system level optimization is achieved in virtualization of the processors. The operating system of a virtual machine registers an area called Virtual Processor Area (VPA) for each of the virtual processors assigned by the hypervisor [4]. The Virtual Processor Area consists of a field called *idle flag*. Whenever a virtual processor on the virtual machine becomes idle, the operating system sets the idle flag and similarly when there is work available for the virtual processor, the idle flag is reset again. In the idle state of a virtual processor, the operating system can invoke a special instruction called *h_cede_hcall* whichyields the physical processor to the hypervisor. After this operation, the virtual processor remains in the *blocking* state and the physical processor associated to it is dispatched to another virtual processor [4]. The blocking state of the virtual processor is dissolved when there is an interrupt or that blocked virtual processor is *prodded* by another virtual processor of the same partition by invoking another system call *h_prod_hcall*. Now, this call enforces the hypervisor to change the state of the virtual processor and this virtual processor is again available to the partition [4].

2.8 Virtualization of Input/output

The VIOS is another essential feature of Power VM virtualization. It is a LPAR with AIX operating system. It facilitates physical I/O resources between the LPARs. VIOS virtualizes physical storage and network adapters. At least two VIOS LPARS are there in every environment [21]. The (normal) LPARs have virtual Ethernet adapters (VNICs) and traffic passing through VNICs is passed to/from VIOS to/from physical adapters.

The software layer of Hypervisor (PHYP) provides a decoupling factor by the virtue of which a level of indirection is introduced between the abstract (logical) and physical. This decoupling is achieved via a virtual I/O server that resides on a Linux partition. The basic idea of this decoupling is the time and space multiplexing of the available resources [21]. There are number of benefits of using decoupling. Some of them are-

- It provides flexible mapping between physical and logical devices with assured portability.
- It provides dynamic migration of the virtual machines from one place to another.
- Decoupling enables suspension and resumption of the virtual machine by saving its state.

One of the main goals of I/O virtualization is achieving the above mentioned benefits with minimum overhead. Number of software and hardware approaches like paravirtualization and virtualization aware devices have been introduced to achieve high level of indirection [21]. The scheduling and prioritization of resource becomes important during the multiplexing the requests from different VMs onto limited number of physical resources.

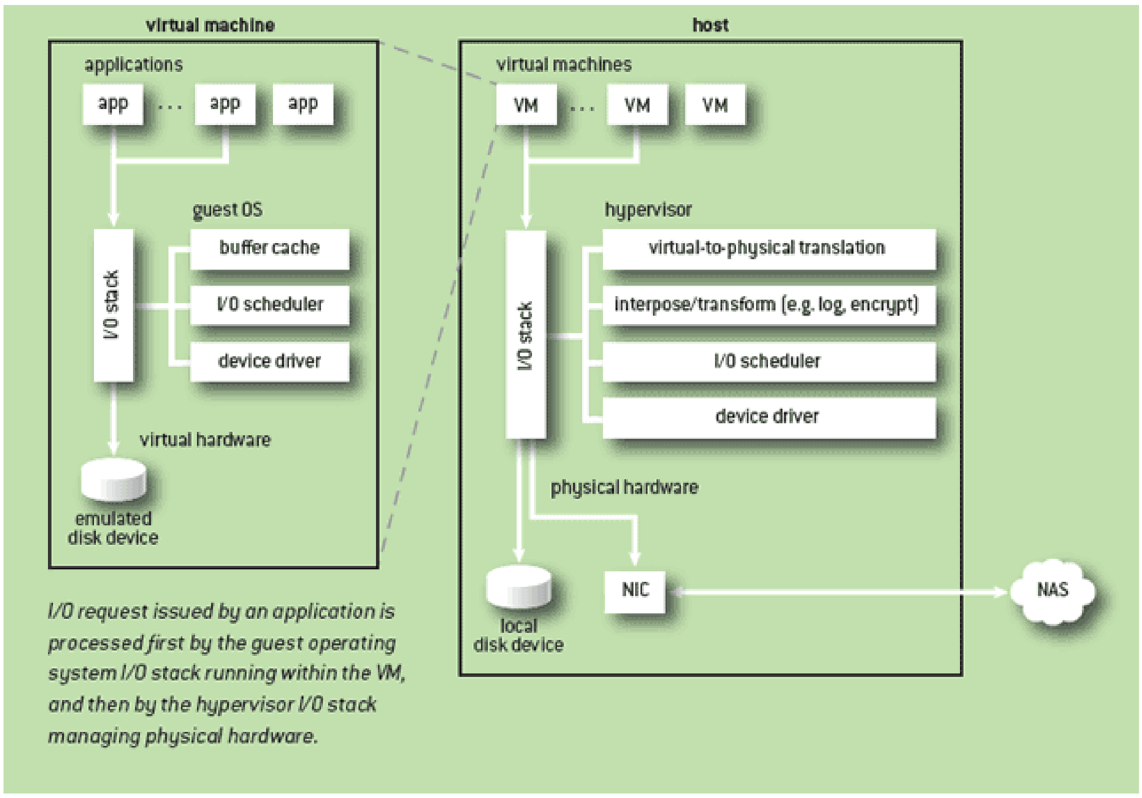

virtual machine
applications
app
app
app
guest OS
I/O stack
buffer cache
I/O scheduler
device driver
virtual hardware
emulated disk device
host
virtual machines
VM
VM
VM
hypervisor
I/O stack
virtual-to-physical translation
interpose/transform (e.g. log, encrypt)
I/O scheduler
device driver
physical hardware
local disk device
NIC
NAS
I/O request issued by an application is processed first by the guest operating system I/O stack running within the VM, and then by the hypervisor I/O stack managing physical hardware.

Figure 13 Processing an I/O request from Virtual Machine [21]

The figure shown above (FIG 13) explains the processing of an I/O request raised by a virtual machine.When an application running on virtual machine raises an I/O request (that is usually made by making a system call), it is initially processed by I/O stack of the guest operating system. After that via device driver present on the virtual machine, it tries to interact with a virtual device. Now this interaction request is intercepted by the Hypervisor which in turn schedules requests from multiple VMs on the physical device via another device driver managed by Hypervisor [21].

After that the two I/O stacks are again traversed in reverse order and finally the physical device generates a completion interrupt. This interrupt is intercepted by the Hypervisor .The Hypervisor intercepts again that to which VM the completion is associated with and generates a virtual interrupt and issues it to virtual device driver of VM.

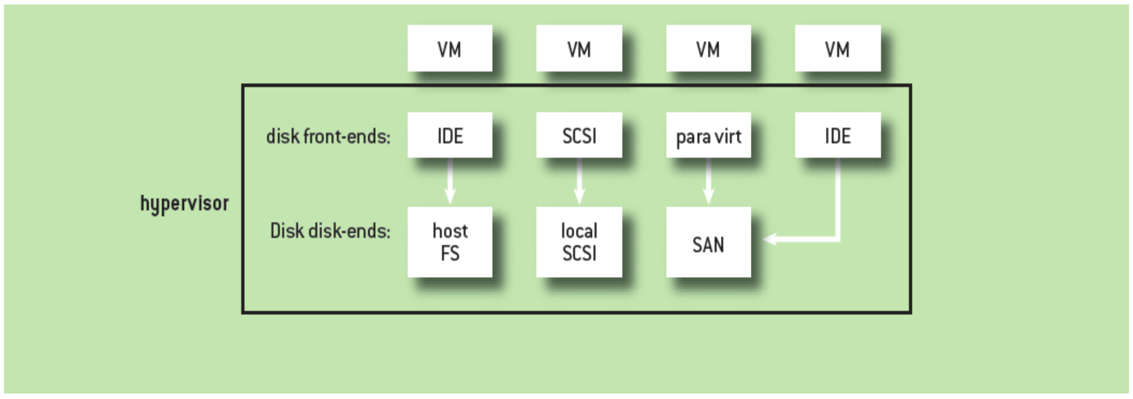


Figure 14 Modern Device Split Device Virtualization [21]

The modern hypervisors use a split implementation where a virtual machine can select among different virtual device interface emulation front-ends as well as multiple different back-end implementations of the device as shown in figure above.

While the device emulation code is specific to the particular device being emulated (e.g., an IDE disk), the semantics of the operations being performed are general and frequently constructed so that the same device emulation can access multiple different back-end implementations.

