Lappeenranta University of Technology
Department of Information Technology

Server Virtualization

The topic of the Thesis has been confirmed by the Departmental Council of the Department of Information Technology on 12 March 2003.
Examiner: Pekka Toivanen
Supervisor: Ari Maaninen

October 13, 2003

Niko Ronkainen
Ojahaanrinne 3 C 39
01600 Vantaa
040-5679096

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto
Niko Ronkainen

Palvelinten virtualisointi

Diplomityö
2003
77 sivua, 17 kuvaa, 16 taulukkoa, 2 liitettä
Tarkastaja: Professori Pekka Toivanen
Hakusanat: virtualisointi, IA-32 arkkitehtuuri, virtuaalikone

Virtualisoinnin ideana on kuvata tietotekniikkaan liittyvät laiteresurssit ryhminä. Kun jonkin tehtävän suoritukseen tarvitaan resursseja, ne kerätään erikseen jokaisesta ryhmästä. Virtualisoinnin yksi osa-alue on palvelimen tai palvelinten virtualisointi, jossa pyritään hyödyntämään palvelinlaitteisto mahdollisimman tehokkaasti. Tehokkuus saavutetaan käyttämällä erillisiä instansseja, joita kutsutaan virtuaalikoneiksi.

Tässä diplomityössä esitellään ja verrataan erilaisia palvelinten virtualisointimalleja ja tekniikoita, joita voidaan käyttää IA-32 arkkitehtuurin kanssa. Eroa virtualisoinnin ja eri partitiointitekniikoiden välillä tarkastellaan erikseen. Lisäksi muutoksia, joita palvelinten virtualisointi aiheuttaa infrastruktuuriin, ympäristöön ja laitteistoon käsitellään yleisellä tasolla. Teorian oikeellisuutta todistettiin suorittamalla useita testejä käyttäen kahta eri virtualisointiohjelmistoa.

Testien perusteella palvelinten virtualisointi vähentää suorituskykyä ja luo ympäristön, jonka hallitseminen on vaikeampaa verrattuna perinteiseen ympäristöön. Myös tietoturvaa on katsottava uudesta näkökulmasta, sillä fyysistä eristystä ei virtuaalikoneille voida toteuttaa. Jotta virtualisoinnista saataisiin mahdollisimman suuri hyöty tuotantoympäristössä, vaaditaan tarkkaa harkintaa ja suunnitelmallisuutta. Parhaat käyttökohteet ovat erilaiset testiympäristöt, joissa vaatimukset suorituskyvyn ja turvallisuuden suhteen eivät ole niin tarkat.

ABSTRACT

Lappeenranta University of Technology

Department of Information Technology

Niko Ronkainen

Server Virtualization

Master's thesis

2003

77 pages, 17 figures, 16 tables, 2 appendices

Examiner: Professor Pekka Toivanen

Keywords: virtualization, IA-32 architecture, virtual machine

The idea of virtualization is to describe computing resources as pools. When a task is performed in a virtualized environment, the required resources are gathered from different pools. Server virtualization is one of the subcategories of virtualization and its main purpose is to enable the efficient use of physical server hardware. This is achieved by special instances called virtual machines.

This thesis concentrates on presenting and comparing different server virtualization schemes and techniques that are available to the IA-32 architecture. The difference between virtualization and various partitioning schemes are also examined as well as changes that the server virtualization causes to the existing infrastructure, environment and hardware. To support the theoretical point of view, various tests were performed with two separate virtualization software.

The results of the tests indicate that server virtualization reduces overall performance and increases the complexity of the environment. New security issues also arise, since physical isolation does not exist in a virtualized environment. In order to benefit from virtualization in a production environment, careful consideration and planning are required. Test environments where performance and security are not the main requirements, benefit the most from virtualization.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ACPI	Advanced Configuration and Power Interface
API	Advanced Programming Interface
ARP	Address Resolution Protocol
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CSIM	Complete Software Interpreter Machine
DMA	Direct Memory Access
HVM	Hybrid Virtual Machine
IA-32	Intel 32-bit processor architecture
IDE	Integrated Drive Electronics
IRQ	Interrupt Request
ISA	Instruction Set Architecture
IP	Internet Protocol
LPAR	Logical Partitioning
MAC	Media Access Control
MMX	Multimedia Extensions
NIC	Network Interface Card
OS	Operating System
PC	Personal Computer
PIO	Programmable Input/Output
PPAR	Physical Partitioning
SCSI	Small Computer System Interface
SMP	Symmetric Multiprocessing
TCP	Transmission Control Protocol
VMM	Virtual Machine Monitor

ACKNOWLEDGEMENTS

I would like to thank both the examiner and the supervisor of this thesis for the advices and guidance that I have received during the writing process. I would also like to present my gratitude to Nordea IT for providing financial support and the possibility to work with server virtualization.

I wish to present a special acknowledgement to my parents and to my sister who have supported and encouraged me during my studies and thesis work.

1. INTRODUCTION

Within the past decade, the processing power of microprocessors has increased significantly. Especially processors based on IA-32 architecture have become popular since they are inexpensive and supported by various operating systems and applications. In addition to use in desktop computers, IA-32 architecture has also become widely used as server platform. [Int03a], [McI03a].

Whereas desktop computers can contain a large number of applications and configuration can be complex, server systems have been traditionally built by running a single application on one physical server. This approach has several benefits since configurations can be kept simple and in case of a hardware failure, only one application would be affected. The drawback is that certain applications do not require and are unable to benefit from the increased processing power. Typical examples of this phenomenon are applications where processing power needed to complete a request or transaction and the level of resource utilization have been the same for a long time. After a normal warranty period, the maintenance costs of hardware typically increases and at certain point, replacing existing hardware with a new one becomes more cost efficient. Transfering applications directly to new and more powerful hardware usually means that the new system becomes underutilized. [McI03a].

Within the past decade the number of actively used applications has increased. Entirely new computing areas such as the Internet have been the main reason for this growth. If the new functionality was impossible to achieve by modifying the existing system, a new system was required. Typically, increment in the number of applications also resulted in an increment in physical servers. Additional needs like a separate test environment has made the situation even worse. When the costs of e.g. maintenance and location facilities are taken account, the overall expenses quickly increase to an intolerable level. In the mainframe environment, a similar phenomenon has not occurred. Main reasons to this have been the price of the mainframe system and available partitioning techniques. Due to the high price, mainframes have been used only in situations where performance and availability of the IA-32 architecture has been insufficient. [McI03a].

There are several approaches available to use resources more efficiently. If several servers are using the same application to provide different content, content management and distribution can be centralized to a single server or a smaller number of servers. Also several separate and independent applications can be combined into a single server. Each solution has its benefits and drawbacks. Since changes to the existing infrastructure should be minimized, a solution where administrators and users would not even know that environment has changed is preferred. Server virtualization can be seen as one solution to these issues. [McI03a].

This thesis concentrates on presenting and comparing different server virtualization schemes and techniques that are available for IA-32 architecture. The difference between virtualization and more traditional partitioning schemes are also examined. In addition to presenting virtualization schemes and techniques, possible changes caused by virtualization to infrastructure, environment and hardware are also discussed. To support the theoretical point of view, various tests were performed with two separate virtualization software. This thesis is also a part of the ongoing study on different virtualization techniques in Nordea.

1.1 Idea of virtualization

The main idea of virtualization is to provide computing resources as pools. Depending on the needs, resources are then assigned to different applications either manually or dynamically from different pools. The scope of virtualization can vary from a single device to a large data center and virtualization can be applied to different areas such as servers, networks, storage systems and applications. The difference between a traditional data center and a virtualized environment is presented in Figure 1. [Hew03].

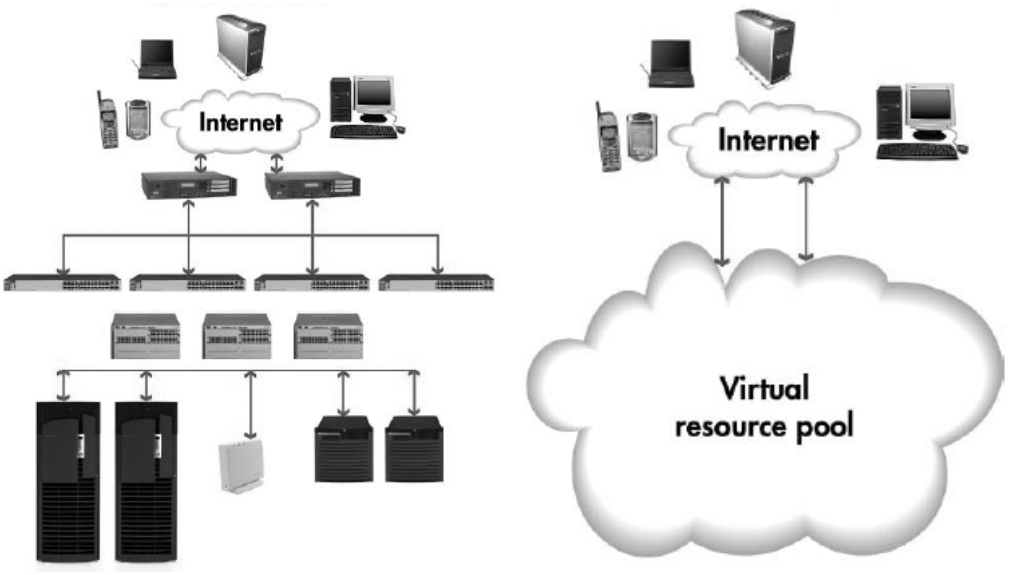

Figure 1 Common data center presented as a traditional and virtualized environment. [Hew03].

The focus of server virtualization is to create virtual machines or virtual environments by using a normal server hardware and a virtual machine software. Virtual machine software enables sharing physical hardware among several instances called virtual machines. Sharing is done by creating a special virtualization layer, which transforms physical hardware into virtual devices seen by virtual machines. The most visible change is the possibility to run different operating systems (OS) within the same hardware concurrently. [Hew03], [Smi01a]. Figures 2 and 3 illustrate the difference between physical servers and virtual machines. Figure 2 presents four servers where each server has its own processor, memory, local disk and network connection. Figure 3 presents four virtual machines and a virtualization layer.

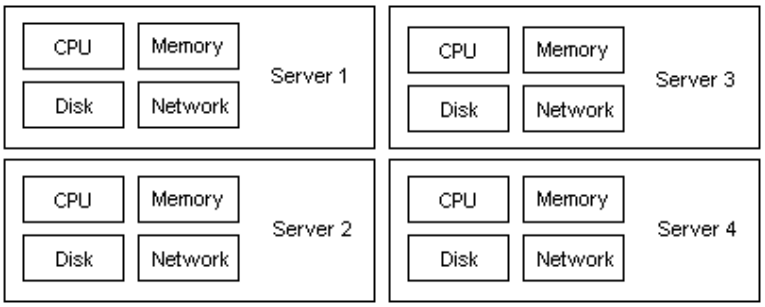

Figure 2 Physical machines.

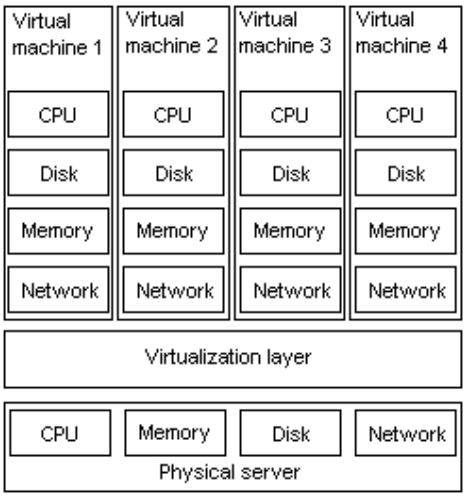

Figure 3 Virtual machines in a single physical server.

1.2 Server virtualization compared to system partitioning

The special feature of mainframe systems is different partitioning schemes. By using these schemes hardware resources can be divided into several partitions. Currently there are two main schemes that are widely used: Logical Partitioning (LPAR) and Physical Partitioning (PPAR). Logical partitioning is similar compared to virtualization, since both of these techniques describe hardware resources as pools. The terms logical partitioning and virtualization are therefore often used in the same context. A practical difference is that within mainframes, the partitioning is typically done without sharing a single processor among multiple partitions and different partitions must use the same OS. The term physical partitioning is used when resources are physically divided in the hardware level. Although resource sharing using physical partitioning is not as flexible as in logical partitioning, the partitions are fully isolated and overhead does not exist. [Int01], [McI03a], [Sun03].

A mainframe architecture typically consists of separate CPU/memory cards, I/O cards and interconnection bus. Combinations of these cards are used to create a building block. An example of a server hardware that contains 4 building blocks is presented in Figure 4. The physical building blocks are the limiting factor when physical partition is created. Logical partition does not have similar limitations. [Int01]. Figures 5 and 6 present the difference

between physical and logical partitioning. In Figure 5 there are two physical partitions: Physical partition 1 (3 building blocks) and Physical partition 2 (1 building block).

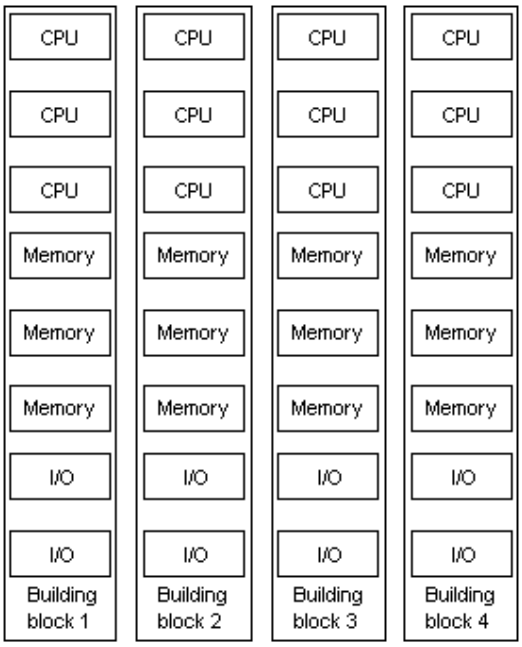

Figure 4 Building blocks.

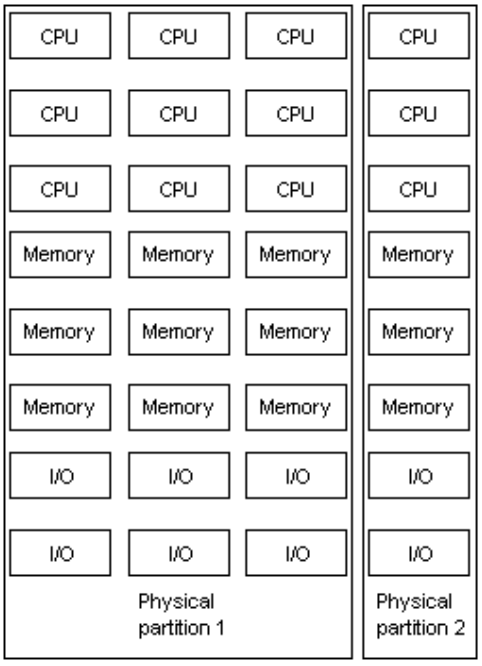

Figure 5 Physical partitioning.

