Czech Technical University in Prague
Faculty of Electrical Engeneering
Department of Computer Science

VIRTUAL DISTRIBUTED COMPUTING SYSTEMS AND THEIR APPLICATIONS

by

Jan Fesl

A dissertation thesis submitted to
the Faculty of Electrical Engeneering, Czech Technical University in Prague

Ph.D. Programme: Electrical Engeneering and Information Technology
Branch of study: Information Science and Computer Technology

České Budějovice, August 2017

Supervisor:
doc.Ing. Jan Janeček, CSc.
Department of Computer Science
Faculty of Electrical Engeneering
Czech Technical University in Prague
Karlovo náměstí 13
121 35 Praha 2
Czech Republic

Abstract and Contributions

This dissertation thesis deals with the architecture, benchmarking, optimization and implementation of virtual distributed computing systems. Large distributed systems representing the performance background of all modern cloud computing architectures have become a very hot topic at present. This dissertation thesis offers an introduction into modern technologies showing a rapid impact on the system performance. One of them is virtualization technology, whose real efficiency was a standalone part of the research. Large distributed systems consume a huge amount of electric power, therefore their optimization was also discussed. New ideas originated from the research were incorporated into the proposal of a new distributed system called Centrální mozek univerzity (CMU). This system is able to manage and automatically optimize large virtualization infrastructures. It is also accessible to the end-users. This system is currently used in teaching many subjects and became a prototype of an educational system of new generation.

In particular, the main contributions of the dissertation thesis are as follows:

1. Methodology, design and implementation of a new software benchmark utility able to measure virtualization efficiency of the distributed architecture.

2. New approach in a possible migration of the entire distributed systems between various data centres.

3. New distributed algorithm for virtual machines consolidation, rapidly reducing the energy consumption.

4. Design, description and implementation of the distributed virtualization system CMU.

Keywords:
virtualization, distributed computing systems, cloud computing, live migration, green computing.

Abstrakt

Disertační práce se zaobírá architekturami, testováním, optimalizací a implementací virtuálních distribuovaných systémů. Rozsáhlé distribuované systémy, které stojí v pozadí všech moderních cloudových architektur, jsou v současné době skutečně žhavým tématem. Práce obsahuje teoretický úvod do moderních virtualizačních technologií, které rapidně ovlivňují výpočetní výkon. Efektivita virtualizačních technologií byla jedním z klíčových témat vlastního výzkumu. Rozsáhlé výpočetní infrastruktury spotřebovávají obrovské množství elektrické energie, přičemž optimalizace spotřeby byla dalším předmětem výzkumu. Inovativní myšleny, které vznikly ve fázi výzkumu, byly integrovány do návrhu a implementace nového distribuovaného virtualizačního systému, který nese název Centrální mozek univerzity (CMU). Tento systém je schopen plně automatizovaně spravovat a optimalizovat distribuované infrastruktury pro virtualizaci. Systém dokáže zprostředkovat přístup pro využití svých výpočetních prostředků koncovým uživatelům. Takováto koncepce systému je současně prototypem výukového systému nové generace, ve kterém je využita architektura datových center pro výuku univerzitních předmětů.

Klíčová slova:
virtualizace, distribuované výpočetní systémy, cloud computing, live migrace, green computing.

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, Doc. Jan Janeček. He has been a coordinator of my research work and helped me with numerous problems and professional advancements.

Further, I would like acknowledge the University of South Bohemia, Faculty of Science for financing all hardware technologies used in the implemented solution.

I would also like to thank to Dr. Petr Šimek, the Head of the Laboratory of Biochemistry and Metabolomics from the Biology Centre, Academy of Sciences in České Budějovice who introduced me into the world of science.

Finally, my greatest thanks go to my girl-friend Marie, my mom and sisters, for their infinite patience and care, and to my friends Jiří Čehák and Michal Konopa who cooperated with me on some topics of the thesis.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

CMU	Centrální Mozek Univerzity
DVMCA	Distributed Virtual Machines Consolidation Algorithm
CT	Communication Traffic
CPU	Central Processing Unit
DRS	Dynamic Resources Scheduling
DWDM	Dense Wavelength Division Multiplexing
GPU	Graphic Processing Unit
IOPS	Input Output Operations per Second
IQR	Inter Quartile Range
JVM	Java Virtual Machine
LR	Local Regression
MAD	Median Absolute Deviation
MCS	Maximum Correlation Similarity
MSU	Maximal System Utilization
MMT	Minimum Migration Time
MTU	Maximal Transfer Unit
NAS	Network Area Storage
NIC	Network Interface Card
RAM	Random Acces Memory
SATA	Serial Advanced Technology Attachment
SLA	Service Level Agreement
SR-IOV	Single Root Input Output Virtualization
SSD	Solid State Drive
UDP	User Datagram Protocol
VDMQ	Virtual Machine Device Queue
VLAN	Virtual Local Area Network
VMDC	Virtual Machine Direct Connection
VM	Virtual Machine
VMM	Virtual Machine Monitor