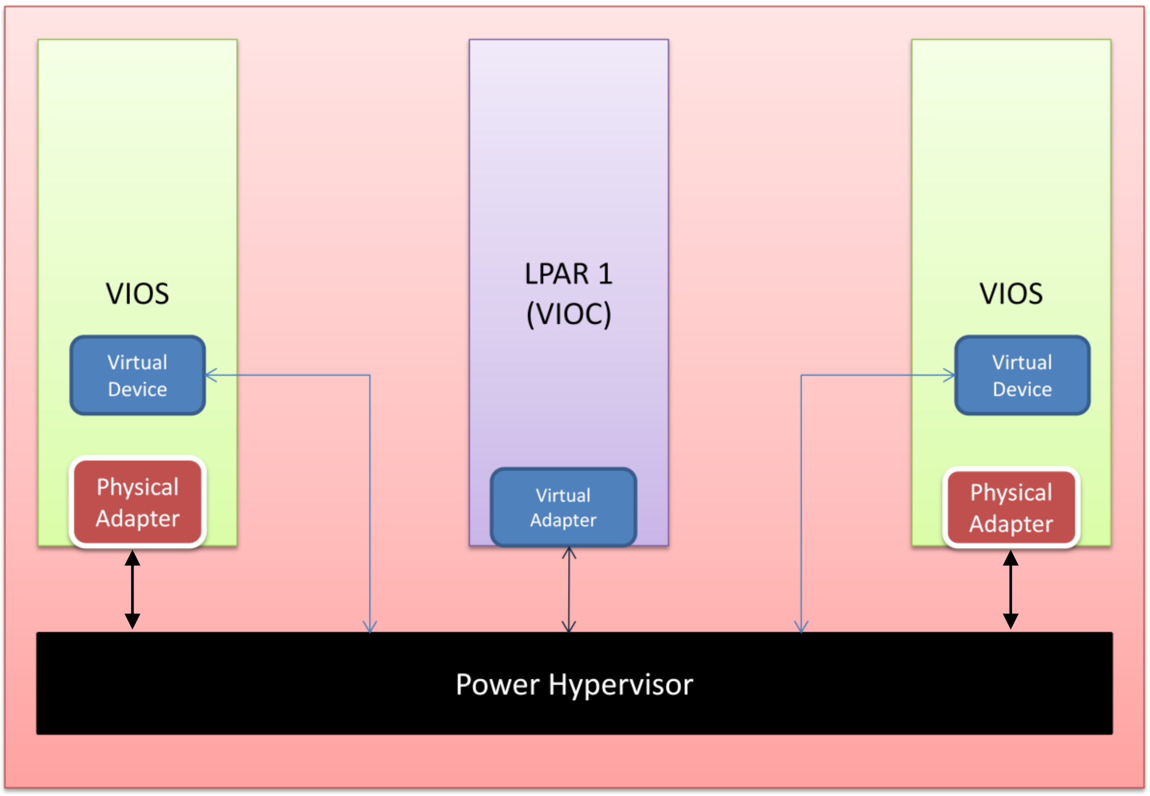


Figure 15 VIOS operation in Power Servers

If we talk specifically about Power Servers, the LPAR requesting a I/O operation is known as a VIO client LPAR and the LPAR containing the I/O server is known as VIOS(Virtual I/O server) .As explained already, the LPAR requesting I/O operation(VIOC) sends a request through a virtual adapter to a virtual device getting request from the virtual device driver operation.

The request for operation pertaining to the virtual device is intercepted by the hypervisor which in turn sends this request to the virtual device on the VIOS. Now VIOS performs the multiplexing of the different requests coming from different VMs onto limited number of Physical devices via physical adapters. As shown in figure, in most of the environments there are more than one Virtual I/O servers.

Chapter 3

PROPOSED ARCHITECTURE

In the previous chapter, we discussed the principle components of the Power Server along with their functionalities. In this chapter, first we will discuss the existing communication path that commands issued by the hardware management console follow. After introducing the existing architecture, we will discuss the proposed communication architecture and will discuss about how this communication path results in reduction in the interaction latency and also reduces the load on the Flexible Service Processor.

3.1 Existing Architecture

The client uses the Hardware Management Console (HMC) to issue the commands through a graphical user interface. The Hardware Management Console is itself a standalone system containing many components. The communication end point of Hardware Management Console is the *Hardware Server.* Every command issued by HMC, in the existing architecture is handled by the Flexible service processor (FSP) or in other words, the FSP acts as the first interaction point on the server site for the commands that are issued remotely from Hardware Management Console. The Existing architecture is depicted in the figure shown below (FIG 16).

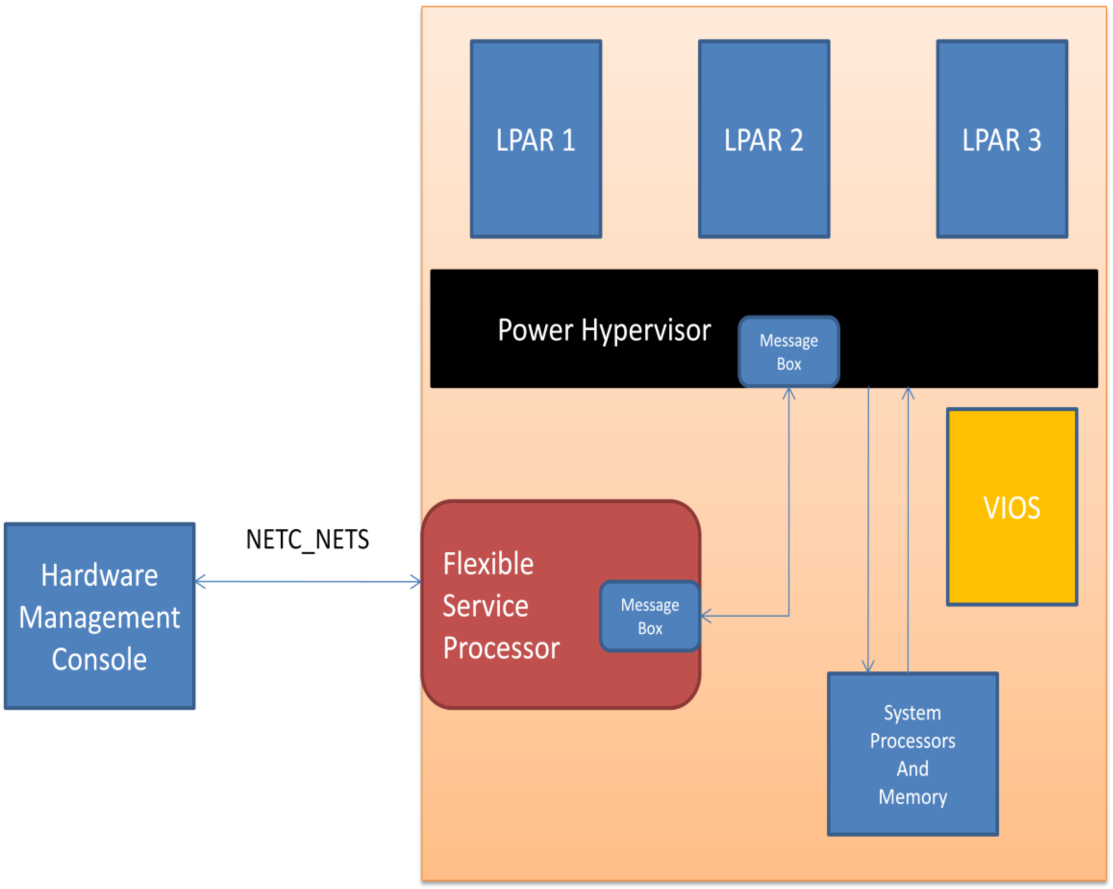

LPAR 1
LPAR 2
LPAR 3
Power Hypervisor
Message Box
VIOS
NETC_NETS
Hardware Management Console
Flexible Service Processor
Message Box
System Processors And Memory

Figure 16 Existing Architecture of Power Servers

The communication takes place on a secured socket layer (SSL). For any communication to take place between HMC and FSP, it is necessary that a SSL connection is established .Once the SSL connection is established, the messages in the form of packets can be transacted from one component to another.