Logical partitioning with three separate partitions is presented in Figure 6. This partitioning example presents the typical scheme of production and testing environments within a single physical system: both environments are separated and the production environment has more resources than the testing environment. Due to the flexibility of logical partitioning, partitions can be configured to support CPU, memory and I/O sensitive applications.

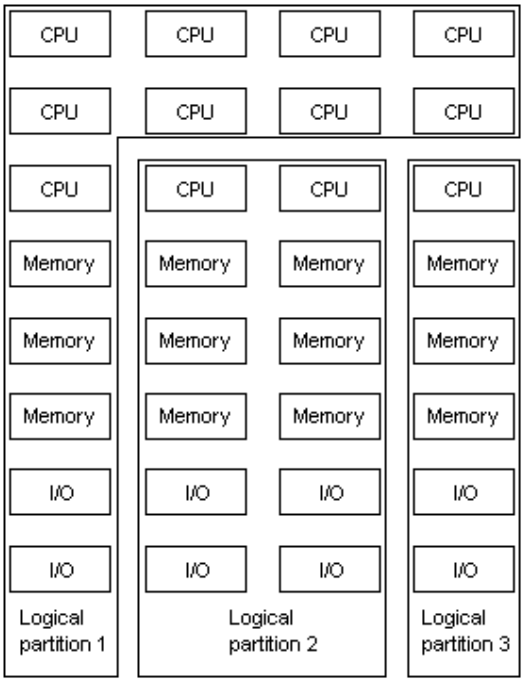

Figure 6 Logical partitioning.

1.3 Server virtualization compared to workload management

Server virtualization and workload management both point to the same target of using resources more efficiently and describing them as resource pools. The practical approach, however, is different. The main idea of the workload management is to provide resources to different tasks as efficiently as possible. A common solution to providing resources is to create a pool by using several physical servers and workload management software. Instead of sharing or partitioning resources of a single physical server, the most suitable system from the resource pool is selected. [Day02].

The common problem with workload management is its limited usage. All systems in the pool must be hardware compatible with each other and they must use the same OS. Besides hardware limitations, there are also restrictions in the software side: only those tasks can be used that the workload management software is capable of distributing. Also the whole concept of separate virtual machines and its benefits does not exist. In order to enable the distribution of tasks, a tight cooperation between the hardware, OS and workload management software is required. Usually this means that the whole system must be obtained from a single vendor. Due to these restrictions, the workload management software is commonly used only in a single vendor UNIX environment. [Day02].

1.4 Server virtualization compared to consolidation

The term consolidation is typically used to describe a process that aims at providing existing services more efficiently and thus save costs. One part of consolidation is server consolidation, which can be divided into three different categories:

- Logical consolidation or standardization. The control of different platforms and environments is centralized under single organization. By using standards, the system management is improved and the complexity of the environment is reduced.
- Location consolidation. The number of physical locations containing hardware is reduced.
- Physical consolidation. The number of physical hardware is reduced. [Cog02], [Int02].

Server virtualization is often mentioned as a consolidation scheme or in the same context as consolidation since virtualization provides similar benefits. When applications are running in underutilized servers, the efficiency can be increased by moving applications to virtual machines and shared hardware. Virtualization software creates a hardware standard, since every virtual machine runs in an identical environment. The benefits of physical consolidation can be achieved by a migration of several physical servers into virtual machines running on single server. Virtualization makes the practical part of consolidation

process easier, but location consolidation cannot be done using it. Virtual machines can be transferred as files over network instead of transferring physical hardware. [Roa03].

Server virtualization can be seen in three major roles in consolidation:

- Multiple existing systems are combined into a single system, whose resources are used efficiently (physical consolidation).
- Instead of replacing aged hardware with new, old systems are migrated to virtual machines.
- Virtualization is used to provide a platform where creating a single system or an entire environment can be done in a short time span without purchasing new hardware.

Physical consolidation using server virtualization is presented in Figure 7. In this scenario, selected targets are assumed to be underutilized with an average load of 1-10%. Half of the servers are assumed to be production servers, the rest of them are either development or test servers. In the current setup, each application is installed to a separate server to make systems as simple as possible. Therefore, a hardware for six separate servers is required. Using virtualization to perform consolidation, separate physical servers are replaced by a single server and software that enables virtualization. Each server is then replaced by a virtual machine. After virtualization is performed, each virtual machine can be administered, backed up and used as if it was an independent machine. The result of the process is that the number of physical servers has reduced.

Transfering systems from old hardware to virtual machine has certain benefits even though obtaining new server hardware would be required. Having underutilized servers is avoided since the new hardware is shared, systems are transferred to a more standardized environment and the number of physical servers is reduced. The workload of migration to virtual machines and of replacing aged hardware is usually the same, since both operations contain similar tasks (e.g. transfering disk partitions). A shift to the virtualized environment is the most suitable solution in situations where changes to OS and application are not needed and virtualization as a technology is acceptable.

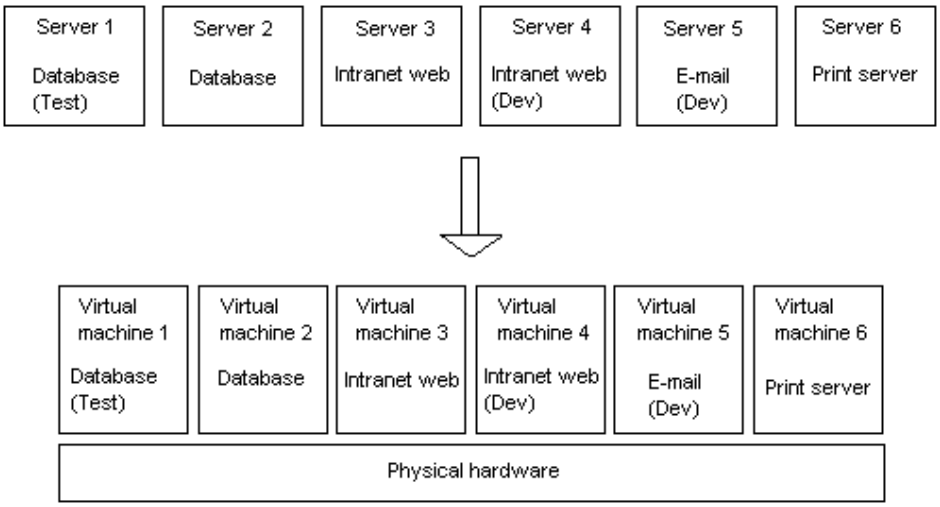

Figure 7 Physical consolidation using server virtualization

During planning, development and testing phases of a new system, a number of different environments are needed. While creating these environments requires hardware, the obtained hardware does not necessarily satisfy the requirements of the final production environment. If a separate production and development environment is needed, both environments also require separate hardware. In the production environment, performance is one of the main criteria while development and testing can be done in more modest environments. With virtualization, the entire environment can be built quickly by using virtual machines instead of separate physical machines. The solution is also cost effective since several virtual machines can run on a single server.

2. SERVER VIRTUALIZATION TECHNOLOGIES

Server virtualization is one example of how virtual machines can be used. The term virtual machine typically refers to a system where direct interaction between the OS and physical hardware does not exist. The interaction is replaced with software that enables execution of virtual machine operations on physical hardware. Generally each system is based on a hardware that implements an instruction set architecture (ISA). ISA contains detailed information about the functionality of a processor, e.g. available instructions, registers and states. An example of an ISA is x86 architecture that is used in Intel processors. Each x86 compatible processor must therefore meet the requirements that the x86 ISA description contains. The same rule applies to OS. If the OS cannot understand the ISA, it cannot execute any programs either. If the ISA of underlying hardware can be shared, running multiple OS simultaneously becomes possible. [Smi01a].

When the selected ISA does not allow sharing of hardware directly, only single OS with direct access to hardware can be used concurrently. A virtual machine, however, can be created by using single OS and a separate program that replicates ISA. A single OS controls the hardware with its own device drivers and provides hardware access in the form of system calls. The replicated copy has the same functionality as the original ISA but no direct access to hardware. Each time the virtual machine requires a hardware access, it performs normal ISA operations to the replication program. The replication program then converts ISA operations to matching system calls and performs them to OS and device drivers. After the operation is completed, the result is sent back to the virtual machine in reverse order. This way multiple OS can run concurrently while only one OS is required to have direct access to hardware. [Smi01a].

If full ISA is replicated and shared among multiple OS environments, the replication software is typically called as Virtual Machine Monitor (VMM). VMM is also usually the main part of server virtualization software. Figure 8 provides an example of VMM usage where the underlying hardware is shared among two different operating systems. At the same time, it represents the idea behind server virtualization. [Pop74], [Smi01a].

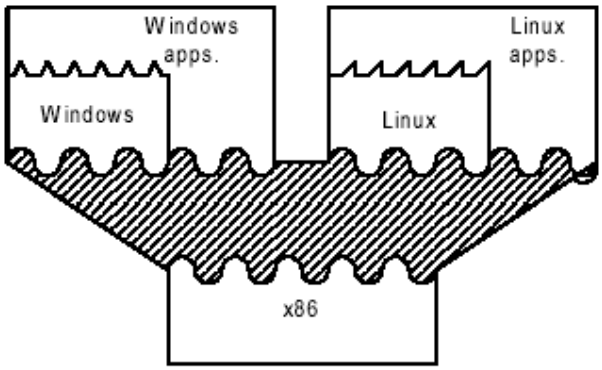

Figure 8 Support for multiple operating system environments on the same hardware. [Smi01a].

While partitioning schemes for mainframe systems were introduced for the first time already in the 1960's, virtualization for the IA-32 architecture was introduced only in the late 1990's. One reason has been the design of Personal computers (PC). Video cards and disk controllers, for example, were designed only to be used with one OS at a time without any sharing. Similarly, the IA-32 architecture processor cannot be virtualized as such due to certain restrictions. Virtualization has also faced a competitor from the emulator. Emulators for x86 processors have been available as well as emulation using advanced programming interfaces (API) for operating systems. Virtualization, however, has some benefits over emulators since it does not require additional APIs and it can provide the ability to run various operating systems while maintaining relatively good performance. [Law99], [Rob00], [Smi01b].

Server virtualization can be divided into three separate layers:

- Host (Physical hardware and operating system)
- Virtualization layer
- Guests (Virtual machines). [VMw99].

Figure 9 presents these three layers as a hierarchical model. At the lowest level, the host contains all procedures that are close to the hardware. On the top, the guest is fully implemented in software.

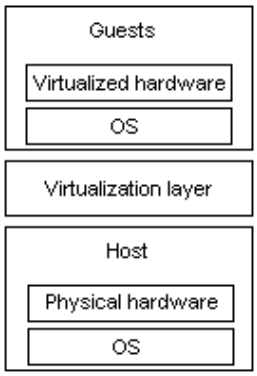

Figure 9 Three layers of server virtualization technology. [VMw99].

The host contains the physical hardware that is being virtualized and OS that is used to allocate hardware resources. The difference between a normal server and a host is in the use: the host focuses only on providing a virtual layer while the normal server is typically used to provide one or more services (e.g. e-mail and database).

The virtualization layer is created with virtualization software. The purpose of the virtualization layer is to share the host's hardware resources among guests. Therefore, virtualization does not change the hardware architecture that can be done with emulation. Besides resource sharing, the virtualization layer can also be used to provide other features such as isolation. Although the host's OS can provide isolation between processes and secure memory management, the isolation between guests is typically handled by the virtualization layer. [VMw99].

Guests are either virtual machines or virtual environments that can only see resources that are provided to them by the virtualization layer. The term virtual machine is used if all physical components of the hardware are virtualized and the guests and the host do not have to possess the same OS. The term virtual environment is used when the same OS is used in both host and guest systems. [SWs03], [VMw99].

2.1 Different virtualization approaches

Normally all instructions issued by the OS are executed directly on hardware. If the hardware is shared among multiple operating systems, a portion of instructions may be required to be executed using the software instead of the hardware. The difference between hardware and software execution portions can be used to determine the VMM type. The following classification is typically used to create a distinction between emulation, real machine and different VMM types:

- Real machine. Everything is executed directly on hardware.
- Virtual Machine Monitor (VMM). A large part of instructions is executed directly on hardware. The rest of the instructions are executed on software.
- Hybrid Virtual Machine (HVM). All privileged instructions are emulated using software.
- Complete Software Interpreter Machine (CSIM). Software is being used to emulate every processor instruction. [Rob00].

VMMs can be divided further into two different categories based on the control over physical hardware. Type I VMM is a minimal operating system, whose main purpose is to provide virtualization layer. Type II VMM is a normal application running under normal OS. It is often called as hosted architecture. Figure 10 presents the difference between Type I and Type II VMM. [Rob00].

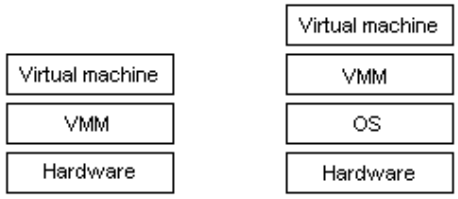

Figure 10 Type I virtual machine monitor (left) and Type II virtual machine monitor (right). [Rob00].

Hardware virtualization can be provided in two different ways: by a replication of the host ISA or by modifying the guest OS. The replication of the host ISA provides full virtual environment, including basic input/output system (BIOS) that is used in hardware

detection. The advantage of the ISA replication is that the guest OS sees shared resources as if they were physical devices. Modifying the guest OS consists of changing hardware specific calls to normal system calls and recompiling the OS. The result is a modified OS that can run as a normal process without the need of direct hardware access. Although only the kernel part of OS would require modification, obtaining the source code of the kernel and creating modifications are not always possible. [Dik00], [Smi01a], [Whi01].

There are currently two vendors that use the ISA replication in commercial server virtualization products. VMware offers GSX Server and ESX Server products and Microsoft is developing a product called Virtual Server. GSX Server is a normal application and requires the OS to provide hardware access (Type II VMM). ESX Server, on the other hand, uses hardware resources directly and it contains a minimal OS to start the virtualization layer (Type I VMM). Microsoft Virtual Server uses the same approach as VMware GSX Server. [Con03], [VMw03c].

Modifying the kernel of OS is possible when the source code of the kernel is available. User-mode Linux and Plex86 are both based on this approach and their underlying principle is the same: A modified Linux kernel is used as a user process on a system that runs Linux kernel. Both User-mode Linux and Plex86 are being distributed as patches to normal Linux kernel. Although Linux kernel is available to a number of different ISA architectures, kernel modifications are only available to the IA-32 architecture. [Dik00], [Law03].

2.2 Hardware virtualization

The focus of hardware virtualization is to enable virtualization in three areas: processor, memory and input/output (I/O) system. Hardware virtualization is required due to restrictions in the IA-32 architecture: it does not support virtualization in the hardware level. Processor, memory and I/O system of a single system are designed to be used only with one OS at a time. Removing this limitation is possible with special software that enables virtualization, thus sharing hardware safely. As a result, the physical hardware does not need changes and multiple OS can be used simultaneously. [Smi01b]. The

following chapter presents different areas of hardware virtualization in a more detailed level.