Chapter 1

Introduction

1.1 Motivation

The main motivation was to introduce the cloud computing technology into the university educational and research environment, and to enable the investigation of internal cloud architectures and technologies both under the stress and in real deployment during the runtime. Therefore, a distributed system being able to manage powerful cloud infrastructure has been gradually created.

Similar systems like OpenNebula or Open-Stack, were available, but they did not meet the needs, as they were developed by thousands of people and their implementation suffered from a number of drawbacks due to a long time of their development, e.g., they were implemented in many programming languages and their main modules were practically impossible to change. The motivation factors for improvement were the abilities of currently used virtualization technologies. One of the tasks to solve was the overhead of various virtualization technologies which causes the performance loss. Another task was the consumption of electric energy. The running system, not fully saturated, continuously consumes a substantial additional amount of energy, which produces subsequent additional unwanted costs.

1.2 Goals of the Dissertation Thesis

1. Methodology, design and implementation of a new software benchmark utility being able to measure the virtualization efficiency of distributed architecture that can contain various virtualization technologies.

2. New approach for a possible migration of the entire distributed system between various data centres.

3. New distributed algorithm for virtual machines consolidation that rapidly reduces the energy consumption.

4. Design, description and implementation of the distributed virtualization system used for the education and research based on cloud computing.

1.3 Structure of the Dissertation Thesis

The thesis is organized into five chapters as follows:

1. *Introduction:* Motivations and goals are described and a list of dissertation thesis contributions is presented.

2. *Current State of the Art:* The reader is introduced to the necessary theoretical background and the current state of the art is surveyed. The descriptions of the principles used in virtualization technologies are mentioned, as well as the algorithms for virtual machines migrations and consolidation.

3. *Overview of Our Approach:* Newly created algorithms and tools are presented and evaluated. The tool for the virtualization efficiency measurement, a new approach for the live migration of the virtual distributed systems and a new algorithm for the virtual machines consolidation are described. The end of this chapter is devoted to the description of a newly created distributed virtualization system.

4. *Main Results:* A brief overview of all important research contributions is introduced.

5. *Conclusions*: The results of our research are summarized and possible topics for further research are suggested.

Chapter 2

Current State of the Art

This chapter summarizes the current knowledge about virtualization technologies, describes their principles in detail and shows their deployment in the modern data centres.

The second half of this chapter contains the description of virtual machine migration algorithms used for virtual machine migration between various physical nodes.

The final part contains the introduction into the virtual machines consolidation, where the basic principles and commonly used heuristic rules are mentioned.

2.1 Virtualization

Basic virtualization principles [1] have been known for many years. The first research was performed in the 70s of the last century. It is associated with the IBM 370/390 computer system. The main idea standing behind virtualization is hardware resource sharing of one standalone computer between many virtual machines (VMs) using the hardware simultaneously. Further important aspect is the isolation of all the running VMs on the same host. The main advantages are the cost and power reduction, portability, possibility of performance consolidation, etc. One type of virtualization, which is discussed in the following section, allows emulating the different CPU architecture than it is in real. The modern virtualization for x86 systems was reintroduced in 1999 by the VMware company [2]. The first direct hardware-assisted virtualization for x86 CPUs was introduced by Intel in 2005, resp. by AMD in 2006. Since 2015 all modern hardware-assisted technologies have been implemented in current desktops and servers.

x86 System Virtualization Architecture

The corner-stone of every virtualization system is a hypervisor. The hypervisor is a software layer [4], mostly a kernel part of an operating system that serves as a provider between the real and virtual worlds. The real world is represented by the physical hardware (bare metal), on which the virtual machines run.