There is a common message format for each of the message that is sent and received known *apriori*to both HMC and FSP. The most important constituents of this message format are *OPCODE* and *TARGET*. Although, the first interaction point on the server is Flexible Service Processor, not all commands are meant for it. Only those commands that are meant for booting up the server or maintenance of the server are intended for the flexible service processor .Rest of the commands is the point of our interest as these are the commands that are meant for creation and management of the virtual machines.

The flexible service processor forwards these commands to the hypervisor and hypervisor in turn takes the responsibility of forwarding these commands to the intended virtual machines. So, the obvious question arises , how does FSP knows about which commands to forward .The answer to this question is given by the *TARGET* field of the message format. Similarly, the function to be performed by a command is determined by the *OPCODE* field of the message format.

If the target of the command is a logical partition, then FSP sends this command to the hypervisor (PHYP) in a manner of storing the received messages in a message box and forwarding it to another message box and forwarding it to another message box present on the hypervisor side. The communication between these two message boxes takes place via IBM's proprietary component Interconnect link.

The hypervisor is now responsible for interacting with the operating systems on the virtual machines (in case the task is of managing the Virtual Machines). Also, the hypervisor is aware of the whole information about the available resources, so during the creation of a logical partition, it can assign the desired amount of resources to the virtual machine that is to be created.

In other words, the commands intended for creation and management of the virtual machines are to be forwarded to the hypervisor because hypervisor is aware of all the available and already resources. Now, it can be analyzed that the existing architecture puts significant amount of load on Flexible Service Processor as all of the commands now have to pass through FSP and then FSP has to decide on the intended target and forward the commands on to hypervisor.

3.2 New Architecture

IBM Software Labs introduced a new architecture in the process of thriving for efficiency of their proprietary Power Servers.In the proposed architecture, the focus is not only to try and reduce the load on the flexible service processor but also try to improve the performance by reducing the time it takes to interact with the servers. In the new architecture, a special logical partition is introduced which will now hold the responsibility of handling the commands intended for the hypervisor and the logical partitions. The new architecture is depicted in the figure shown below. As mentioned before, the hardware server forms the terminating communication point for HMC. So, instead of sending all the commands to the Flexible Service Processor, a decision would be taken on the hardware server itself based on the target and send it to the respective path. The special logical partition would be running an application to handle the commands that are being sent and this application is responsible for interacting with hypervisor and rest of the logical partitions.

One of the most important challenges in the new architecture is the communication between HMC and the newly introduced logical partition. The interaction from HMC to the newly introduced logical partition happens in the same way as the communication between HMC and FSP take place.

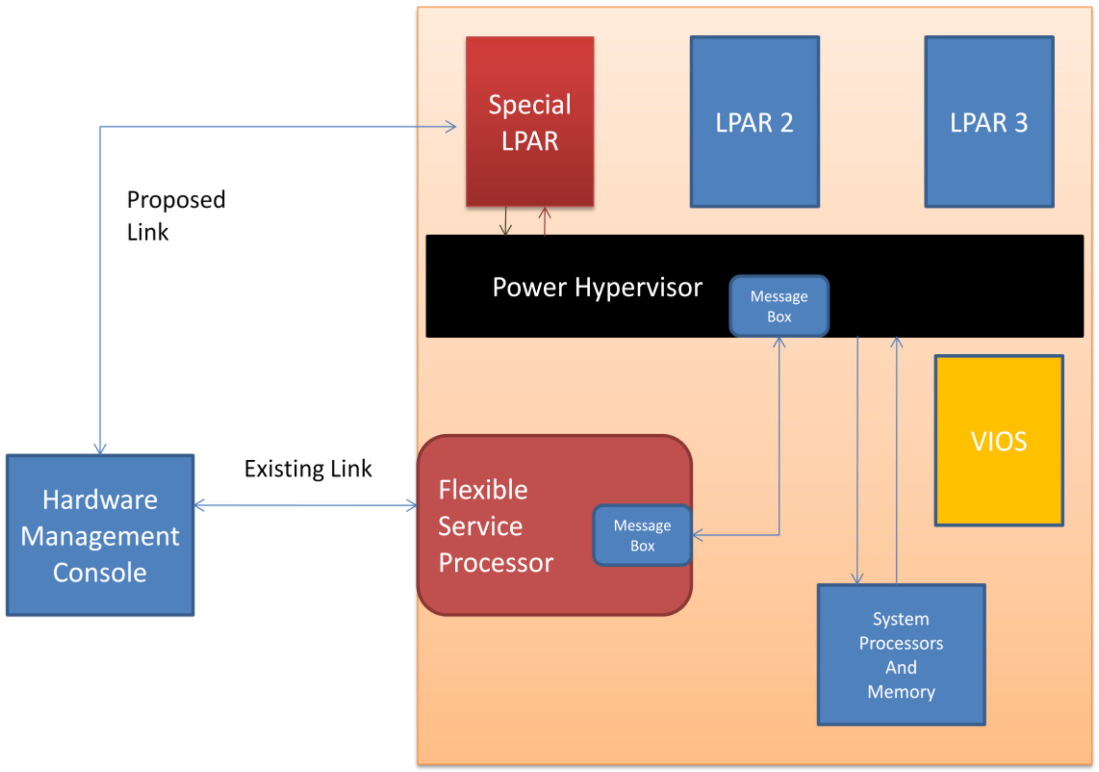


Figure 17New Architecture that support new communication channel

Similar to the existing architecture, a secured connection (SSL) is established between HMC and this new LPAR and the application running on this LPAR is designed to understand the message format of the packets that are being transacted.

The remote HMC communicates with FSP on a TCP/IP connection. Similarly, in a virtual environment the new active partition would also be identified by its IP address and similar set of formal commands would be sent from HMC to establish a connection on a secured layer. The Ethernet adapter to this logical partition can be dedicated one or it can be a shared one.

Now, in the next step a communication path is set between newly introduced LPAR and the targeted virtual machines (or the hypervisor). The communication process between the logical partitions happens through a Virtual LAN (VLAN) setup. In this mechanism, the logical partitions acts as nodes on a local area network and the message can be unicasted from one LPAR to another. Similarly, in our case the newly introduced LPAR would be communicating with the targeted LPARS. The newly introduced LPAR is an able communicator with PHYP also as the application running would be over the top of a hypervisor aware operating system. So, it's possible for this application to interact with the hypervisor. This is needed for instance, in case of creation of new virtual machines.

Since, there are two channels of communication now; both of the intended results are achieved. The commands intended solely for the FSP can be sent on the existing channel while the commands meant for creation and management of the Logical Partition would be sent via the proposed link as shown in the FIG 17.

Chapter 4

METHODOLOGY AND SYSTEM DESIGN

In the previous chapter, we discussed the architecture proposed by IBM .In this chapter we detail our contribution in the project which was to develop and modify hardware server to establish the connection with the special LPAR. We start with explaining the interface architecture of HMC along with the functions of each component and then discuss the algorithm and implementation details.

4.1 Interface Architecture of HMC

In this section, we first describe the functionality of each component in the existing interface architecture of HMC. We go on to describe the explain the new interface architecture of HMC along with the new functionalities

4.1.1 Basic Interface Architecture of HMC

The basic interface architecture of HMC with the Power Servers is shown below in Figure 18.As depicted in the diagram, HMC provides both GUI and CLI interfaces to issue commands. The core of HMC is formed by the CIM which describes the common information model for the managed resources. The managed resources include network system, applications, devices and communication equipment. The managed resources are represented in the form of objects and these objects are stored in a repository in the managed object format (.mof). CIM object management (CIMOM) includes

- CIM client
- CIM server
- CIM object manager

CIM client forms the management system where application logic is written to manage the resources while CIM server, CIM object manager along with the repository forms the part of the managed system.