2.2.1 Processor

The processor provides resources for program execution in the form of instructions and registers. Instructions define single operations while registers are used to store code, data and state information. Besides executing commands and storing information, the processor also provides protection in the form of operating modes and levels. Most of the instructions and registers are used in normal program execution. The protection mechanisms in cooperation with OS ensure that the environment where programs are executed is safe. To enable e.g. changing the operation mode of a processor, registers are used to share information and instructions to perform the actual change. Due to the design of the IA-32 architecture, protection and sharing mechanisms work flawlessly only when a single OS is used at the same time. [Int03a].

The purpose of processor virtualization is to enable the use of execution resources and protection mechanisms of the processor without any limitations. A large portion of processor resources already support virtualization: their behavior and the result of operation is always the same regardless of the number of OSs running simultaneously. There are exceptions, though: there is a total number of 20 processor instructions that cause problems. These instructions can be divided into sensitive register instructions (8 instructions) and protection system references (12 instructions). Sensitive register instructions contain instructions that either read registers, change register values or change memory locations. Examples of altered registers are clock register and interrupt register. Protection system references, on the other hand, contain instructions that refer to storage protection system, memory or address relocation system. A list of sensitive register instructions and their operations are presented in Appendix 1. Protection system references and their operations are listed in Appendix 2, respectively. [Int03b], [Rob00].

A common feature of sensitive register instructions is that they modify or read values containing information about operating mode, status and the state of the processor. These instructions are normally used only by the OS under a privileged mode. In addition, due to

the register sizes, only values for one processor can be stored at a time. Facing problems is inevitable since sensitive register instructions can read register values also in an unprivileged mode. In a virtualized environment, certain registers are shared among several OSs. Without any additional protection, each virtual machine and the host OS itself would be capable of changing e.g. operating modes at the same time. Instead of executing these instructions directly on processor, virtualization software must emulate the execution and, for each virtual machine, create separate registers to prevent sharing. [Int03a], [Rob00].

Protection system references are related to instructions that either require certain protection level to enable execution or verification of the protection level during execution. In the IA-32 architecture, the protection is implemented using four privilege levels. Accessing a higher privilege level can be only done using a tightly controlled and protected gate interface. The highest privilege level (level 0) is normally used only by the kernel of the OS. The lowest level (level 3) is mostly used by normal applications, in other words user processes or applications running in a non-privileged mode. In the virtualized environment the OS of the virtual machine expects to have the same privilege levels available as if it would be the only OS in the whole system. The virtual machine, on the other hand, is usually executed as a normal user process with a privilege level 3. Virtualization software must therefore provide an illusion to the virtual machine that it can use protection levels without exceptions. The virtual machine itself is running at the same time as an user-mode process in the host OS. If a process running in a non-privilege mode performs a privileged call, a special trap is always generated. The VMM detects these traps and manages them by using software emulation to execute instructions. [Int03a], [Rob00].

2.2.2 Physical memory

Physical memory of the computer is used in cooperation with the processor. The IA-32 architecture contains various memory management features including segmentation and paging. Although using more sophisticated memory management techniques disables the possibility to use direct memory addressing, additional features enable more flexible use of memory and reliability for program execution. When the program allocates the memory and memory management features are enabled, the memory is not addressed directly, but using one of the three different memory models. Depending on the used memory model,

essential features such as address spaces and addressing model differ from another. After the operation mode of the processor and memory model are selected, the OS will know how the memory is handled. [Int03a].

The OS expects that a specific memory area is available for use and the area begins from a certain address. Firmware (BIOS) provides information to OS about the size of total memory available. Running multiple OSs at the same time causes errors since each of them is trying to use the same memory area. Memory management in the protected mode also uses several registers that are called Local Descriptor Table Register (LDTR), Interrupt Descriptor Table Register (IDTR) and Global Descriptor Table Register (GDTR). Using these registers is problematic since in a single physical processor, there are only one register of each type. Using multiple OS at the same time means that these registers and their content are shared between different operating systems. [Int03a], [Rob00].

Memory virtualization is done by address translation, since an additional level of translation is needed to provide memory mapping. The mapping is being done between addresses of VMM memory and virtual machine OS. While this solution creates an additional layer to memory management, the process itself is quite simple and the implementation requires only little overhead. In addition, separate LDTR, IDTR and GDTR registers are created for each virtual machine. [Ros03], [Smi01b], [Wal02].

2.2.3 Input/Output System

The communication of I/O peripherals is typically done using I/O ports that are provided by the system hardware. A processor can transfer data from I/O ports in a same way as in using memory based on addresses and instructions. I/O ports can be accessed in two different ways: Using separate I/O address space or memory-mapped I/O. The difference between these two alternatives is that memory-mapped I/O can be used as if it would be normal memory (e.g. same instructions as in normal memory operations). Separate I/O address space contains 65536 individual 8-bit ports that reside in a special area separated from physical address space. Transmitting data by separate address space is done using IN/INS and OUT/OUTS instructions and special registers. Likewise memory management, additional protection mechanisms are available when processor is used in a protected

mode. Mechanisms include privilege level and permission bit map for control access. In memory-mapped I/O, additional memory management features such as segmentation and paging also affect I/O ports. [Int03a].

In addition to I/O ports, Direct Memory Access (DMA) and interrupts are common parts of the system. DMA enables transfering data from peripherals directly into memory without using a processor. DMA transfer as an operation is fairly simple, since only the starting addresses, block length and operation type are needed. Interrupt is one of the two ways to stop execution of a currently running program on processor. Once the interrupt event is triggered, the processor halts the execution and switches over to handling the interrupt by using information in interrupt descriptor table (IDT). [Int03a], [Koz01].

In a virtualized environment, the guest OS expects to have the same I/O ports, DMA and interrupts available as in a real machine. Since the number of IRQ and DMA values are limited, this has caused problems already before virtualization. The most common solution to this problem has been using devices that can share e.g. IRQ value with another device. Virtualization uses the same technique by accepting I/O calls from the virtual machine, translating I/O call to suitable system call for underlying physical hardware and then performing the operation. Similarly, the results of the operation must be converted back to I/O call and relayed to the virtual machine. Addition to sharing, instruction SIDT used in interrupts is a part of sensitive register instructions. SIDT instruction is used to obtain information from IDT register that contains values for address and the size of the register. SIDT instruction can be called from a normal program without an exception and therefore e.g. reading IDT register values of the wrong virtual machine is possible if additional protection is not available. By creating separate IDT register for each virtual machine, this issue can be avoided. [Int03a], [Int03b], [Koz01], [Rob00].

2.3 Process and thread management

Virtualization does not basically change the behavior and usage of processes and threads. Virtual machines can be seen as normal processes in the host OS or sub processes under the VMM. The ISA replication typically hides single processes from virtual machines and

the host OS only sees the VMM process. Kernel modification requires some additions to normal process creation. Since trap generation and monitoring is required for signaling, it is useful to create a new process as a child of the process that handles trapping. Since trap generation and monitoring is usually performed in the host OS, creating a process in virtual machine means that the new process is seen also in the host OS. This approach also simplifies context switching because processes of virtual machines can directly notify the process to the host OS that manages trap monitoring. [Dik00], [VMw99].

2.4 Memory management

In virtualization, no additional modification is needed apart from address translation. The VMM allocates memory from the host OS as a normal application and manages address translation. The guest OS sees the memory area that the VMM provides as if it would be physical memory. Figure 11 presents a layered structure of memory management. Since all memory handling of the virtual machines is managed by the VMM, the VMM can also see the content of the memory that the guest OS as well as its applications use. Therefore, the security and isolation between virtual machines is provided by the VMM if ISA replication is used. If kernel modification is used, it must contain the required security and isolation features. [Dik01], [VMw99].

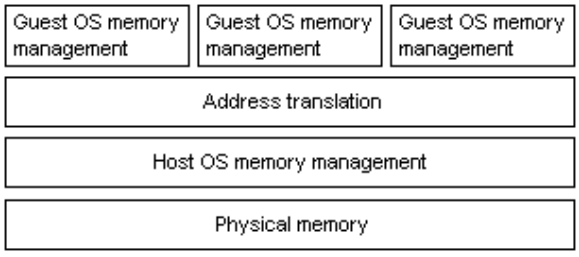

Figure 11 The layered structure of memory management.

The OS typically includes memory optimizations to improve memory management and overall performance. Since all memory management of the virtual machines is performed in a single place, separate optimizations can be implemented to VMM.

2.5 Disk management

While disk virtualization requires similar translation compared to memory virtualization, it provides some additional flexibility. Usually the operating system is installed to a physical partition in the hard disk. Besides partitions, virtual machines can also be installed to raw devices or as files in the host OS. The host OS can therefore see the virtual machine disk as a normal file in its own file system and e.g. the replication of virtual machine can be done by creating a copy of this file.

Disk virtualization is performed by the VMM, which provides physical disk partition or a file from the host system as a virtual device to the guest. The guest OS then sees the virtual device as a normal device with I/O addresses and interrupts. If a guest OS has drivers for the device, it uses them to install and configure hardware. Every time the guest OS performs a reading or writing operation to disk, the VMM receives the I/O request from the virtual device. A request is then translated to matching I/O action for the hardware based on a virtual machine disk configuration (raw device, physical partition or file in a host file system). After the operation is performed in the host system, the results are sent back to the guest using a virtual device. Figure 12 illustrates the difference between a normal and a virtualized I/O system. [Smi01b].

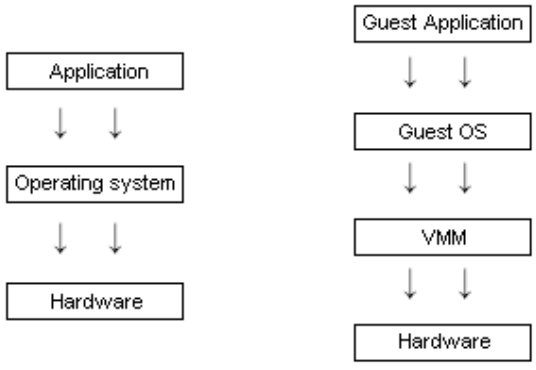

Figure 12 Normal (left) and virtualized (right) Input/Output system. [Smi01b].

2.6 Network management

In addition to processor, memory and disk, a network connection is also one of the main components of a server hardware. As with disks, the typical requirement of a network card is low latency and high throughput. Since additional configurations such as load balancing and fail-over are often used in servers, the same functionality should be also available to virtual machines. These configurations might require that e.g. a single virtual machine can use several physical NICs of the host.

In the ISA replication, providing a network connection to the virtual machine is basically done the same way as with a disk apart from one exception. Creating a network can be accomplished without a physical network interface card (NIC). A connection to the existing network is not necessarily required either, since the VMM manages all requests. The VMM provides virtual I/O addresses and a virtual IRQ that are used with the virtual network adapter. The guest OS then sees the virtual network adapter as a normal device and uses device drivers to enable communication. Each time a virtual machine sends a packet to the network, all I/O operations are first performed with the VMM and then with the host OS and actual hardware. Receiving a packet is done in reverse order. Figure 13 presents a process of sending a packet to the network and receiving a packet from the network in the ISA replication. [Sug01].

In kernel modification, networking can be arranged by creating a special device in the host that enables communication between the kernel of the host OS and the virtual machines. The modified kernel of guest OS then knows that the network connection is provided by a special device instead of the normal NIC. The host OS sees the created device as a normal network interface, which can be configured to send and receive packets to network using NIC. Communication between virtual machines only can be arranged e.g. by creating a separate instance that routes packets between virtual machines. [Dik00].

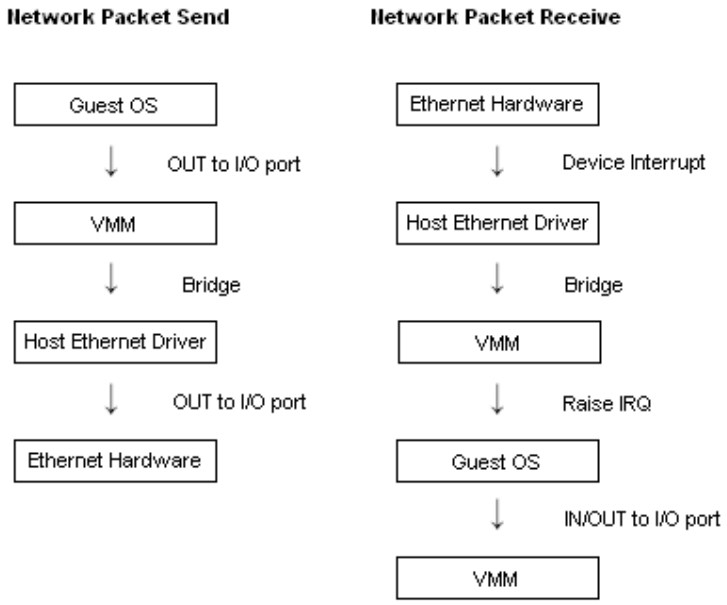

Figure 13 The process of sending and receiving packet between virtual machine and network. [Sug01].

Communication directly to the network requires additional changes to the host system. Each NIC has a special hardware address called Media Access Control (MAC) that has been assigned to it during manufacturing process. MAC address is used in physical data transmission to identify the sender and receiver. For each manufacturer, separate address spaces have been assigned in order to avoid overlapping. Since a virtual machine does not have physical hardware, the host system's NIC is used to communicate with the network. Therefore the host system and VMM must have the capability to send and receive packets that contain a MAC address different from the actual physical hardware. In addition, VMM must provide unique MAC addresses to virtual machines. [VMw02].

2.7 Device and hardware access

In addition to disk and network, there are several other devices that are used during normal computer operation. Most common ones are display adapter and various input devices such as keyboard and mice. If the host OS can detect the hardware and provide a suitable driver, sharing devices among virtual machines is possible. The VMM must then either know how to convert and monitor device calls or allow direct execution ("pass through") using

hardware. If the shared device uses e.g. privileged instructions, additional monitoring and conversion is needed. [Sug01].

Because of the large amount of different devices available in the market, providing support for each of them in a virtualized environment is very difficult. Using certain devices is not practical either, since overhead caused by virtualization can dramatically decrease the performance. Therefore, providing a combination of a few basic components without e.g. multimedia features is the most common solution.

2.8 Isolation and security

One of the most important issues in virtualization is the isolation between separate virtual machines and between a virtual machine and the host OS. While the host OS can provide efficient isolation between processes, hardware virtualization in the ISA replication can create certain problems. The same processor instructions, for example, are available to virtual machines as well as to the host OS, since the ISA will not change. Problems can occur if the virtual machine uses undocumented or undefined features of the hardware. If the software that provides virtualization does not detect e.g. instructions that allow reading register values without hardware protection, isolation and therefore security are not guaranteed. There is no good solution available to this problem, since new operating systems and modifications to existing ones are continuously introduced. [VMw99].