The virtual world is created by the virtual hardware, which is simulated by the Virtual Machine Monitor (VMM). VMM is mostly a part of the hypervisor. Its model is depicted in Figure 2.1. The functionality of every hypervisor is strongly dependent on the used virtualization type and its implementation. Each VMM is responsible for sharing the CPU, memory and I/O devices to successfully virtualize the system. x86 architecture supports four privilege levels or rings, where the most privileged one is denoted as 0 and the least privileged ring as 3. Operating systems run in Ring 0, user applications in Ring 3, and Rings 1 and 2 are not typically used.

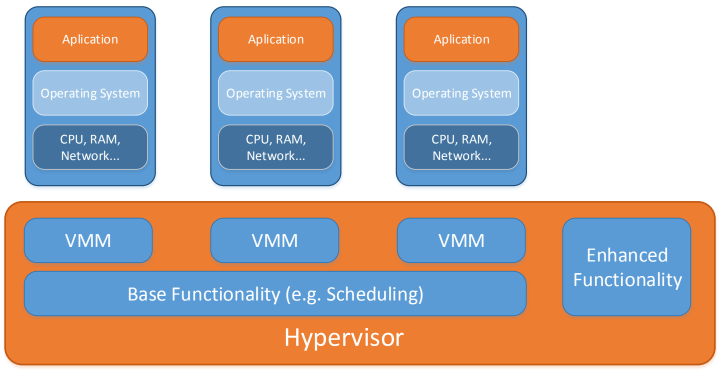


Figure 2.1: Architecture of the general virtualization system. The hypervisor manages all virtual machines running on the physical computer.

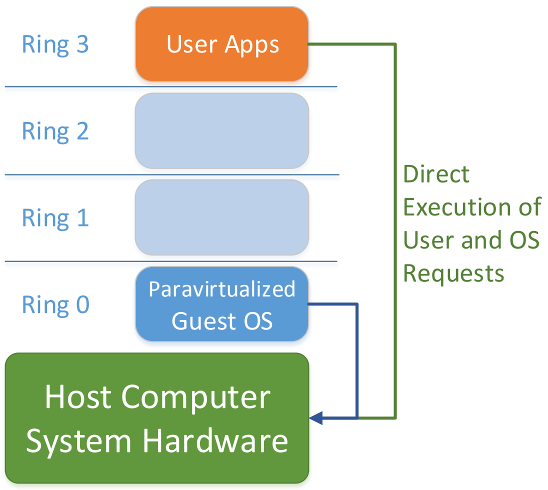

Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Paravirtualized Guest OS
Host Computer System Hardware
Direct Execution of User and OS Requests

Figure 2.2: Virtualization architecture privilege levels (rings).

2.2 Virtualization Principles

2.2.1 Emulation

In hardware emulation, the virtualization software (hypervisor) creates a virtual machine by emulating the entire hardware platform. The operating system is loaded into a virtual machine and it is not necessary to be modified. As it makes calls for system resources, the hardware emulation software catches the system calls and redirects them to the hypervisor. The hypervisor itself makes calls to the actual physical hardware. Hardware emulation is often called bare-metal virtualization, to symbolize the fact that no software is between the hypervisor and the "metal" of a virtualization host. In this approach, the hypervisor catches the system calls from the virtual machines and coordinates the access to the hardware directly. More detailed information is available in [7].

2.2.2 Paravirtualization

This virtualization type does not emulate hardware in a software layer like in the previous case. A paravirtualization hypervisor multiplexes the access to the hardware resources of the virtualization host. In paravirtualization, the hypervisor resides in hardware (paravirtualization is a bare-metal virtualization architecture) [4]. One or more guest operating systems (equivalent to the virtual machines in hardware emulation) run on the top of the hypervisor. A privileged guest runs as a guest virtual machine with the privileges allowing direct access to underlying hardware resources. The principle is demonstrated in Figure 2.3. The most famous paravirtualization hypervisor today is Xen. Every operating system running in paravirtualized environment must have a special modification for the specific hypervisor.

2.2.3 Full Virtualization with the Binary Translation

This virtualization type translates the guest operating system kernel code to replace nonvirtualizable instructions by the new sequences of instructions that have the intended effect on the virtual hardware. User-level code is directly executed on the processor for high performance virtualization. Each virtual machine monitor provides each Virtual Machine with all the services of the physical system, including a virtual BIOS, virtual devices and virtualized memory management. This combination of binary translation and direct execution provides full virtualization, since the guest OS is fully abstracted from hardware by the virtualization layer. The guest OS is not aware of being virtualized and requires no modification. Full virtualization requires no hardware assistor to virtualize sensitive and privileged instructions. The hypervisor translates all guest operating system instructions on the fly and saves the results for further use. User level instructions run unmodified at the native speed. Full virtualization offers good isolation and security for the virtual machines, and simplifies the migration between various virtualization hosts. The principle is graphically shown in Figure 2.4.