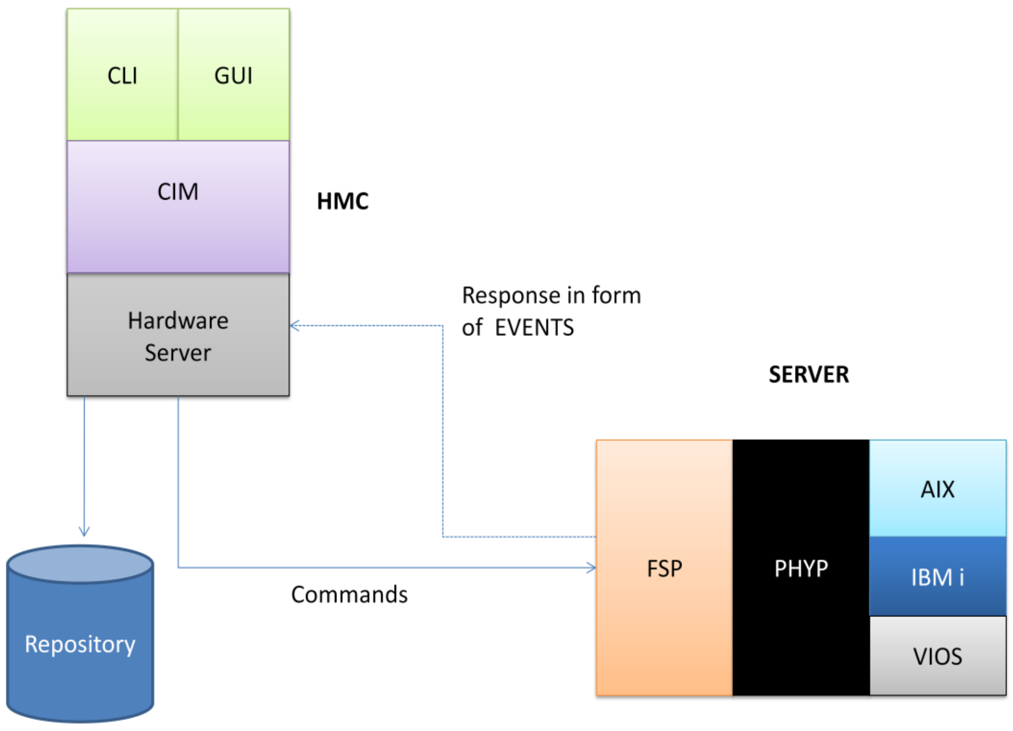


Figure 18HMC Interface Architecture

In the communication process, the CIM server which acts as a framework supporting virtualization performs initialization operations where functionalities and methods of a managed object are taken as properties of the object stored in the repository. In the communication path, the next component is the hardware server. Hardware server has many responsibilities which include –

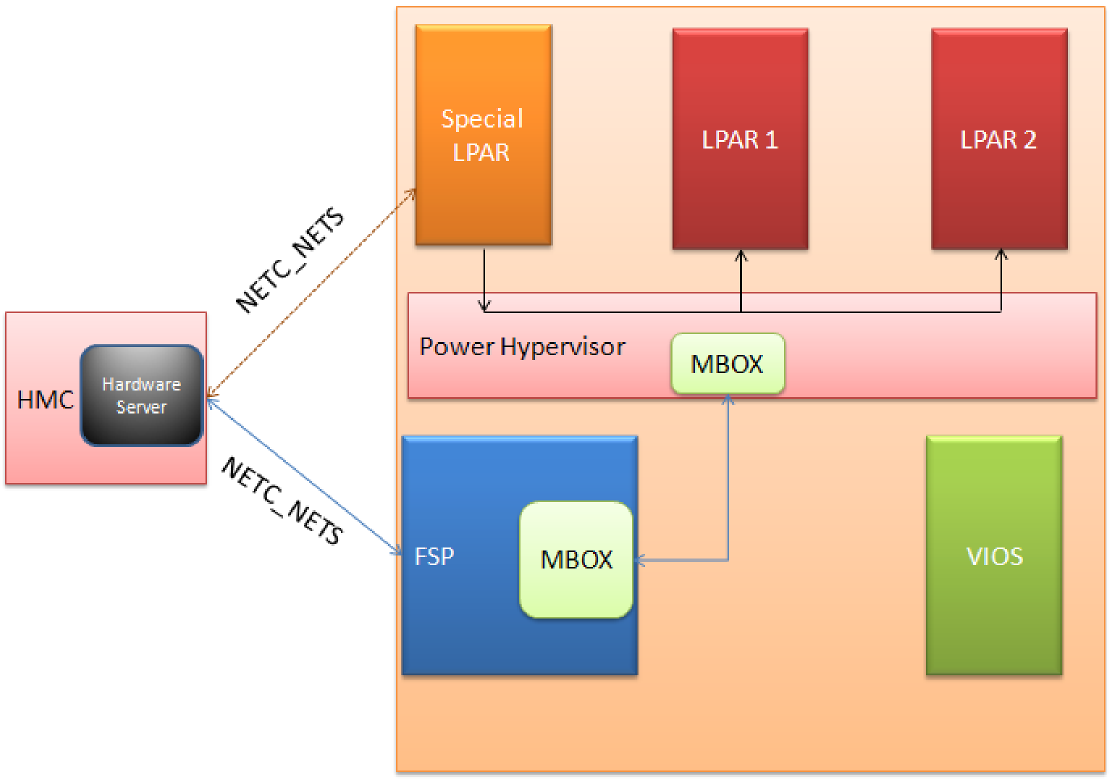
- Management of the connection between FSP and HMC
- Transferring the command packet from HMC to FSP
- Opening of PHYP connection
- Management of "HMCNetConfig" file. The HMCNetConfig file stores the configuration information of the servers that were connected previously from the corresponding HMC or it creates a new entry for the new connection

From the hardware server, every command is forwarded to FSP.

4.1.2 New Interface Architecture to support special logical partition

As mentioned above, hardware server holds the responsibility of managing the connection between HMC and FSP. Similarly, we added the functionality such that now, hardware server would be able to establish the connection between HMC and the special logical partition as shown in Figure 19. The establishment of connection however should be possible once the Power Server is booted on and the special LPAR is activated with the application to gather the commands running on it. So, we made the hardware server to be aware of the fact that the special logical partition is up and running with the application to gather the commands is running. As the server boots up, we created a mechanism such that the id of the special LPAR would be exchanged and it would updated in HMCNetConfig file. We modified the HMCNetConfig file such that once the entry is made; the HMC would automatically try to establish the whole connection process if a connection made is not a new one but an attempt has already been made for the connection to this server. This process includes the booting of the server and activation of the special logical partition. The connection between HMC and the special LPAR was established on the lines of NETC_NETS and same common message format was followed for the communication process.

The next task was to modify the hardware server code, so that on the client side, a decision would be taken on which channel to use for the communication. This decision is required to be taken because a new channel is now introduced. The existing channel which leads to the Flexible Service Processor are now primarily is the carrier of only those commands that are for maintenance of the server. While the proposed channel has the primary task of carrying the commands that are issued in order to create new virtual machines or to manage the already exist



.Figure 19 New Interface Architecture

We modified the hardware server code in such a way that it is now capable of taking a decision of choosing the channel of communication based on the TARGET of the command. The TARGET was the id of the special LPAR that was exchanged during the connection establishment process. In the existing architecture, the commands that meant for the creation and management of the virtual machines were forwarded to the hypervisor via FSP which put a significant amount of load on FSP, but since we can always differentiate between the two targets – PHYP and FSP. We made the hardware server enable to choose the communication path of the command.

4.2 Detailed Design to support special logical partition

In this section we present the detailed design and algorithm that has been implemented for –

(i) The Activation of the special logical partition

(ii) The connection establishment of HMC with the special logical partition.

(iii) The modifications in the hardware server module to take a decision based on the *target* of the command

Before proceeding with the above tasks there are two prerequisites to follow –

- A connection should be established between the HMC and the Power server (via FSP).
- The Power Server should be in the POWER ON state.

In the following phases we discusshow the HMC establishes the connection with the new logical partition along with the implementation details.

1. **Connection Establishment with the Server:** In this step, the user has to establish a connection to the server if it is not already connected as shown in Figure 20. The commands issued by the user are always sent to the CIM object manager where various parameters are initialized and validation of the commands is performed. After the validation process, through hardware server, commands are forwarded to

the FSP. On a secured connection(SSL), exchange of messages takes place over NETC_NETS protocol.