While the kernel modification enables the operating system to run as a user process that has limitations, additional security provided by the kernel mode with an assistance of the hardware is lost. Since the OS runs as a process, the kernel memory resides within the same address space of the process and can therefore be changed by a user space program. Security can be arranged by creating a write protection to the kernel memory when the user process is running and releasing it in the kernel mode. [Dik00].

Security issues must be considered due to the basic nature of virtualization. Since resources are shared among multiple instances, separating critical applications to separate physical machines is no longer possible. If the server uses e.g. confidential information during

operation, virtualization should be carefully considered, since providing the same security to virtual machine than physical server is very difficult. The following example illustrates this point: when virtual machines share NIC, it means that the same physical hardware is used to send and receive packets among virtual machines. Since it is possible for a virtual machine to receive packets destined to a different system, additional protection mechanisms are needed. Similar problems appear also in the disks of virtual machines regardless of the disk type. In addition, the host OS has access to all hardware so it can also see the disks of virtual machines and possibly read and modify their contents. [Rob00]. While security features such as encryption of network traffic and disk storage would provide necessary security level compared to a physical machine, these operations often consume a large amount of processing power and thus the overhead caused by virtualization will increase.

Generally, kernel modification can be considered as a better solution than ISA replication. The main reason is that in the ISA replication, virtualization is done by additional drivers and there are no guarantees that the driver can handle all requests issued by the OS. Through kernel modification, also harmful instructions can be handled. They are disabled and converted to a more appropriate form without risking the host OS.

2.9 Optimizations for performance

Although the idea of server virtualization is to use all resources efficiently, optimizing the virtualization software can enable running a larger number of instances at the same time. The most common optimization methods are reducing the overhead caused by virtualization and sharing similar resources among virtual machines. The virtualization overhead is caused by operations that cannot be executed directly on hardware and by additional mappings that are used to provide the virtual machine a normal environment.

Using the full emulation of the IA-32 architecture causes significant overhead. This is because every hardware call must be verified separately and converted to a suitable system call format for the host OS. Even though kernel modification enables the reduction of hardware related operations, the tracking of system calls generate overhead. Additional

overhead is further caused by the need to forward normal signals from the host OS back to the virtual machines. An alternative approach is to analyze the code and execute insecure instructions as an emulation by the VMM. After a large portion of the code is analyzed and marked safe while the number of insecure instructions is low, overhead, too, is relatively low. To provide an optimal performance, the code of virtual machines should be executed as much as possible on a physical processor. [Dik00], [Rob00].

While virtualization creates the possibility to use resources more efficiently, it does not reduce the amount of required processing power, memory, network or storage capacity to perform a specific task. Depending on the number of virtual machines running concurrently and their similarity of tasks and OSs, resources can be reduced by sharing. This is especially true with disks and memory. If multiple virtual machines are running exactly the same OS, they are also likely to run the same services and use the same libraries. Typically an application uses some memory, whose content is not being changed during execution. Running five copies of the same application means that there are five possible memory areas, whose content will remain intact until the execution of application has ended. Memory usage can be reduced by creating only one copy of that memory and sharing it among applications. [Wal02].

Sharing memory and disk requires different approaches but both of them benefit the most of sharing common operating system components. For example, several virtual machines can be started by using the same system image. When differences occur among the execution of virtual machines (e.g. time stamps on log files), changes compared to the original system image can be created. Disk sharing can thus be eliminated by creating a separate copy of the system image and applying the changes that a single virtual machine had created.

Sharing memory between virtual machines cannot be done directly, since there are no guarantees that a certain memory area contains similar data in each of the virtual machines. Memory content must be identified e.g. by creating hash values. Similarities can be then found by comparing these values of the memory areas between separate virtual machines. While in disk sharing changes are tracked using a separate file, a change in memory requires the creation of a separate copy of the shared area and applying changes. After

changes are applied, the hash value can be recalculated and sharing can occur if areas with similar content are found. In addition to sharing memory between the virtual machines, passing information between applications, drivers, host OS and guest OS using shared memory is generally efficient. [Bug97], [Dik01], [Sug01], [Wal02], [VMw02].

While resource management does not directly provide better performance or decrease overhead, it has become one of the main features especially in commercial products. Traditionally, different operating systems contain resource management that enables fluent execution of the user program and efficient use of hardware resources. A modern OS typically contains a scheduler that shares the execution time of the processor among different processes. Scheduler also changes the order of processes through prioritization. By changing the rank order of processes, the amount of execution time can be adjusted. Even though changing priorities and scheduling processes requires additional resources, the benefits usually exceed the overhead. Virtual machine resource management is a similar concept where virtual machines can have different priorities. Resources that can be prioritized typically contain those parts of the server hardware that contribute to performance most: memory, processor, disk and network.

Prioritizing is easy to implement in the VMM since it handles all hardware related calls. Most commonly used sharing schemes are proportional shares, percentages and explicit values. When virtual machine resources are underutilized, resource sharing does not basically need to interact at all. However, if e.g. managing overall peak load requires more processing power that the hardware can provide, resource management can be used to ensure that the most important virtual machines obtain adequate resources. [VMw02].

Besides saving memory and disk resources, reducing the usage of privileged instructions is the main target in optimization. Other instructions can be run natively with small overhead. Processes that require or produce heavy I/O load create bottlenecks in performance due to context switching. Instead of the I/O performance being the bottleneck, the processor power becomes inadequate to manage the overhead. [Rob00].

Practical optimization approaches are fairly simple. For example, it is possible to gather multiple packets of network traffic together and perform a send or receive process during

one switch. While optimizations can reduce the overhead of virtualization, sharing resources always lead to isolation and security issues. Providing the same performance by virtualization compared to a native environment is very difficult. [Sug01], [VMw02].

3. EFFECTS OF SERVER VIRTUALIZATION

Introducing new technology usually affects the traditional operating environment. Similarly, applying server virtualization causes changes to existing infrastructure, development of new systems, daily maintenance routines as well as unusual situations. To evaluate whether changes are positive or negative, measurements and tracking changes can be done e.g. by practical examination and testing. Based on the results, the difference between two or more environments can be found and virtualization as a solution can be evaluated. The gathered information can then be used further in the creation of the server virtualization strategy.

Effects measurement can be divided roughly into two categories: general differences compared to traditional environment and changes of a single server. While creating accurate tests to measure operational environment and various scenarios is difficult, comparing e.g. the performance of the physical server and the virtual machine is comparatively simple.

3.1 Differences to traditional environment

An environment is called traditional if there is no virtualization applied to it. A traditional environment can be described as “one server, one OS, one application” -concept. When a new application is introduced, server hardware is ordered and after its arrival, the OS is installed. After the OS installation the application itself is installed and configured. At this point the system is ready for production use, and normal maintenance tasks are started (e.g. monitoring hardware and applications, creating backups). [McI03b].

Server virtualization does not change normal tasks such as OS installations, configuring applications and creating backups. It changes the way how these tasks are done. Some tasks, however, remain the same: Examples of this are obtaining server hardware for host system and installing the OS to host. The most obvious change of server virtualization is probably the creation of a new server as a virtual machine instead of obtaining hardware

and installing the necessary software. Major areas where changes occur in virtualization are the following:

- Hardware
- Creating new systems
- Maintenance and troubleshooting
- Backing up
- Planning.

In the following chapters, these areas are examined individually.

3.1.1. Changes in hardware

The most visible change in hardware is the reduced number of hardware instances. The practical result of server virtualization is that every virtual machine reduces the number of physical servers by one. This affects many areas since e.g. the need of hardware monitoring and maintenance is also reduced. In addition to the reduced hardware, every independent virtual machine sees exactly the same hardware. Virtualization software provides the same virtual devices to each virtual machine, thus creating a hardware standard. This feature also enables transfering virtual machines between hosts.

If a single physical server in the traditional environment is lost, usually only a small portion of all applications is affected. A hardware failure in the host system of virtualization environment can disable several applications at once, since virtual machines are relying on the host system and underlying hardware. Due to this, the physical server hardware that is used in building the host system should contain high availability characteristics.

In the traditional environment, physical limitations of hardware (e.g. expendability) are taken into account during purchase planning. Virtualization can then create problems if there are special requirements in the existing environment. For example, if each virtual machine is connected to different network and requires its own physical NIC, there may not be enough expansion slots available in the host system.

3.1.2 Duplicating systems

In server virtualization, provisioning a new server is done by creating a new virtual machine. This operation includes creating configuration file and installing OS to the virtual machine. The whole operation can be performed without any interaction with physical hardware. Installing the OS step by step can be avoided by creating and using template images. At first, a virtual machine is created and the OS installed using basic options. After installation all individual identification information is removed (e.g. network settings) and only the necessary information remains. A copy of the virtual machine is then created by copying its disk and configuration information. This copy can now be used as a template when another, new virtual machine is created. Using this technique, a standard for OS images can be created.

3.1.3 Maintenance

A traditional system maintenance includes monitoring physical hardware, OS and application. In a virtualized environment, the amount of hardware monitoring is decreased, since the number of physical machines is reduced. The number of monitored OSs, however, is increased, since in addition to each virtual machine, the host OS must also be taken account. Because services are produced in a similar way in both traditional and virtual environment, application monitoring remains the same. An entirely new element to monitor is the virtualization software. Since the virtualization software replaces the hardware of virtual machines, monitoring virtualization software can be considered as monitoring the hardware state of virtual machines. The complexity of maintenance and monitoring increases due to the two additional layers in overall structure. Figure 14 presents the overall structure of traditional and virtualized environments as layers. Additional layers in a virtualized environment compared to the traditional one are virtualization software and virtual machine OS. Troubleshooting, too, becomes more complex, since the number of possible error points increases. To make sure that virtualization does not cause errors, a similar environment must be built using physical servers where applications are tested.

The greatest change to daily maintenance routines is that virtual machines do not have physical hardware. In practice, managing and controlling the OS of a virtual machine and its services must be done remotely. While hardware maintenance still exists, the management and maintenance of virtual machines is no longer limited by the requirement of physical access to the hardware.

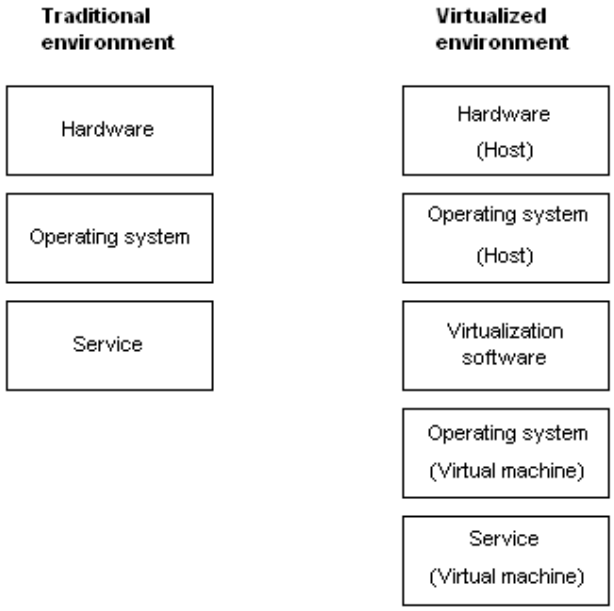

Figure 14 The additional layers of virtualization.

3.1.4 Backing up systems

While backing up the host OS and its application remains the same in virtual environment, backing up virtual machines can be done in two different ways. A virtual machine can be backed up as if it would be a physical system. Transfering a physical machine to a virtual one does not mean that the procedure of creating backups changes. In general, the host OS can also see virtual machines either by files in its file system or as partitions on physical disk. Creating a backup of the virtual machine can thus be done by copying virtual machine disk files or by creating a copy of the partition. Creating these backups by copying files can be referred to as creating virtual machine templates. This operation is the most beneficial in a situation where modifications or changes to the virtual machine are required but performing the operation could damage it. By creating a copy of the virtual

machine before performing the operation, the system can be restored quickly to operational state if the operation fails.

When a failure in physical server hardware occurs, the restoration of the system back to operational state can take a long time. Creating full copies of all virtual machines consume a lot of space and therefore, incremental backups are more common. This restoration process obviously takes more time compared to the traditional environment where only one system would need to be restored. During the restoration phase e.g. available network bandwidth or the number of NICs can become bottlenecks. Therefore, in the planning stage of server virtualization, failover and backup processes should be carefully examined so that services can be restored within a reasonable time span.

3.1.5 Planning

Since server virtualization is used to combine several underutilized servers, appropriate virtualization targets must be selected carefully. The virtualization of existing servers is easier compared to the implementation of a new application, because information about existing server behaviour and resource requirements are available. Capacity planning becomes important especially in the production environment. If in the planning stage it is expected that the application will require e.g. high processing power due to its popularity, implementing the application using a virtual machine should be considered. In the worst scenario, hardware resources are only sufficient to run a single virtual machine. In such cases the benefits of virtualization are practically lost and moving the application to a traditional environment or e.g. to cluster would be a more suitable solution. It is also worth noticing that moving an existing application back and forth between virtualized and traditional environment takes a long time if appropriate tools are not available.

Another issue that can create problems in server virtualization is using it in essential parts of the basic infrastructure without proper backup planning. Since virtual machines are fully dependent on the underlying hardware and OS, a situation where basic services are not available due to host hardware failure is not favorable. An example of this type of this type of scenario is a situation where application used to authenticate the user is installed to a virtual machine. If the virtual machine is not available (e.g. due to host hardware failure),

applications that require authentication are not available for users. In the worst case, the access to host system itself is disabled, since it relies on the virtual machine. Because a single hardware failure in a virtualized environment can affect or disable several applications, additional redundancy should be provided. A simple solution is to configure the application to run on two virtual machines that reside on different physical systems.

The benefits of server virtualization become visible when virtualization is used in the planning stage for testing purposes. Easy and fast provisioning of virtual machines enable building an entire environment without obtaining any additional hardware. While performance measurements cannot be done in the virtualized environment, everything else can be tested. Testing larger changes with virtual machines is easier, since snapshots of them ensure an easy return to the previous situation.

3.2 Changes within single server

In server virtualization, the main change from the single server aspect is the lack of physical hardware compared to traditional environment. Thus e.g. resetting the system by powering it off is replaced by a similar software function. Every operation done to the server is basically performed using a software based remote management system. If physical servers are already managed remotely, the virtualized environment does not introduce any changes to the existing situation. Virtualization does not affect normal operation. When comparing the physical server and virtual machine over remote management system, the only difference is the hardware that the OS sees. A virtual machine sees only those hardware resources that the virtualization software provides to it. Since every virtual machine sees the same hardware, the only way to separate the machines is to examine the basic individual information such as host name and IP address.

3.3 Measuring virtualization effects by tests

In addition to the theoretical point of view, the effects of virtualization can be measured by tests. These tests can be divided into three main categories based on their nature:

- Pure performance
- Operation under different situations
- Environment security and isolation.

Performance tests in a virtualized environment are similar compared to the traditional environment: the goal is to find out the best possible performance level. A typical counter value that is used to represent the performance level is the reading speed of disk or database transactions per second. To support the theoretical viewpoint of virtualization effects, similar operations can be performed both in traditional and virtualized environment.

An example of an operation test could be a situation where the system must be restored from backup. Although pure performance affects the results of operation tests, it also takes into account other essential features such as system management and integration to existing infrastructure. If restoring virtual machine from backup requires special tools or additional phases compared to restoring a traditional system, the difference can be found by using these operation tests.

Environment security and isolation tests measure how well virtualization software can provide isolation and security. The goal here is to confirm that similar security or isolation can be provided in both virtualized and traditional environments. An example of a security and isolation test could be memory protection where the virtual machine is trying to access the memory area that belongs to another virtual machine.

Pure performance tests can be used to measure possible overhead that virtualization causes compared to performance using native hardware. Operation tests, on the other hand, give better overall image of the differences between a traditional and virtualized environment. In addition, performance tests can be used to verify different optimization and resource sharing schemes. Environment security tests are used to ensure that virtualization software is capable of providing similar security and isolation features than separate physical systems in traditional environment. Performing tests in a virtualized environment does not basically differ from testing a traditional environment. The only major difference is that

instead of using several physical systems, a large part of testing can be done in a single physical system.

3.2.1 Test types of performance test

Performance tests can be divided roughly into three different categories based on what part of the system is the main target of the test. The categories are the following:

- Hardware. These tests measure performance that the hardware can provide. The influence of OS is minimized so that only device drivers and necessary parts of OS are used to perform the test.
- Software. The target of testing is the OS and other components whose performance is mainly based on different software solutions.
- Overall performance. This includes both hardware and software specific functions.

Creating a clear distinction between hardware and software category is difficult. While the hardware possesses a certain performance according to specifications and standards, poor software implementation of e.g. device drivers can decrease this performance significantly. An example of a typical hardware test is measuring the disk's reading and writing speed, while typical software test can be e.g. testing memory management or the scheduler of OS. Overall tests usually contain performing a common task that uses both hardware and software resources intensively.

In a typical server hardware, the most essential parts are processor, memory, disk and network connection. Testing performance between a virtualized and traditional environment is simple if the OS can be the same in both the host and virtual machine. Differences are easily discovered by running the same tests in the host without virtualization and in virtual machine. Performance tests combined with statistical data about resource utilization can be used to provide an estimate of how many virtual machines the selected hardware is capable to run fluently. In the planning stage of server virtualization, this information is valuable since selecting suitable hardware configuration becomes easier.

3.2.2 Test types of operation tests

Operational tests contain normal maintenance tasks as well as more complicated and unusual tasks. Since virtualization does not change the basic tasks but the way they are done, time and complexity can vary between a traditional and virtualized environment. To evaluate e.g. skill requirements of maintenance work, knowing the differences between these environments is important.

The following common tasks were identified as subjects to testing:

- A new server is installed
- Processing power is increased (processor, memory)
- Disk space is increased
- Applications are moved to different physical hardware
- The system is restored to operational state after software failure
- The system is restored to operational state after hardware failure
- Server is removed.

An example of a normal maintenance task is managing a situation where the system is running out of disk space. Depending on the situation and system installation, the amount of free disk space can be increased by e.g. deleting unnecessary files, moving files to another file system that has more capacity – or installing larger physical disks to the system. In this example, deleting and moving files does not differ whether the environment is traditional or virtualized. Increasing virtual machine disk space can be done simply by creating a new disk file and moving the existing partition to a new file. This option, however, requires that there is enough physical disk space available in the host system. In case the physical disk space of the host system is insufficient, creating new virtual machine disks can be performed only after the disk space of the host is increased. In a traditional environment, the increment of disk space is always done by increasing the physical disk space.

3.2.3 Test types of environment security and isolation tests

In a traditional environment, each system is physically separated and accessing the system is done by using a local console or network. In a virtualized environment, physical separation between systems does not exist, since the virtual machine is using the memory and disk space of the host system. The most important basic feature of the virtualization software is isolation, which ensures that failure in the virtual machine does not affect the operation of another virtual machine or the host system. Thus, isolation in a virtualized environment could be referred to as OS protection mechanisms that prevent improperly behaving processes to affect the whole OS. Once isolation functions in a proper manner, security issues must be considered. Accessing the information of a virtual machine memory or disk should be protected so that reading or changing their content is not possible without appropriate permissions.

The following basic scenarios can be used to examine security and isolation features of a server virtualization software:

- Reading specific content from virtual machine disk
- Changing the content of virtual machine disk
- Receiving network traffic destined to virtual machine
- Accessing memory and processes beyond the configuration of the virtual machine.

4. TEST SCENARIOS AND RESULTS

To examine the effects of virtualization, various tests were performed by using different server virtualization software. Tests were divided into three different categories based on the tested feature. The purpose of performance tests was to measure differences between a normal system and virtual machine, thus finding out the overhead that virtualization causes. Operational tests included performing similar operations in both traditional and virtualized environment in order to examine how virtualization changes e.g. management work. The goal of security and isolation tests was to verify whether virtualization software can provide a protected environment for virtual machines and whether host hardware can be shared securely.

4.1 Tested server virtualization products

Two different server virtualization products from two different vendors were available for testing purposes: VMware ESX Server and Microsoft Virtual Server. Since both products are commercial and their purpose is to support various user scenarios, they use the ISA replication to create the virtual environment. Therefore both products support running different OSs in virtual machines. The main difference between ESX Server and Virtual Server is the underlying platform and the VMM type. Production versions used in the tests were ESX Server 1.5.2 and Virtual Server 1.0 (pre-beta, build 219). All performance tests were performed using the exact same hardware. It should be noticed that since the tested version of Virtual Server was pre-beta, any assumptions about the final product should not be made based on the tests.

4.1.1 VMware ESX Server

ESX Server is Type I VMM and it uses minimal OS that is based on a modified Red Hat Linux distribution. Minimal OS is called Console OS and it can only see and use limited resources that are configured to it. The main purpose of Console OS is to set up and change the configuration of the virtualization layer and enable remote management capabilities.

Accessing hardware devices is separated between Console OS and the virtualization layer. The virtualization layer has separate device drivers that are based on Linux drivers. The purpose of separate drivers is to provide better performance compared to original drivers and to enable resource management and the quality of service support. The architecture picture of hardware sharing in VMware ESX Server is presented in Figure 15. Virtual machines use the same physical processors as Console OS but they have separate memory area, disk partition and NIC. [VMw02], [VMw03a].

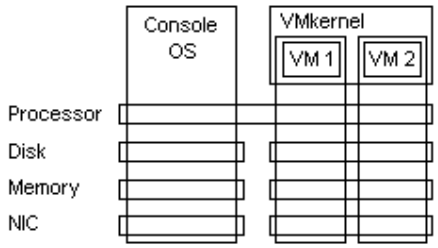

Figure 15 VMware ESX Server architecture.

In the ISA replication, the guests see the underlying hardware as virtual devices that are provided by the virtualization layer. The OS of the guest then uses its own device drivers to access virtual devices. The most essential hardware devices and matching virtual devices that ESX Server provides to virtual machines are presented in Table 1. [VMw03c].

Table 1 Devices and matching virtual device of ESX Server. [VMw03c].

Device	Virtual device
Processor	The same as physical processor
BIOS	PhoenixBIOS 4.0 Release 6
Chipset	Intel 440 BX
SCSI	Mylex (Buslogic) BT-958
Network	AMD PCnet-PCI II
Display adapter	Custom device

A special feature of VMware ESX Server is a content-based memory sharing technique that can be used to reduce the amount of physical memory needed. After the hash value of

the memory block is calculated, the value is compared by using a hash table. This table contains values from those memory areas that are already examined. If the hash value of the examined memory area matches with a value in the hash table, then a memory block with the same content already exists. Memory translation tables of the virtual machine are then updated so that only one memory block with the exactly same content exists. If one of the virtual machines makes changes to the shared memory block, a separate copy of the shared block is created where changes are done. A hash value is then being recalculated and the search process described above is reproduced. [Wal02].

4.1.2 Microsoft Virtual Server

Microsoft Virtual Server is Type II VMM and it runs as a normal application on top of Microsoft Windows OS. A virtualization layer runs as a service on the host OS and the host OS is used to access physical hardware. While virtual machines share hardware with the host OS, it is possible to assign e.g. a physical partition or a physical NIC to virtual machine use only. The main difference between ESX Server and Virtual Server is that all virtual machine execution and data can be seen by the host OS in Virtual Server but not in Console OS of the ESX Server. Virtual Server architecture is presented in Figure 16. [Pro02].

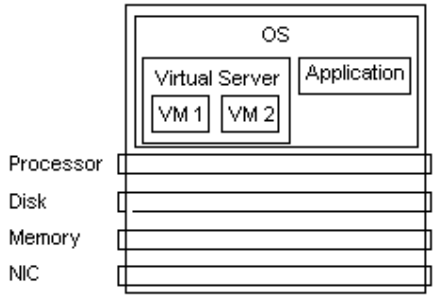

Figure 16 Microsoft Virtual Server architecture.

Similarly to ESX Server, Virtual Server provides hardware resources of the host to virtual machines in the form of virtual devices. Hardware devices and matching virtual devices that the Virtual Server provides to virtual machines are presented in Table 2. The Virtual

Server version used in the tests did not contain virtual device implementation for SCSI and therefore, tests were done using the IDE interface [Pro02].

Table 2 Devices and matching virtual device of Microsoft Virtual Server. [Pro02].

Device	Virtual device
Processor	The same as physical processor
BIOS	Amibios 8.0 based ACPI
Chipset	Intel 440 BX
SCSI	Adaptec 7870
Network	DEC 21140
Display adapter	S3 Trio64 with 4 MB memory

4.2 Performance tests

While the main goal of performance tests is to find out the overhead that the virtualization causes, the results are valuable also in the cases where operational tests are analyzed and server virtualization is planned. For example, the suitable number of virtual machines per host system or preferred virtualization targets are easier to select when performance information is available. The performance tests for this thesis were done using exactly the same hardware for three different configurations. The idea was to provide an identical platform to the test programs and enough resources to the virtualization software for running a single virtual machine. The configurations were the following:

1. A normal system containing Windows 2000 Server with Service Pack 3 as OS. The amount of physical memory was 128 MB. This configuration is later on referred as Native.
2. A virtual machine under VMware ESX Server 1.5.2. The amount of physical memory was 320 MB and 128 MB of it was configured for Console OS. Virtual machine was configured to use 128 MB of memory and its OS was Windows 2000 Server with Service Pack 3. This configuration is later on referred as ESX.
3. A virtual machine under Virtual Server Beta 1.0 (Build 219). The host OS was Windows 2000 Server with Service Pack 3. The amount of physical memory was

320 MB and Virtual machine was configured to use 128 MB of memory. Virtual machine OS was Windows 2000 Server with Service Pack 3. This configuration is later on referred as Virtual Server.

The management of virtual machines were done by using remote management software that the vendor had provided with the virtualization software. To make sure that the network traffic generated by remote management as well as other general traffic would not affect network intensive tests, a separate network card was installed to the system. The settings of the network interface (100 Mbit/s and Full Duplex) were kept the same during all tests. Therefore, the virtual machine would have a similar connection compared to the native configuration. The disk system contained a single physical disk that was shared between the host and virtual machine in Virtual Server and ESX configurations.

During tests the OS was configured with only minimal services required for operation. Each test was iterated five times and average values calculated from the results. These values were then used as the basis for analysis. Running tests multiple times is especially important in disk system tests, if e.g. seeking position is done randomly. Between each run a system reset was performed to ensure that the results would not be affected by e.g. disk or memory cache. Also if a result of a single test differed substantially from the values of previous tests or if the OS performed some background activity that may have affected the result, the same test was repeated to ensure unbiased results. Further on, separate test programs for the same test types were used to verify results.

The following programs were used in testing:

- Iometer, version 2003.05.10
- Netperf
- Passmark Software, PerformanceTest, version 4.0
- SiSoftware, Sandra Standard, version 2003.7.9.73

In addition to test programs that measure a certain part of the system, overall performance was tested separately by executing common operations such as upgrading the system.

4.2.1 Processor

Processor performance was measured by math and MMX tests of PerformanceTest and by CPU arithmetic test of Sandra. The math test measures the processor's capabilities of performing integer and floating point operations such as addition, subtraction, multiplication and division. During each integer test a large array of random 32-bit integer numbers are used in calculations. Similarly in the floating point tests, 32-bit single precision floating point numbers are used. Both integer and float point tests are single threaded and thus suitable for virtualization software that cannot provide SMP capabilities to virtual machines. MegaFLOPS value presents the maximum number of floating point operations per second. In MegaFLOPS, all overhead caused by computation (e.g. looping and branching) is removed and therefore the result is closer to the theoretical value than the practical one. In the integer and floating point tests, the overhead is included in the final value. The MMX test is similar compared to the math test except that it uses 64-bit integers and 64/128-bit floating point numbers. Since the test platform was based on Intel PIII processor that supports SSE instructions, floating point tests were performed using 128-bit numbers. The summary of math test results are provided in Table 3 and MMX test results are provided in Table 4, respectively. [Pas03].

The CPU Arithmetic test of Sandra Standard performs an identical test that the math test of PerformanceTest. Dhrystone provides information about integer performance while Whetstone measures floating point performance. The results of CPU Arithmetic tests are presented in Table 5. [Sis03].

Table 3 The results of PerformanceTest math test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Integer Addition	95,98	65,48	58,70	- 31,78	- 38,84
Integer Subtraction	96,34	66,24	57,00	- 31,24	- 40,83
Integer Multiplication	100,26	67,94	60,28	- 32,24	- 39,88
Integer Division	13,00	11,48	10,98	- 11,69	- 15,54
Floating Point Addition	87,90	61,94	53,60	- 29,53	- 39,02
Floating Point Subtraction	87,88	61,64	53,56	- 29,86	- 39,05
Floating Point Multiplication	88,68	62,58	53,70	- 29,43	- 39,45
Floating Point Division	14,80	13,16	12,38	- 11,08	- 16,35
Maximum Megaflops	123,68	117,32	113,04	- 5,14	- 8,60

Table 4 The results of PerformanceTest MMX test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Addition	154,24	109,28	93,62	- 29,15	- 39,30
Subtraction	154,46	109,6	94,50	- 29,04	- 38,82
Multiplication	148,14	108,92	93,30	- 26,47	- 37,02
SSE/3DNow!	335,12	250,12	221,26	- 25,36	- 33,98

Table 5 The results of Sandra Standard CPU Arithmetic test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Dhrystone ALU	1347,00	1299,20	1206,00	- 3,55	- 10,42
Whetstone FPU	667,80	640,20	598,00	- 4,13	- 10,45

4.2.2 Memory

Memory performance was tested with the Memory Access test of PerformanceTest and with Memory Bandwidth and Cache & Memory tests of the Sandra Standard. Memory Access tests included measuring memory reading, writing and allocation. The allocation test included allocating and then freeing small 100 KB memory blocks with zeroed content. The reading and writing test contained normal memory reading and writing operations. The only difference between a cached and uncached reading test was block size. With a small block size the reading can be done entirely from the cache. In the cached tests the size of the block was made large enough so that the available cache would not be beneficial. The results of Memory Access tests are presented in Table 6. [Pas03].