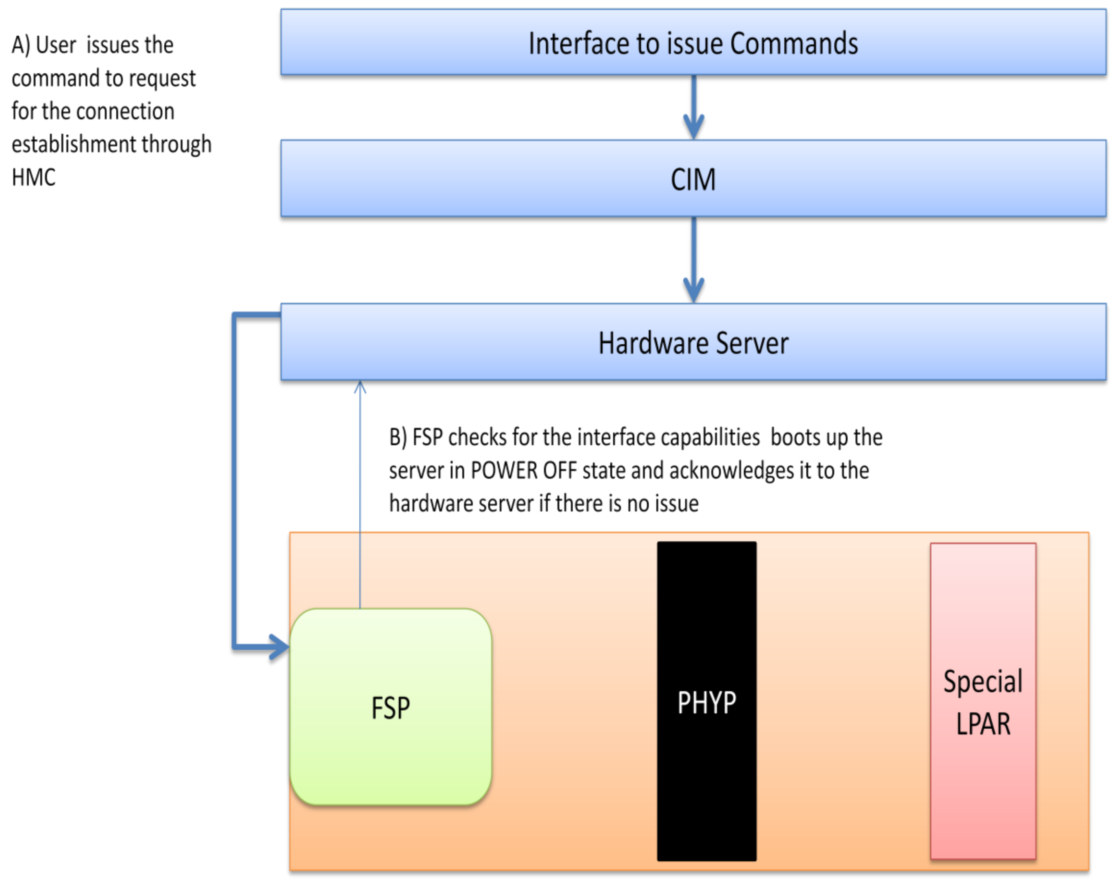

A) User issues the command to request for the connection establishment through HMC
Interface to issue Commands
CIM
Hardware Server
B) FSP checks for the interface capabilities boots up the server in POWER OFF state and acknowledges it to the hardware server if there is no issue
FSP
PHYP
Special LPAR

Figure 20 Connection Establishment between HMC and the Power Server

While connecting to a new server HMC checks for the interface capabilities of the server and then decides whether or not it is eligible to make a connection. If a connection is possible, HMC appends an entry into a file called “HMCNetConfig” which holds the information about all the connected servers like IP address, machine address and some parameter values. With modifications on the FSP site, we now had the

ID of the special LPAR also retrieved by the hardware server. We created a new entry for the ID of this special LPAR in the HMCNetConfig file.

2. **Changing the state to POWER ON:**In the next step, the user issues the command to POWER ON the server and this command is also handled by the FSP as shown in Figure 21. In this step FSP opens up the connection to the PHYP.

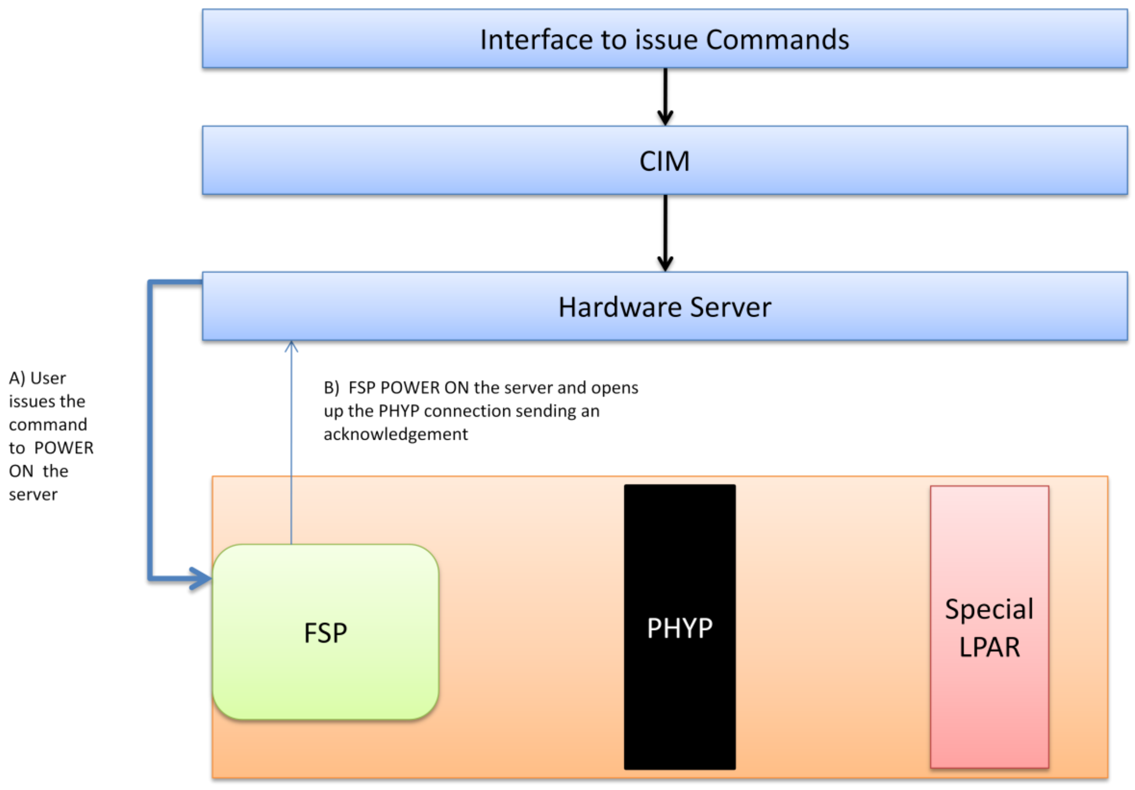

Interface to issue Commands
CIM
Hardware Server
A) User issues the command to POWER ON the server
B) FSP POWER ON the server and opens up the PHYP connection sending an acknowledgement
FSP
PHYP
Special LPAR

Figure 21 Change of state from Power OFF to Power ON

3. **Automation of the Activation process of special LPAR**: As Hardware server has the ID of the special logical partition, we automated a step in which an activation command for the special LPAR is issued just after PHYP channel is opened up as shown in Figure 22. For activation of a LPAR, ID of that LPAR is required and Hardware server already got the ID of the special LPAR in the previous step. The

activation process also involves the initiation of the application that runs as a server on the special LPAR. The task of managing and keeping the application running was handled by the PHYP.