Table 6 The results of PerformanceTest Memory Access test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Allocation – Small Block	223,54	213,48	193,82	- 4,50	- 13,30
Read – Cached	220,88	205,12	195,94	- 7,14	- 11,29
Read – Uncached	118,34	114,10	98,64	- 3,58	- 16,65
Write	169,96	150,98	141,22	- 11,17	- 16,91

The Memory Bandwidth test of Sandra Standard can be used to measure sustained memory bandwidth. The test uses dynamic data as well as dynamic alignment of the data streams to

select the best possible combination. The test is also designed to use aggressive scheduling and overlapping instructions so that the limiting factor is always memory bandwidth, not the processor. The Cache & Memory test does not use streams but blocks with different sizes instead. Thus different caches and their sizes affect the test result most. The results of Memory Bandwidth tests are presented in Table 7 and the results of Cache & Memory tests in Table 8, respectively. [Sis03].

Table 7 The results of Sandra Standard Memory Bandwidth test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Int Buff iSSE	703,00	708,60	340,00	+ 0,80	- 51,64
Float Buff iSSE	667,2	664,8	384,4	- 0,36	- 47,78

Table 8 The results of Sandra Standard Cache & Memory test.

Test type / Block size	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
2 KB	3380,20	4731,60	3158,80	+ 39,98	- 6,55
4 KB	3543,00	4944,00	3318,20	+ 39,54	- 6,34
8 KB	3627,60	5070,40	3390,20	+ 39,77	- 6,54
16 KB	2917,60	3625,20	2670,60	+ 24,25	- 8,47
32 KB	820,80	834,40	747,20	+ 1,73	- 8,90
64 KB	733,80	745,40	665,80	+ 1,58	- 9,27
128 KB	736,00	747,40	666,00	+ 1,55	- 9,51
256 KB	733,60	744,40	652,60	+ 1,47	- 11,04
512 KB	707,80	664,40	522,00	- 6,13	- 26,25
1 MB	370,60	358,20	333,60	- 3,35	- 9,98
4 MB	348,40	337,40	306,60	- 3,16	-12,00
16 MB	348,80	338,20	295,60	- 3,04	- 15,25
64 MB	346,20	337,60	222,80	- 2,48	- 35,64

4.2.3 Disk

The speed and response times of the disk system were examined using four different test programs: Disk Speed and Advanced Disk Speed of PerformanceTest, File System test of the Sandra Standard and Iometer. Both Disk Speed and File System tests are simple tests that only report basic values such as overall performance. Advanced Disk Speed provides the ability to test disk reading and writing performance using different accessing methods. Iometer, on the other hand, is a highly configurable test program that provides the ability to e.g. create separate tests for different usage scenarios. [Iom03], [Pas03], [Sis03].

The Disk Speed test measures the disk performance of the system by using three different tests. These tests include sequential writing and reading test where a large data file is first created and the read from the beginning to the end. Each individual writing and reading operation in these two tests is performed using 16 KB data size and the cache. The third test is a combination of random writing and reading operations while using the cache. The test contains creating a large data file, seeking random position within data file and performing reading or writing operation using 16 KB block size. After the operation is performed, a new seek is started. Using these three tests, the performance of the disk system in both sequential and random cases can be measured. The results of PerformanceTest Disk Speed test are presented in Table 9. [Pas03].

Table 9 The results of PerformanceTest Disk Speed test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Sequential Write	4,82	4,00	2,16	- 17,01	- 55,19
Sequential Read	7,66	6,54	5,32	-14,62	- 30,55
Random seek + Read/Write	2,6	2,36	1,5	- 9,23	- 42,31

The Advanced Disk test enables testing disk writing and reading speed separately by using C/C++ function calls (fwrite and fread functions) with the cache enabled or by using standard Win32 API functions (WriteFile and ReadFile functions) with and without the cache enabled. During the test, a temporary file is created and all operations performed within that file. The file and used data block size can be adjusted separately. A default file size of 127 MB and block size of 16 KB were used in the tests. The results of PerformanceTest Advanced Disk Speed test are presented in Table 10. [Pas03].

Table 10 The results of PerformanceTest Advanced Disk Speed test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
C/C++ API, Write	5,02	4,08	3,35	- 18,76	- 33,25
C/C++ API, Read	7,55	6,55	5,33	- 13,28	- 29,33
Win32 API (Cached), Write	5,01	4,11	3,38	- 18,11	- 32,67
Win32 API (Cached), Read	7,52	6,51	5,31	- 13,41	- 29,30
Win32 API (Uncached), Write	5,87	5,05	3,60	- 14,01	- 38,65
Win32 API (Uncached), Read	7,41	6,73	4,71	- 9,12	- 36,35
Raw Disk Read	12,67	10,24	18,81	- 19,22	+ 48,44

Sandra Standard File System test uses various tests to provide a single value that represents the performance of a file system. The test includes measuring the three most common areas of the disk system: reading, writing and seeking. In reading and writing tests, the

performance is tested separately by using buffer, random and sequential operations. Similarly compared to the other disk tests, the operations in File System test is performed within a single file. The file size is equal to the size of the system memory. Table 11 contains the results of File System test. [Sis03].

Table 11 The results of Sandra Standard File System test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Drive Index	6752,80	6358,20	3972,00	- 5,84	- 41,18

Iometer was used to create three different tests, each providing information about the usage scenario. Two of the tests are related to backing up and restoring the system. The backup test simulates creating a system backup to a tape drive with 64 KB block size. The restoration test is performed in reverse order compared to the backup test. Both writing and reading are performed sequentially and all operations are performed within a single 500 MB file. In addition to speed, the number of performed I/O operations per second and the average response time are measured. The results of the backup and restore tests are presented in Table 12.

Table 12 The results of Iometer Backup and Restore tests.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Read IO/s	133,40	110,06	64,27	- 17,50	- 51,28
Read MB/s	8,46	6,88	4,02	- 18,70	- 52,54
Average Response Time (ms)	7,38	9,07	15,49	- 22,90	- 102,92
Write IO/s	121,71	99,74	60,85	- 18,05	- 50,01
Write MB/s	7,61	6,24	3,80	- 18,01	- 49,99
Average Response Time (ms)	8,21	10,01	16,37	- 21,95	- 99,37

The third Iometer test scenario was called file sharing. The purpose of the test was to simulate the usage of file server by performing reading and writing operations in random order with different file sizes. While the read/write ratio was set to 60/40 ratio, the order of operations was random. Four different file sizes were used, each configured with different access portion. The smallest file size was set to 100 KB with a 40 % portion. The largest file size was 4 MB with a 10 % portion. The two other file sizes were 1 MB and 2 MB with 30 % and 20 % portions. Due to the random nature of the test, the results will always contain some variation. In addition to the overall amount of I/O operations and transfer rate per second, the read and write portions are also listed in the following table. The overall values are merely a sum of write and read portions. Table 13 contains the results of file sharing tests.

Table 13 The results of Iometer File Sharing tests.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
IO/s	6,26	6,23	4,17	- 0,38	- 33,34
Write IO/s	2,58	2,47	1,63	- 4,34	- 36,90
Read IO/s	3,68	3,76	2,54	+ 2,34	- 30,85
MB/s	7,15	7,07	4,73	- 1,17	- 33,85
Write MB/s	2,98	2,77	1,79	- 6,79	- 39,72
Read MB/s	4,17	4,23	2,92	+ 1,34	- 30,14
Average Response Time (ms)	159,74	160,33	239,54	- 0,37	- 49,96
Average Response Time (ms), Write	181,16	166,46	257,27	+ 8,11	- 42,11
Average Response Time (ms), Read	146,68	158,20	228,21	- 7,85	- 55,58

4.2.4 Network

Network performance was measured with two similar tests: PerformanceTest Network Bandwidth test and Netperf. Both tests contain two separate parts: measuring bandwidth by both sending and receiving data. In the test results, server mode means that data is being received by the virtual machine. Client mode, on the other hand, means that data is being sent by the virtual machine. During the Network Bandwidth test, the default data block size of 4096 bytes was used and the test time set to 60 seconds. Netperf's default test is TCP Stream test that opens a TCP connection and then sends variable size data over the established connection. The Network Bandwidth test presents results as bytes transmitted per second, while the Netperf results are presented as a throughput of 10^6 bytes per second. The Network Bandwidth results are presented in Table 14 and Netperf results in Table 15. [Net03], [Pas03].

Table 14 The results of PerformanceTest Network Bandwidth test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Server mode	53806,60	50477,20	34967,00	- 6,19	- 35,01
Client mode	36149,8	19997,60	14687,00	- 44,68	- 59,37

Table 15 The results of Netperf test.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Server mode	94,58	72,55	44,47	- 23,29	- 52,98
Client mode	94,91	43,49	30,68	- 53,94	- 67,50

4.2.5 Overall performance

Making the difference between pure hardware and software tests is difficult, since measuring hardware performance means typically measuring virtualization overhead. Virtualization overhead, however, is based on the software implementation of virtualization platform and thus measuring it means measuring software performance. In addition to testing specific areas of virtualization in this thesis (processor, memory,

network and disk), more general tasks were performed and execution times measured for comparison. The tasks included increasing the service pack level of the operating system, compiling software and examining the performance of two commonly used server software.

The first test consisted of installing Service Pack 4 to a system that was running Windows 2000 Server operating system with Service Pack 3 installed. The second test measured the compilation time of OpenSSL 0.9.7b software package. This compilation was done using Borland C++ 5.5 compiler. Installing the new service pack and compiling software uses both processor, memory and disk resources intensively. In addition, OpenSSL was used to create hash values from a binary file with the size of 127 MB. The hash values were calculated using five different algorithms including MD2, MD4, MD5, RMD160 and SHA1. The results of these tests are presented in Table 16. [Ope02].

Especially network and disk performance were examined by installing ftp-server software and uploading a 127 MB binary file. The general server performance was also tested by installing www-server software and generating a load with Microsoft Web Application Stress Tool. The Stress Tool program was configured to create a new connection to server every 20 ms and load a web page that caused either dynamic content generation or transmitting the static content. The access scheme was set to 50 % for content generation and 50 % for static content. During the test, the processor usage was monitored and the average value recorded. The results of these tests are presented in Table 16. [Mic03].

Table 16 Performance under general tasks.

Test type	Native	ESX	Virtual Server	ESX (%)	Virtual Server (%)
Installation of Service Pack 4	13:01	19:20	22:05	- 32,67	- 41,06
Compiling OpenSSL 0.9.7b	4:51	11:01	14:53	- 55,98	- 67,41
Hash calculation	3:54	4:03	4:28	- 3,70	-12,69
File upload	0:23	0:35	0:49	- 34,29	- 53,06
Web Application Stress	37,8 %	44,3 %	64,4 %	- 14,67	- 41,30

4.3 Operational tests

Several operational tests were performed in different hardware configurations to examine the differences between a traditional and virtualized environment. The focus of these operational tests was to see how operating a virtual machine differs from traditional servers and whether virtualization creates any limitations compared to the traditional environment. The tests were performed using with both ESX and Virtual Server software to enable comparison of the products.

4.3.1 Installing new server

Installing a new server as a virtual machine can be done in two different ways: either by using normal installation method or by using system images. Normal installation contains partitioning hard disk and creating a file system, copying necessary files, installing required drivers for different devices and preparing the OS so that after installation, the system is fully operational. System image is a snapshot of a working system, usually taken directly after the normal installation has ended. Instead of going through all steps of the normal installation process, only the system image is transferred and necessary identification information is changed. After a virtual machine is installed using normal process, the system image can be created simply by copying the virtual machine's image file.

The main difference between a normal installation and an image file process is the time consumed in the installation process. If a virtual machine is created using the image file, the only time-consuming action is creating a copy of the existing image file. The normal process, on the other hand, usually lasts much longer, since installation operations are done in the virtual machine instead of the native hardware. During the tests, both virtualization products were able to use the existing system images as a base of virtual machine. There were no errors or problems in this area. Installing a virtual machine using normal process resulted in no exceptions, either.

4.3.2 Increasing processing power

There are several ways to increase the processing power of a single virtual machine:

- Acquiring optimizations to the virtualization software that reduce the overhead
- Adjusting priority between virtual machines within different resource areas
- Removing boundaries or limitations from the virtualization software that disable virtual machine scaling
- Optimizing the performance of the host system
- Increasing the physical processing power of the host system.

Increasing processing power through different optimizations provides usually only small additions to the existing performance level. The idea of optimization is to find a new way to do the existing task or operation by using fewer resources than previously. The typical areas of improvement are network throughput and disk transfer speed, for example. While optimizations increase the performance, they should not be the key element or way to improve performance.

Adjusting prioritities between virtual machines can be used to gain more resources to the selected virtual machine. Priority adjustments affect only when the capacity of the virtual machine's host system is exceeded. In other situations, there are enough resources for every virtual machine. The virtualization software can create limitations to e.g. memory size that the single virtual machine can use.

In this thesis, prioritizing was tested by running equal loads on two separate virtual machines and using two different priority settings. In the first test setting, both virtual machines had similar access rights and priorities to the resources of the host system. During the second test, the first virtual machine was preferred over the other. The overall test result showed that the priority adjustment had an effect and the first virtual machine was able to perform the same operations faster than the second virtual machine.

Increasing the processing power of a host system can be seen as the most efficient way to increase the overall processing power. If the virtualization software does not cause any

limitations, the increased power can be fully utilized by virtual machines. After physical changes to the hardware is performed, the host OS must be able to detect changes. The tested products were both capable of dealing with the changed hardware when it was supported by the vendor. This means that the virtualization does not imply any problems in increasing the processing power of a host system. Instead, the main challenge discovered in the testing was enabling SMP support when the system previously had only single processor.

4.3.3 Increasing disk space

Increasing disk space of a virtual machine consists of two parts: ensuring that the host file system has enough capacity available and creating a new system image file for the virtual machine. If the host file system does not have enough capacity, it must be increased before a new image file can be created. Once the new system image has been created, virtual machine sees it as a normal physical drive. This process was tested by increasing the size of the system partitition in a virtual machine. The test was performed using common tools of a traditional environment. The results imply that there were no differences between a virtual and a traditional environment in increasing disk space.

4.3.4 Transfering applications to different physical hardware

The operation of transfering applications does not change when virtual machines are used. Since applications generally use standardized APIs or system calls of the OS, the impact of the changed hardware is small. The main difference in a virtualized environment is that the hardware that the virtual machine sees remains the same after the initial transition to a virtualized environment is performed. Thus, the main operational benefit is the standardized hardware environment. This means that the virtual machine image files can be copied to another physical host system without any changes needed.