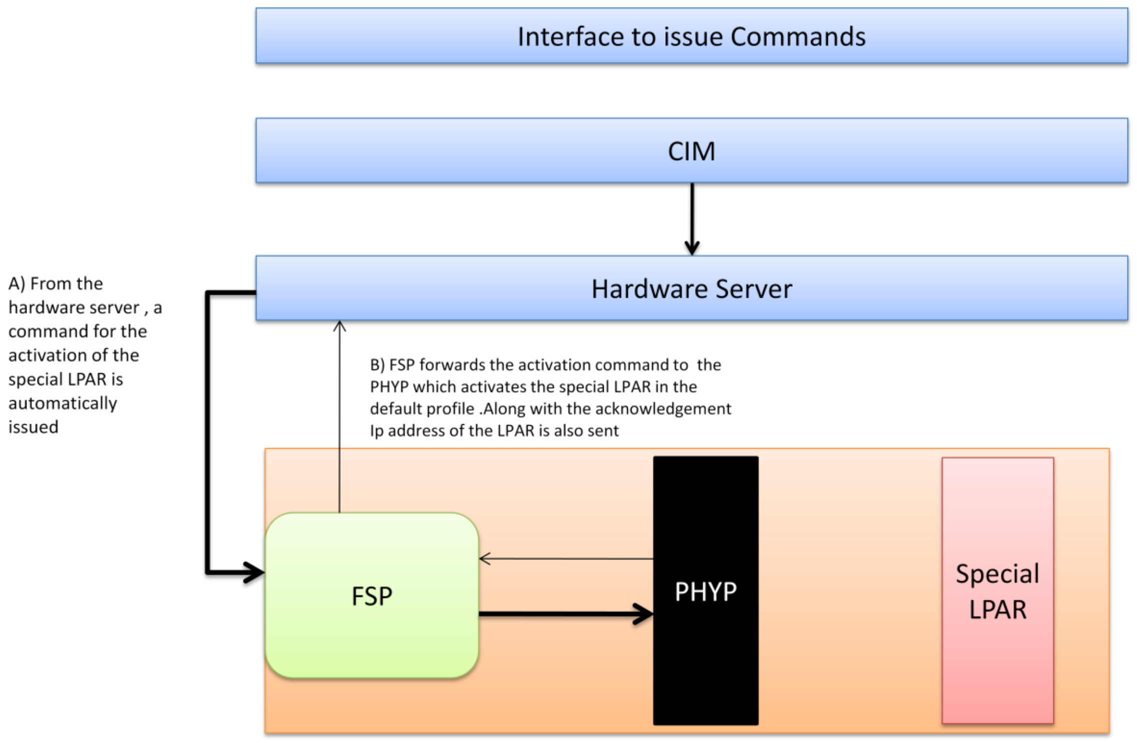

Interface to issue Commands
CIM
Hardware Server
A) From the hardware server , a command for the activation of the special LPAR is automatically issued
B) FSP forwards the activation command to the PHYP which activates the special LPAR in the default profile .Along with the acknowledgement Ip address of the LPAR is also sent
FSP
PHYP
Special LPAR

Figure 22 Activation of the Special Logical Partition

4. **Connection Establishment with special LPAR**: With activation of special LPAR and the application in the form of server running, HMC can now attempt a secured connection with the special LPAR as shown in Figure 23. We implemented a mechanism on the hardware server to automatically send a "make connection" set of commands to the special LPAR. Before issuing these commands are validated by CIM object manager and then sent to the special LPAR. The implementation wasdone on the lines of NETC_NETS keeping in mind that the connection was

being

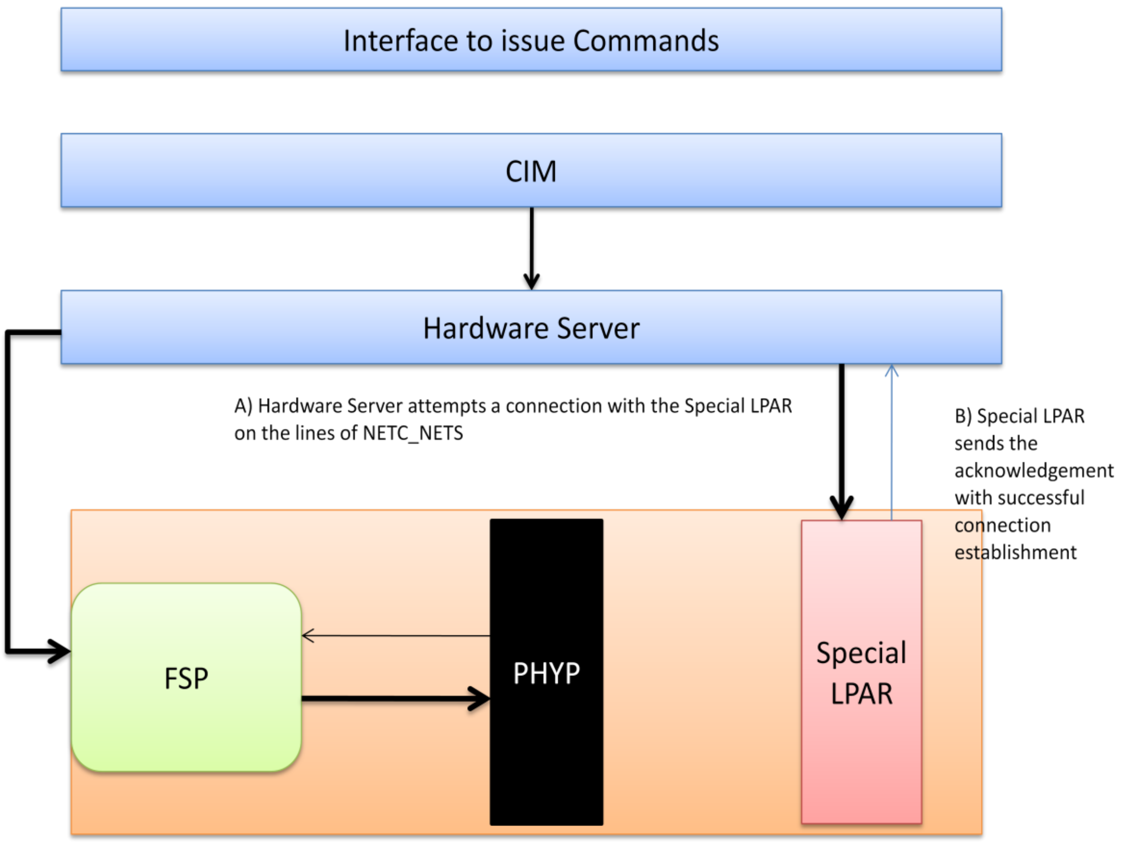

Interface to issue Commands
CIM
Hardware Server
A) Hardware Server attempts a connection with the Special LPAR on the lines of NETC_NETS
B) Special LPAR sends the acknowledgement with successful connection establishment
FSP
PHYP
Special LPAR

Figure 23 Connection Establishment between HMC and the special Logical Partition

5. **Channel Selection:**We now have two channels of communication available. We modified the code on the hardware server in a way that it could now take a decision on which channel of communication to be used. On the hardware server, we created a new target value that was based on the ID of the special LPAR. Hence Hardware Server could dynamically a channel of communication for a command as shown in Figure 24.

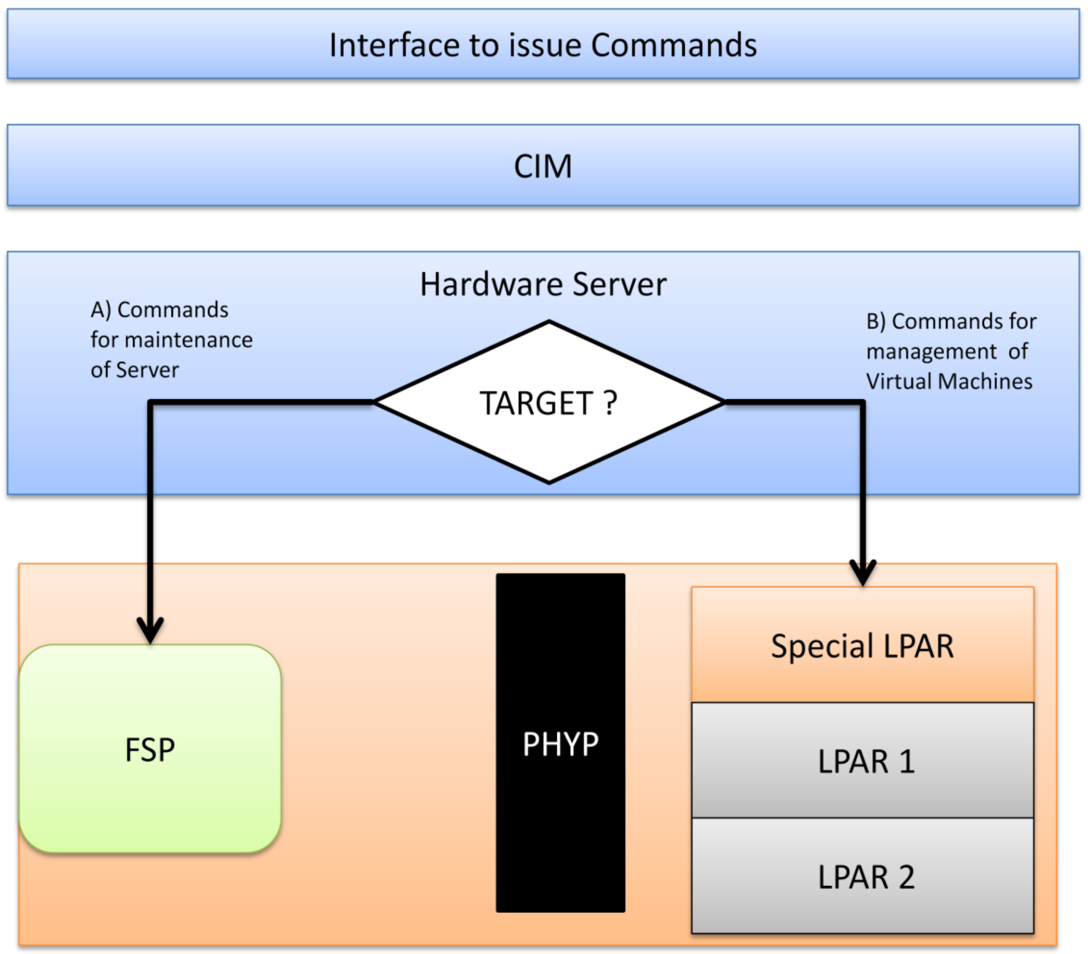

Interface to issue Commands
CIM
Hardware Server
A) Commands for maintenance of Server
TARGET ?
B) Commands for management of Virtual Machines
FSP
PHYP
Special LPAR
LPAR 1
LPAR 2

Figure 24 Hardware Server takes a decision based on the Target of the command

Besides, the modification and implementation on HMC side, we also established an experimental setup for the result analysis. The main challenge for the experimental setup was to choose the appropriate *flavor* of Power Server. By flavor we basically mean the capabilities of a server. Since our main focus was to analyze the results on the most complex commands, we chose the server with the most recent capability of running application migration via which a server can transfer a virtual machine with a running application from one server to another without the knowledge of the user. We also made sure that the HMC used was compatible to the flavor of the server we chose. The experimental

4.3 Resource Constraints identification for special LPAR

We basically worked on the HMC side; another big task of the project however was the designing and implementation of the special Logical partition. We contributed in that part by understanding some of the modules that were selected and implemented as an application on the special logical partition and identified the resources needed for that LPAR. The resources were selected keeping in mind the basic responsibility of the virtual machine. The resource constraints were - .

1. Operating System

The operating system we used was AIX (Linux flavor of IBM). We could have chosen any other operating system also; the major capability the operating system required here was it to be hypervisor aware as the power servers use the concept of *paravirtualization*.

2. Resources

The special logical partition was created as a *dynamic logical partition (DLPAR)*as the capabilities and the functions of this partition are supposed to be increased in future. Sufficient amount of primary memory, CPU cycles and I/O interaction devices were assigned as a part of "profile" of the logical partition.

In this section, a special mention of the I/O adapter is necessary. The special logical partition was made capable of interacting with the power server on the lines of NETC_NETS. The NETC_NETS protocol works over a Secure Socket Layer (SSL) with underlying protocol being TCP/IP. So, the NETC_NETS identifies the client and server from their IP addresses.

This required the special LPAR to be assigned with a I/O adapter to interact with the HMC. In our work, we assigned the special logical partition with a dedicated adapter considering the importance of role of the partition.

Chapter 5

EXPERIMENT AND RESULTS

5.1 Experimental Setup

To simulate the proposed idea, one Hardware Management Console (HMC) was used and two different Power systems (of the same series) were used. One of the Power systems was employed with no change in architecture and another system was the modified one on which we created the special LPAR.

To measure the latency, we used the hardware server (terminus of Hardware Server) logs and CIM-server logs to check for the time taken to complete the variable number of commands.

Some of the common functions involving maintenance and managerial tasks were selected and these commands were issued to the two Power Servers via common HMC.Various tasks involve multiple number of commands with some tasks like handshaking between the HMC and the server involve the number of commands in the order of hundreds while others like increasing the memory share of Dynamic Logical Partitions(DLPAR) involve the commands in order of thousands.

5.2 Results

The same set of tasks was given to both the servers and an improvement was noted in the proposed system which is in line with the expected results. The graph drawn below illustrates the comparison of the same set of commands that were sent from HMC to each of the servers in an independent experiment.

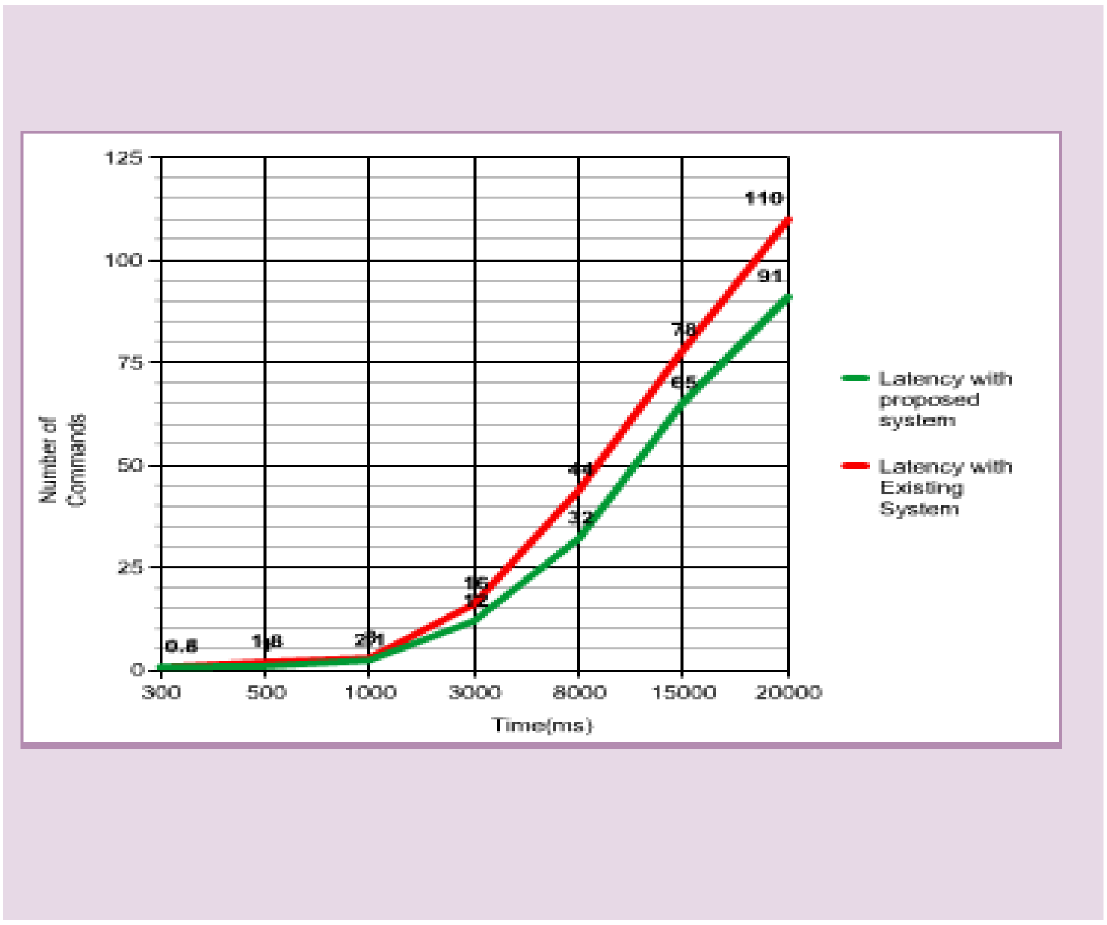


Figure 25 Interaction Latency as observed with proposed system and Existing system

The experimental set that was used contained 7 tasks. In the case of existing system, there is only a single channel of communication, so only one command can be issued while in the proposed system, there are two channels of communication, so two commands can be sent simultaneously on two different targets.

On the Hardware Server, based on the target of the command a decision would be taken and the command would be sent on one of the channels. Since, the proportion of the commands that are sent for management of the virtual machines is much more as compared to the proportion commands sent for maintenance, if both channels are free both the channels would be used for carrying the commands of management of Virtual Machines.