4.3.5 Restoring system to operational state after software failure

When an application causes failure, usually the OS can continue in an operational state and manage the effects caused by the failure. Restarting the application and verifying data

integrity are typically the only operations needed to restore an application. If the data becomes corrupted or the application is unable to start, the possible solutions are restoring the data from backups or a reinstallation of the application. When an application is running on virtual machine, these operations are exactly the same as in a traditional environment. If the OS is unable to handle application failure or if a part of the OS itself causes failure, the OS usually detects the error and changes to a state where the system cannot be used without the necessary repair operations. Depending on the failure type, resetting the system could be the only operation needed. If a failure causes damages to the basic settings or files of the OS, it might be necessary to reinstall the whole OS and applications, and restore data from backups.

4.3.6 Restoring system to operational state after hardware failure

A hardware failure of the host system affects also virtual machines. The most sensitive parts to a hardware failure are the data that resides in a hard disk and the data that has not yet been written to the disk. The virtual machine has its vulnerable parts, too. The most sensitive ones are system image files, since a failure in an image file can be referred to as a physical error in a hard disk. Because the file system of a virtual machine resides typically within a large file on top of another file system, the probability of a system image file corruption increases.

When the recovery from a hardware failure was compared in both traditional and virtual environment, the bottleneck was found in different places. In a traditional environment, restoring process stresses heavily on the network and backup system. This applies especially to those situations where several machines are being restored at the same time. In a virtualized environment, the processing power of the host becomes quickly the limiting factor. When simulated restoring tests were performed to two virtual machines concurrently, nearly 80 % of the host's CPU resources were used. Therefore restoring a host system that contains several virtual machines takes significantly longer time compared to restoring a single system in traditional environment.

4.3.7 Removing server

Virtual machine related information usually consists of two parts: configuration file and system images. The configuration file contains common information related to a single virtual machine such as memory size, disk images and network setup. System images, on the other hand, contain virtual machine disk partitions. If a virtual machine needs to be removed permanently, the only operation required is removing the configuration and system image files. A temporary removal can be done simply by shutting down the virtual machine and ensuring that the necessary information is backed up.

4.4 Security and isolation tests

Security and isolation tests were performed to find out whether virtualization changes any security aspects and whether virtualization software can provide isolation between virtual machines. The main goal of the tests was to examine situations where the same physical resource is being shared between virtual machines and between a virtual machine and a host. Three different areas were under examination: disk image files, network traffic and the isolation of memory and processes.

4.4.1 Modifying and accessing disk image files

A common method for installing virtual machine is to use disk image file. This image file is seen by the virtual machine as a physical disk while the host file system sees it as a normal file. The arrangement creates a possibility to use normal file operations such as reading and modifying to the image file. In addition, the behaviour of a virtual machine under corrupted system image file can be tested by modifying the image file from the host file system.

The image file reading capability was tested simply by saving a special text string to a file that resides within a virtual machine. This means that the text string will be saved within a system image file that is located in the host file system. The image file can then be altered to a common string using searching programs to examine whether the content is readable.

A point worth noting is that in order to perform this test, the format of the image file does not need to be known.

The result of the test was that both ESX and Virtual Server disk image content was readable and the string saved within a file was found. However, modifying the image file is a much more complicated task if the file format is unknown. Tools such as the Virtual Disk Driver for VMware file format can be used to mount the image file as a physical disk. This solution enables the reading and writing of an image file as if it would be a normal partition in the system. [Kat03].

Managing image file corruption was tested by modifying a file and inserting additional bytes to it. The changes were done to a fully working virtual machine's image file and after the modification, the original image file was replaced with the modified version. To make sure that the images files differed from another, a hash value was calculated from the image file before and after modification. The virtual machine was then restarted using the modified file and effects observed. In all test situations, even the smallest change in image file caused either the virtual machine software to report an on about a broken file or the virtual machine was unable to start correctly.

4.4.2 Examining network traffic

The basic feature of a network based on Ethernet switch is that once the switch knows which system is behind each port, it can make a clear selection on how traffic should be routed. Therefore, a single system cannot receive any traffic that is destined to another system if both systems reside in different ports of the switch. Virtual machines change this concept since in the common setup situation, two or more virtual machines share the same physical NIC. In this case, a separate MAC address is generated to each virtual machine and the physical NIC is placed into a mode that enables receiving and sending packets for different MAC addresses.

Testing network traffic included two parts: examining traffic between a virtual machine and a physical host and between two virtual machines in the same physical host. The tests included opening common TCP connections between the examined system and another

physical system that resides within the same LAN segment. During the connection establishment, the network traffic was captured and then analyzed.

If a virtual machine cannot see or capture traffic of another virtual machine, the network connection can be seen as a normal switched environment. ESX and Virtual Server have different approaches due to their architectures, since the network interfaces in ESX for the host and virtual machines are separated and the host system cannot see the physical NIC destined to the virtual machines. Virtual Server, on the other hand, can use any NIC that the host system can provide. Thus the network traffic of the virtual machines that reside on ESX cannot be examined directly from the host system, while in Virtual Server it is possible. This feature was also tested by installing Ethereal – a program capable of capturing network packets – to the host OS of the Virtual Server and using virtual machine resources remotely. In the test, Ethereal was used to capture all traffic that passed through the NIC and was dedicated to the virtual machines. An analysis of the captured packets revealed that the host system can be used to capture the network traffic of a virtual machine. [Eth03].

The second, more important part of the tests was to examine the possibility to see or capture network traffic from a virtual machine while the actual destination or source is another virtual machine in the same host system. Figure 17 represents the test environment that contained two separate physical computers, the other to host the virtualization software and the other to establish a connection to the virtual machine. Two virtual machines were running under the virtualization software (VM 1 and VM2), the first being used as a connection target and the other for capturing and monitoring traffic. The physical network was based on switched Ethernet. The host system of the virtual machines contained two NICs, one dedicated to be used by the two virtual machines and another for the host system. Two separate network monitoring and capturing software, Ethereal and Ettercap, were installed to the virtual machine. Ethereal was used to capture packets from the network while Ettercap was used to examine whether the network connection to the virtual machine acts as a switched network. [Eth03], [Ett03].

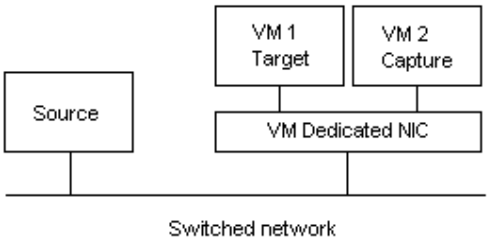

Figure 17 Test environment for examining network traffic.

The test results differed between ESX Server and Virtual Server. The ESX Server provided a network connection similar to the switched network for virtual machines. It was not possible to capture or see the traffic of VM 1 from VM 2 using normal capturing techniques. A more sophisticated method called ARP Poisoning was used to make sure that VM 1 and the Source computer had similar (switched) connections. When the same tests were performed under Virtual Server, VM 2 was capable of seeing and capturing all traffic related to VM 1. Thus the network connections of virtual machines running under Virtual Server should not be considered as switched.

4.4.3 Process and memory isolation

The isolation of memory and processes is one of the most important features of a modern OS. This feature can guarantee that a badly behaving process does not affect the operation of other processes or the operating system itself. Isolation in a virtualized environment is performed by the virtualization software. Together with the host OS, the virtualization software is responsible for providing a safe environment for virtual machines.

The memory and process isolation was tested by examining the resources that a virtual machine can see and by creating failures and memory leaks within the virtual machine. The same OS was installed to native hardware and virtual machines, and the resources were compared against one another. The tests showed that the virtual machine can only see devices and memory areas that the virtualization software provides to it.

Failures were created by removing essential parts of the guest OS while it was running. Also in Virtual Server, the host resources that the virtualization software process used (e.g. file handles, threads) were either modified or removed. Two virtual machines were used in

the test: the other was crashed and the other remained intact for monitoring purposes. When the first virtual machine crashed due to the OS failure, the second virtual machine was not affected by it, which goes to show that the isolation between virtual machines functions well.

Memory protection was tested by running a program that reserves memory but does not release it. After the program had operated for a while, all memory available was used and the OS of the virtual machine started swapping intensively to provide more memory resources. When the limit of the swap file size was achieved, the OS informed that the system was running out of memory. At this point, the memory usage was examined from both the virtualization program and the host OS. In Virtual Server, the results showed that the first virtual machine had only used the memory that was configured to it. In ESX Server, the memory sharing feature had detected similar memory contents in large areas and enabled each virtual machine to use memory over the limit configured to them.

5. DISCUSSION

When the performance of Virtual Server is compared to either native environment or ESX Server, it should be noticed that the Virtual Server version used in the tests is a pre-beta one. Therefore, the results of Virtual Server tests should not be used evaluate the final product. ESX Server, however, is a product that has been publicly available for a few years.

5.1 Performance

Performance tests revealed that if the virtualization process is simple and straightforward, the overhead is usually small. This can be clearly seen when the results of memory and disk tests in a virtualized environment are compared to a traditional environment. In general, memory and disk operations of a virtual machine running in ESX Server are 5 - 20 % less in performance compared to the native configuration. With Virtual Server, the results have a similar pattern except that the performance of the disk is even worse than with ESX. One reason for the bad disk performance is missing SCSI support. The results from both file upload and hash calculations confirm the conclusion, since the tests include performing I/O operations in memory and disk. The greater difference in the upload result between virtual machines and native configuration is caused by lower performance in network traffic handling.

There were two test results in memory and disk tests that caused confusion: the ESX Server results of the Sandra Standard Cache and Memory test, and the Virtual Server result of the PerformanceTest's Raw Disk Read. The results of these tests indicated that the virtual machine outperforms the native configuration. There was no distinct reason found to this behaviour within the available time span.

As stated previously in this thesis, processor virtualization is a complex process. In order to share a physical processor, state information must be kept correct and unsecured instructions performed on software. These requirements can be seen clearly in both

mathematical and compilation tests. Performing basic mathematical operations is roughly 10 - 40 % slower on a virtual machine than on a normal system. The compilation test that uses most of the processor's time and a large variation of its instructions shows that the overall performance of a virtual machine depends highly on the task type. Compiling the same software in virtual machine running on ESX Server takes over two times longer than compilation in the native system. With Virtual Server, the results are similar but due to larger overhead in many areas, the overall result is significantly lower compared to the ESX Server. When considered that the compilation uses memory and local disk that have lower overhead, processor virtualization can be seen to cause the largest portion of the overhead.

Sending and receiving data over a network connection requires additional phases in a virtualized environment, since the communication to the network is done using a physical NIC of the host system. The overhead caused by additional phases can be seen clearly in the results of the network performance tests. Based on the tests, receiving data causes smaller overhead than sending it. One reason for the smaller output in sending data is using a system call when data is sent from the VMM to the host driver. In receiving data, the same process can be done by copying data directly to the memory region of the virtual machine's device driver and by notifying the guest OS about the received data. Also using fixed size data blocks in sending and receiving causes smaller overhead compared to the variable size option. Because the amount of data sent over network varies constantly in normal operation, the overhead in variable size data transmission can be seen as more practical than with the fixed size. The network performance of a virtual machine running under ESX Server is roughly 5 to 55 % lower than the same configuration in the native environment. With Virtual Server, the performance is roughly 35 to 70 % lower compared to the native environment.

The overall performance of both ESX Server and Virtual Server can be described as in the following: Using a virtual machine to perform resource intensive tasks is not reasonable due to the overhead. This does not mean, however, that the virtual machine could not be used to perform the same operations as normal systems. Generally virtualization can be seen to reduce the performance by one third. Although comparing the performance of the tested virtualization products may seem simple based on the results presented here, the

overall performance is always a sum of many components. One part that affects the performance is device drivers used by the OS. If a virtual device differs in two virtualization products, also the driver used by the virtual machine OS is different. This difference, then, affects directly the performance of the virtual machine.

The most important area where the effect of lower performance should be examined is the production environment. After virtualization overhead and peak reserve are removed from the overall system resources, the number of virtual machines in a single host system must be decided. These machines will then run on the remaining resources. The higher the number of machines, the more beneficial virtualization can become. In areas such as testing environments where there are no high performance requirements, server virtualization is the most suitable solution.

5.2 Operational changes

Operational tests revealed that there are three separate levels where operations are performed: host, virtualization software and virtual machine level. The host level includes managing hardware and the host OS, i.e. issues that affect each virtual machine. The virtualization software level contains managing virtual machine resources and the virtual machine level contains operations that are destined to a single virtual machine only. The main reason for different levels is the layered structure where each part requires separate management.

Depending on the operation or maintenance tasks, a single task can require performing actions in all three levels. A good example of this is a situation where increasing disk space of the virtual machine requires increasing disk space also in the host level. The process requires adding one or more physical disks and creating a file system in the host level, creating new image file and assigning it to the virtual machine at the virtualization software level and finally, switching the existing system image file to a new one at the virtual machine level. If an increase in physical space is not required, host level operations are not required either. Due to variation in task sizes depending on the current situation, direct comparison between a virtual and a traditional environment is difficult. The obvious

fact, however, is that additional levels make virtual environment more complex than a traditional one.

In addition to tests presented above, a separate operational area was briefly examined. It consisted of migration processes that are generally used when transformation between a traditional and virtualized environment is performed. An example of transformation is moving existing physical machine to a virtual machine. The opposite process, moving virtual machine to a physical machine, is used if e.g. virtualized environment did not met the expectations that was destined to it. Due to the fact that server virtualization is a rather new technology, proper tools to assist the migration processes have been introduced only recently. Although the migration itself can be performed using the same tools as in the traditional environment, a large amount of manual work is required. The tests in this area contained migrating some existing physical machines to virtual machines. The overall time consumed in the migration process was significantly longer compared to creating a virtual machine with identical OS and applications from scratch.

Selecting a suitable host system is one of the key factors when task sizes in a virtualized environment are minimized. When changes to the host system are kept minimal, different tasks affect only virtualization software and virtual machine levels. Since both these levels are based on software solutions, they can be fully administered using remote management software.

5.3 Security and isolation

The security and isolation tests showed that there are two important aspects that must be considered in server virtualization: providing an environment to the virtual machine that corresponds traditional environment, and isolating virtual machines from the host system. During the test period, no such situation could be generated where the virtual machine behaviour would have had an impact on the host system.

If the Virtual Server network connection is not taken into account, the isolation worked flawlessly between virtual machines. However, the influence and importance of the host

system was clearly seen in Virtual Server, since it was running on top of normal OS. All threads, file handles and used locks were visible under virtualization software process and available for e.g. deletion. Network traffic destined to virtual machines passed through a network adapter that was configured and fully visible in the host OS. Configuration allowed the capturing of traffic both from the host and another virtual machine.

The architectural solution of the ESX Server to separate the management part shows its advantages in security and isolation areas. Obtaining information about a single virtual machine is more difficult than with Virtual Server. Separated memory spaces of Console OS and virtual machines ensure that accessing the virtual machine memory region through Console OS is not possible. Similarly, the separate NIC that is used to provide network connection to virtual machines cannot be accessed from Console OS. In practice this means that the Console OS cannot be used to manipulate or listen to the network traffic of virtual machines. While the operation of the virtual machines cannot be directly affected by the Console OS, the system is still used to manage virtual machines. If an access to the Console OS can be obtained by using e.g. security hole in remote management program, virtual machines can be powered off and the system image files accessed.

5.4 Future of server virtualization

The Interest towards virtualization in the IA-32 architecture has increased steadily within the past few years. This can be seen from the number of different virtualization solutions that are currently available. The solutions have also matured to a level where using virtualization in a production environment can be considered.

In addition to currently available software-based virtualization, there have been first signs of enabling virtualization in hardware level. During Intel Developer Forum in Fall 2003, a new technology named Vanderpool was introduced and demonstrated. The technology is a part of the processor and enables hardware-based virtualization, thus using multiple OS concurrently. [Ote03].

6. CONCLUSIONS

The performed tests indicate that virtualization causes a reduction in the overall performance and it creates an environment that is more complex to manage. On the other hand, virtualization enables creating an entire environment with a small number of physical hardware.

The results of the performance tests indicate that the additional operations required by virtualization can be verified using test programs. The amount of additional operations and their type affect directly to performance in the form of overhead. In memory and disk areas, where the virtualization process was straightforward, the test results indicate that the amount of overhead is small. Processor and network virtualization, on the other hand, was more complex and therefore the overhead is larger. When the overall performance of a virtual machine running in VMware ESX Server is compared to a normal system, the virtualization causes roughly a decrease of one third in performance. With Microsoft Virtual Server the decrease is nearly 50 %.

The management of virtualized and traditional environment was compared by operational tests. These tests included common operations from a traditional environment. The results indicate that the amount of management work varies in a virtualized environment. When virtualization is performed after a careful planning phase, the operation sizes and thus management work can be minimized. Despite this, virtualized environment was found out to be more complex to manage than the traditional one. One of the main benefits in virtualized environment is the flexibility of management: All management operations to the virtual machines can be done using remote management software.

The results of isolation and security tests indicate that virtualization depends highly on the host system and the virtualization software. Based on the tests, both ESX Server and Virtual Server are capable of providing isolation between virtual machines. The isolation between the host system and virtual machine, however, is better in ESX Server. The reason to this is based on the design of ESX Server that separates the host and virtual machines efficiently. When the security of the virtualized environment was tested, the results show a

clear difference between the traditional and virtualized environment in one specific area: system image files. Due to this, virtualization is not a suitable solution if sensitive information is handled.

While server virtualization can be used in production environments with certain restrictions, it is the most convenient solution in test environments. Using virtual machines a large change can be easily tested or an entire environment can be created with a single physical machine. Even though the performance is not as good as in a traditional environment, the same operations can be performed in a virtualized environment.

REFERENCES

[Bug97] Bugnion, Edouard. Devine, Scott. Rosenblum, Mendel. Disco: Running Commodity Operating Systems on Scalable Multiprocessors. Proceedings of the 16th Symposium on Operating Systems Principles (SOPS). [pdf-document]. October 1997. [retrieved July 14, 2003]. From: http://citeseer.nj.nec.com/rd/37462260%2C30732%2C1%2C0.25%2CDownload/http://citeseer.nj.nec.com/cache/papers/cs/2623/http:zSzzSzwww-flash.stanford.eduzSz%7EbugnionzSzdisco-tocs.pdf/bugnion97disco.pdf

[Cog02] Cognizant Technology Solutions. SOS Solutions. Server Consolidation. [pdf-document]. 2002. [retrieved June 13, 2003]. From: http://www.cognizant.com/wizards/Thpapers/ServerConsolidation.pdf

[Con03] Connectix Corporation. Virtual Server: Product Overview. [www-document]. 2003. [retrieved August 4, 2003]. From: http://www.connectix.com/products/vs.html

[Day02] Day, Brad. Server Workload Consolidation, Part 2 – Evaluating Unix Workload Management. Giga Information Group, Inc. [pdf-document]. August 21, 2002. [retrieved June 26, 2003]. From: http://www.hp.com/products1/unix/operating/manageability/partitions/library/server_workload.pdf

[Dik00] Dike, Jeff. A user-mode port of the Linux kernel. [www-document]. 2000. [retrieved June 24, 2003]. From: http://user-mode-linux.sourceforge.net/als2000/index.html

[Dik01] Dike, Jeff. User-mode Linux. [www-document]. 2001. [retrieved June 25, 2003]. From: http://user-mode-linux.sourceforge.net/als2001/index.html

[Eth03] Ethereal: Free network protocol analyzer for Unix and Windows. [www-document]. September 10, 2003. [retrieved September 17, 2003]. From: http://www.ethereal.com/

[Ett03] Ettercap: Multipurpose sniffer/interceptor/logger for switched LAN. [www-document]. July 11, 2003. [retrieved September 17, 2003]. From: http://ettercap.sourceforge.net

[Hew03] Hewlett-Packard Company. HP Virtualization. [pdf-document]. 2003. [retrieved June 6, 2003]. From: anonymous ftp: ftp.hp.com/pub/enterprise/Virtualization_5-5-03.pdf

[Int01] International Business Machines. White paper: Partitioning for the IBM @server pSeries 690 System. [pdf-document]. October, 2001. [retrieved June 9, 2003]. From: http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/lpar.pdf

[Int02] International Business Machines, Server Consolidation with the IBM @server xSeries 440 and VMware ESX Server. November 2002. ISBN 0738427330

[Int03a] Intel Corporation. IA-32 Intel Architecture Software Developer's Manual. Volume 1: Basic Architecture. [pdf-document]. 2003. [retrieved July 10, 2003]. From: http://www.intel.com/design/pentium4/manuals/24547012.pdf

[Int03b] Intel Corporation. IA-32 Intel Architecture Software Developer's Manual. Volume 2: Instruction Set Preference. [pdf-document]. 2003. [retrieved July 10, 2003]. From: http://developer.intel.com/design/pentium4/manuals/24547112.pdf

[Iom03] Iometer. I/O subsystem measurement and characterization tool for single and clustered systems. [www-document]. 2003. [retrieved September 5, 2003]. From: http://iometer.sourceforge.net/

[Kat03] Kato, Ken. Virtual Disk Driver Version 2. [www-document]. July 30, 2003. [retrieved September 15, 2003]. From: http://chitchat.at.infoseek.co.jp/vmware/vdk.html

[Koz01] Kozierok, Charles M. The PC Guide – System Resouces. [www-document]. April 17, 2001. [retrieved July 15, 2003]. From: http://www.pcguide.com/ref/mbsys/res/index.htm

[Law99] Lawton, Kevin. Running multiple operating systems concurrently on an IA32 PC using virtualization techniques. [www-document]. June 26, 1999. [retrieved June 24, 2003]. From: http://os.inf.tu-dresden.de/~jn4/diplom/lawton.txt

[Law03] Lawton, Kevin. The Plex86 x86 Virtual Machine Project. [www-document]. 2003. [retrieved June 24, 2003] From: http://plex86.sourceforge.net

[McI03a] McIsaac, Kevin. Intel Server Consolidation: Part 1 – Virtualization. Meta Group. February 20, 2003.

[McI03b] McIsaac, Kevin. Intel Server Virtualization: Part 2 – Picking the Low-Hanging Fruit. Meta Group. February 20, 2003.

[Mic02] Microsoft Corporation. Microsoft Virtual Server: Administrator's Guide. Emulated Hardware of Virtual Server. [www-document].

[Mic03] Microsoft Corporation. Microsoft Web Application Stress Tool. [www-document]. 2003. [retrieved September 15, 2003]. From: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/intranet/downloads/webstres.asp

[Net03] Netperf. Network benchmarking software. [www-document]. 2003. [retrieved September 5, 2003]. From: http://www.netperf.org/

[Ope02] OpenSSL: The Open Source toolkit for SSL/TLS. [www-document]. 2002. [retrieved September 8, 2003]. From: http://www.openssl.org/

[Ote03] Otellini, Paul. Transcript of the keynote speech in Intel Developer Forum, Fall 2003. [www-document]. September 23, 2003. [retrieved October 5, 2003]. From: http://www.intel.com/pressroom/archive/speeches/otellini20030916.htm

[Pas03] Passmark Software. PerformanceTest. [www-document]. 2003. [retrieved August 28, 2003]. From: http://www.passmark.com/products/pt.htm

[Pop74] Popek, Gerald J. Goldberg, Robert P. Formal Requirements for Virtualizable Third Generation Architectures. Communications of the ACM. Volume 17, Issue 7. July 1974. p. 412-421

[Pro02] Proctor, Tom. Microsoft Virtual Server: Administrator's Guide. Emulated Hardware of Virtual Server. [www-document]. 2002.

[Roa03] Roach, Steven. Making the move to Windows Server 2003: Migration, Integration, & Consolidation (Part 2). Microsoft USA Presentations. [Powerpoint presentation]. May 8, 2003. [retrieved June 21, 2003]. From: https://www.microsoft.com/usa/presentations/Making_the_Move_to_WS2003.ppt

[Rob00] Robin, John Scott. Irvine, Cynthia. Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor. Proceedings of the 9th USERNIX Security Symposium. [pdf-document]. August 2000. [retrieved June 30, 2003]. From: http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/robin/robin.pdf

[Ros03] Rosenblum, Mendel. Operating Systems and Systems Programming. Virtual Machine Monitors. [pdf-document]. 2003. [retrieved June 26, 2003]. From: http://www.stanford.edu/class/cs140/lectures/vmm.pdf

[Sis03] SiSoftware. SiSoftware Sandra – The Diagnostic Tool. Q & A Document. [www-document]. 2003. [retrieved August 29, 2003]. From: http://sisoftware.co.uk/index.html?dir=&location=qa&langx=en&a=

[Smi01a] Smith, J.E. An Overview of Virtual Machine Architectures. [pdf-document]. October 27, 20001. [retrieved June 6, 2003]. From: http://www.ece.wisc.edu/~jes/papers/vms.pdf

[Smi01b] Smith, J.E. ECE 902: Special Topics in Computers. Virtual Machine Architectures, Implementations, and Applications. Operating System VMs. [pdf-document]. October 1, 2001. [retrieved June 24, 2003]. From: http://www.ece.wisc.edu/~jes/902/ovhds/osvm_2.pdf

[Sug01] Sugerman, Jeremy. Venkitachalam, Ganesh. Lim, Beng-Hong. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. Proceedings of the 2001 USERNIX Annual Technical Conference. [pdf-document]. June 2001. [retrieved June 30, 2003]. From: http://www.usenix.org/publications/library/proceedings/usenix01/sugerman/sugerman.pdf

[Sun03] Sun Microsystems Inc. White paper: Sun Fire [tm] 12K and 15K Servers – High Availability Whitepaper in Data Center. [pdf-document]. February, 2003. [retrieved June 12, 2003]. From: http://www.sun.com/servers/wp/docs/Availability_12K15K_Jan30_PM.pdf

[SWs03] SWsoft. Virtual Environments. [www-document]. 2003. [retrieved June 24, 2003]. From: http://www.sw-soft.com/en/products/virtuozzo/ve/

[VMw99] VMware, Inc. Technical White Paper. [pdf-document]. February, 1999. [retrieved June 24, 2003]. From: http://www.ece.wisc.edu/~jes/902/papers/vmware.pdf

[VMw02] VMware, Inc. VMware ESX Server. User's Manual. Version 1.5. 2002.

[VMw03a] VMware, Inc. Enterprise-Class Virtualization Software. [www-document]. 2003. [retrieved June 24, 2003]. From: http://www.vmware.com

[VMw03b] VMware, Inc. VMware ESX Server. I/O Adapter Compatibility Guide. Version 1.5.2. [pdf-document]. 2003. [retrieved August 5, 2003]. From: http://www.vmware.com/pdf/esx_io_devices_15.pdf

[VMw03c] VMware, Inc. VMware Server Products: Enterprise-Class Virtual Machine Software for Intel Servers. 2003. [retrieved June 25, 2003]. From: http://www.vmware.com/products/server/

[Wal02] Waldspurger, Carl A. Memory Resource Management in VMware ESX Server. Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02). [pdf-document]. December 2002. [retrieved June 26, 2003]. From: http://www.waldspurger.org/carl/papers/esx-mem-osdi02.pdf

[Whi01] Whitaker, Andrew. Shaw, Marianne. Gibble, Steven D. Denali: Lightweight Virtual Machines for Distributed and Networked Applications. University of Washington Technical Report. [pdf-document]. February 2, 2001. [retrieved June 25, 2003]. From: http://denali.cs.washington.edu/pubs/distpubs/papers/denali_usenix2002.pdf

Appendix 1

Sensitive Register Instructions in IA-32 architecture processor virtualization. [Int03b], [Rob00].

Instruction	Operation
SGDT – Store Global Description Table Register	Stores contents of the global description table register (GDTR) to specified memory location.
SIDT – Store Interrupt Description Table Register	Stores contents of the interrupt descriptor table register (IDTR) to specified memory location.
SLDT – Store Local Descriptor Table Register	Stores segment selector from local descriptor table register to general-purpose register or memory location.
SMSW – Store Machine Status Word	Stores the machine status word to general-purpose register or memory location.
POPF – Pop Stack into EFLAGS Register	Pops word from top of stack and stores it to EFLAGS register.
POPFD – Pop Stack into EFLAGS Register	Pops doubleword from top of stack and stores it to EFLAGS register.
PUSHF – Push EFLAGS register onto the Stack	Decrements stack pointer by 2 and pushes the lower 16 bits of the EFLAGS register onto the stack.
PUSHFD – Push EFLAGS register onto the Stack	Decrements stack pointer by 4 and pushes entire contents of the EFLAGS register onto the stack.

Appendix 2

Protection System References in IA-32 architecture processor virtualization. [Int03b], [Rob00].

Instruction	Operation
LAR – Load Access Rights Byte	Loads the access rights from the segment descriptor (register or memory location) to general-purpose register.
LSL – Load Segment Limit	Loads the unscrambled segment limit from the segment descriptor (register or memory location) to general-purpose register.
VERR – Verify a Segment for Reading	Verifies whether the code or data segment specified source (register or memory location that contains segment selector) is readable from the current privilege level (CPL).
VERW – Verify a Segment for Writing	Verifies whether the code or data segment specified source (register or memory location that contains segment selector) is writable from the current privilege level (CPL).
POP – Pop a Value from the Stack	Loads the value from the top of the stack to general-purpose register, memory location or segment register.
PUSH – Push a Word or Doubleword Onto the Stack	Decrements the stack pointer and then stores the source operand on top of the stack.

(continues on next page)

CALL – Call Procedure	Saves procedure linking information on the stack and branches to the procedure specified with the destination operand (immediate value, general-purpose register or memory location).
JMP – JUMP	Transfers program control to a different point (immediate value, general-purpose register or memory location) in the instruction stream without recording return information.
INT – Call to Interrupt Procedure	Generates a call to the interrupt or exception handle specified as interrupt vector.
RET – Return from Procedure	Transfers program control to a return address located on the top of the stack.
STR – Store Task Register	Stores the segment selector from the task register (TR) in to general-purpose register or memory location.
MOV – Move	Moves the contents of control register (CR0, CR2, CR3 or CR4) to a general-purpose register or vice versa.