Chapter 6

CONCLUSION

The server virtualization technology holds a firm grip on affecting the performance of Power servers. So, it's necessary to try and reduce the latency of interaction and balance the workload of components. The Hardware management console (HMC) is what a client uses to interact with the power servers and in this process of interaction, all the commands issued pass through a single communication channel to a flexible service processor and FSP based on the command issued was for itself or for PHYP. The commands issued and intended for PHYP forms the focus of our attention.

We try to reduce the workload of FSP by directing these commands directly to a newly introduced logical partition. On the HMC side, a decision would be taken based on the target of the message, whether to send the message to the newly introduced LPAR or to the FSP.

The function of this LPAR is to directly interact with the hypervisor and thus the workload of FSP would be reduced as well as latency of interaction would be reduced.

In the current architecture, there is only one special logical partition available, but one more redundant special partition can be introduced which will help in achieving better communication as one more channel of communication would be introduced and will also provide robustness as in case of failure of one of the partitions. The basic overhead by introducing this redundant partition is the workload of maintaining the same state on both the partitions as now the communication could be possible from any of the available partitions.

In future, more workload of FSP would be shifted onto this special LPAR to further improve the efficiency.

REFERENCES

`

[1]Mendel Rosenblum,TalGarfinkel “Virtual machine monitors: current technology and future trends “ ,IEEE Computer Society,2005

[2]James E.Smith, Ravi Nair “The Architecture of Virtual Machines”,IEEE Computer Society,2005

[3]An executive guide to IBM’s strategy and roadmap for its integrated operating environment for Power Systems, IBM White Paper, 2012.

[4]W. J. Armstrong, R. L. Arndt, D. C. Boutcher, R. G. Kovacs, D. Larson, K. A. Lucke, N. Nayar, and R. W. Swanberg, ‘‘Advanced Virtualization Capabilities of POWER5 Systems,’’ IBM Journal of Research & Development 49, No. 4/5, 523–532 2005.

[5]Yaozu Dong*, XudongZheng*, Xiantao Zhang*, Jinquan Dai*, Jianhui Li*, Xin Li*, Gang Zhai*,Haibing Guan “Improving Virtualization Performance and Scalability with Advanced Hardware Accelerations”,IISWC,2010 .

[6]Wing-Chi Poon , Aloysius K. Mok”Improving the Latency of VMExit Forwarding in RecursiveVirtualization for the x86 Architecture” ,Hawaii International Conference on System Sciences, 2012 .

[7]“AMD64 Architecture Programmer’s Manual, Volume 1, 2 and 3”,http://www.amd.com/ (Publication number 24592, 24593,24594; revision 3.15), November 2009.

[8] “IA-32 Intel Architecture Software Developer’s Manual, Volume1, 2A, 2B, 3A and 3B”, http://www.intel.com/ (Order number253665-025US, 253666-025US, 253667-025US, 253668-025USand 253669-025), November 2007

[9]DulyawitPrangchumpol, Siripun Sanguansintukul1and PanjaiTantasanawong “Server Virtualization by User Behaviour Model using a Data Mining Technique – A Preliminary Study”Internet Technology and Secured Transactions, 2009.

[10]Allyson Brito,LoïcFura,Bartłomiej Grabowski “IBM PowerVM Virtualization Active Memory Sharing”,IBMRedpaper

[11]G. J. Popek, R. P. Goldberg “Formal Requirements for Virtualizable Third Generation Architectures”, Communications of ACM,July 1974, pp.412-421

[12]John Scott Robin “Analyzing the Intel Pentium’s Capability toSupport a Secure Virtual Machine Monitor”, Master’s Thesis, Sep1999, Naval Postgraduate School, Monterey, California, USA

[13]Rich Uhlig,GilNeiger,DionRodgers,Amy L.,Santoni, Fernando,C.M. Martins, Andrew V., Anderson, Steven M..Bennett,AlainKägi, Felix H.,Leung. ,Larry Smith “Intel Virtualization Technology”,IEEE Computer Society,

[14]“VirtualizationOverview”http://www.vmware.com/solutions/whitepapers/virtualization.html.

[15]Nicolas Guerin,JimiInge,NarutsuguIto,RobertMiciovici,RajendraPatel,ArthurTörök "IBM PowerVM Virtualization Managing and Monitoring", IBM redbooks

[16]Wei Chen, Hongyi Lu, Li Shen, Zhiying Wang, Nong Xiao, Dan Chen "A Novel Hardware Assisted Full Virtualization Technique" , The 9th International Conference for Young Computer Scientists , 2008.

[17]www. Wikipedia.org.

[18]StephenHochstetler,JunHeumMin,MattRobbins,NancyMilliner,NarendChand,Syamsul Hidayat "Hardware ManagementConsole V7 Handbook",IBM Redbooks

[19]EnriquilloValdez ,ReinerSailer, Ronald Perez "Retrofitting the IBM POWER Hypervisor to Support Mandatory Access Control", 23rd Annual Computer Security Applications Conference,2007

[20]www.howstuffworks.com

[21]Mendel Rosenblum, Carl Waldspurger, "Decoupling a logical device from its physical implementation offers many compelling advantages.",Acmqueue

[22]Ricardo Lent "Evaluating the Performance and Power Consumption of Systems with Virtual Machines" Third IEEE International Conference on Coud Computing Technology and Science, 2011

[23]David Watts,RandallDavis,RichardFrench,LuHan,DaveRidley,Cristian Rojas “IBM PureFlex System and IBM Flex System Products and Technology” IBM Red Books

[24]W. J. Armstrong,R. L. Arndt,T. R. Marchini,N. Nayar,W. M. Sauer” IBM POWER6 partition mobility: Moving virtual servers seamlessly between physical systems

[25]G. T. McLaughlin ,L. Y. Liu,D. J. DeGroff,K. W. Fleck “IBM Power Systems platform: Advancements in the state of the art in IT availability”,IBM Journal, 2008

[26] Paul Barham_, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,
Alex Ho, Rolf Neugebauery, Ian Pratt, Andrew War_eld “Xen and the Art of Virtualization”,ACM 2003

[27]“IBM PowerVM Virtualization Technology on IBM POWER7 Systems .A Comparison of PowerVM and VMware vSphere (4.1 & 5.0) Virtualization Performance”, White Paper, www.TheEdison.com.

[28]Allyson Brito,LoïcFura,Bartłomiej Grabowski “IBM PowerVM Virtualization Active Memory Sharing”,IBMRedpaper

[29]”Under the Hood: POWER7 Logical Partitions”,April 2013,IBM

[30]KeigoMatsubara ,Matt Robbins ,Ron Barker,TheeraphongThitayanun “Effective System Management Using the IBM Hardware Management Console for pSeries”, IBM Redbooks

[31]StuartDevenish,IngoDimmer,RafaelFolco,MarkRoy,StephaneSaleur,OliverStadler, Naoya Takizawa “IBM PowerVM Virtualization Introduction and Configuration”,IBM Redbooks

[32]Nicolas Guerin,JimiInge,NarutsuguIto,RobertMiciovici,RajendraPatel,ArthurTörök “IBM PowerVM Virtualization Managing and Monitoring”, IBM redbooks

[33]ww.techtarget.com

APPENDIX

ABBREVIATIONS

HMC - Hardware Management Console

FSP- Flexible Service Processor

VM- Virtual Machine

LPAR- Logical Partition

DLPAR- Dynamic Logical Partition

SSL- Secure Socket Layer

PHYP-Power Hypervisor

VIOS- Virtual I/O server

VIOC- Virtual I/O client

VNIC-Virtual Network Interface card

HDECR-Hypervisor Decrementor

SMT- Simultaneous Multithreading

PURR- Performance Utilization Resource Register

AMS-Active Memory Sharing

AME-Active Memory Expansion

CMM – Collaborative Memory Manager

VASI - Virtual Asynchronous Service Interface

AMSM - Active Memory Sharing Manager

LMB - of Logical Memory Blocks

TCE- Translation Control Entry

PLIC - Platform Licensed Internal Code

TLB - Translation Look Aside Buffer

MMU - Memory Management Unit

DHCP – Dynamic Host Configuration Protocol

ASMI - Advanced System Management Interface

SCSI – Small Computer System Interface