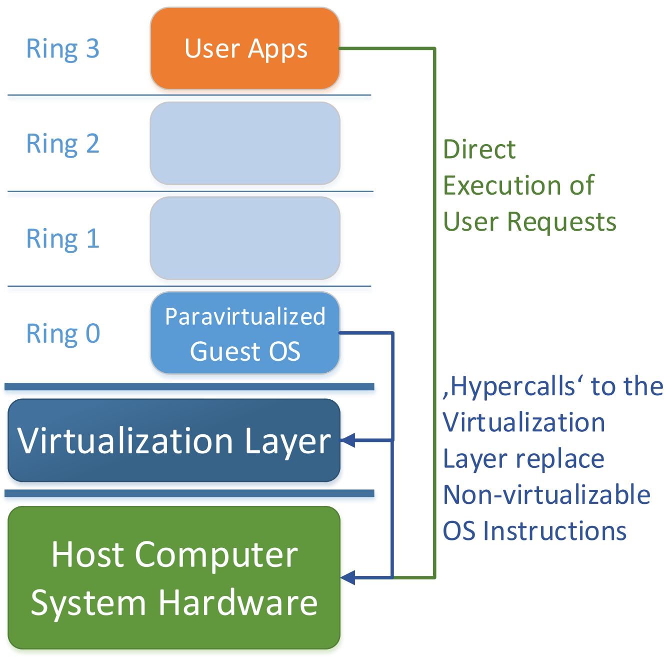

Ring 3
User Apps
Ring 2
Ring 1
Ring 0
Paravirtualized Guest OS
Virtualization Layer
Host Computer System Hardware
Direct Execution of User Requests
‚Hypercalls' to the Virtualization Layer replace Non-virtualizable OS Instructions

Figure 2.3: Paravirtualization principle. The hypervisor multiplexes the access of the virtualization guest to real hardware.

2.2.4 Hardware Assisted Virtualization (Native Virtualization)

The first generation of this virtualization type was introduced in 2005 by Intel (called VT-x), resp. by AMD in 2006 (called VT-v). The main goal of this technology is to maximally accelerate the virtualization process and delete the operation overhead caused by binary translation. The principle is depicted in Figure 2.5. The state of every guest virtual machine is stored or reloaded into/from the special structure by the periodical virtual machine/host switching. More detailed description is available in the following Section. This virtualization technique is sometimes known as "native virtualization". More info can be found in [8] or [62].

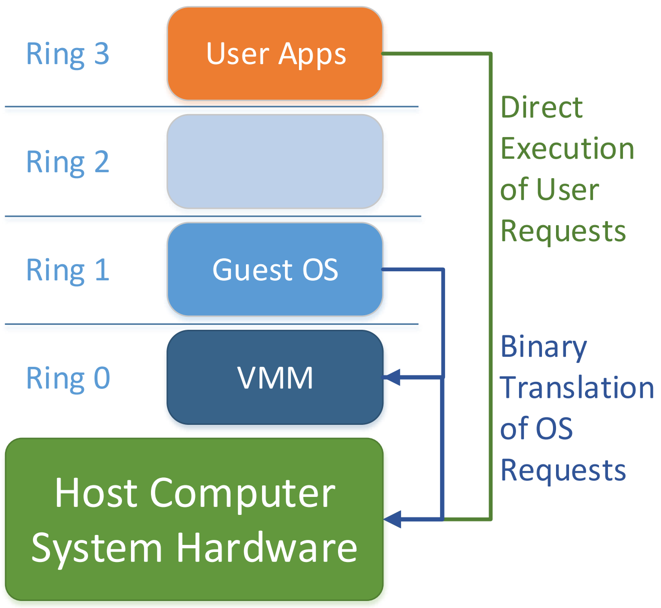

Ring 3
User Apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host Computer System Hardware
Direct Execution of User Requests
Binary Translation of OS Requests

Figure 2.4: Full virtualization with binary translation. The nonvirtualizable kernel instructions are replaced on the fly and cached.

2.2.5 Intel VT-x and AMD VT-v Technologies

Both technologies have the same goals and principles. Intel VT-X Technology introduces new modes of the CPU operation - VMX root and VMX non-root operations. Both operating modes support the execution in all four privilege rings. The VMRUN instruction activates a VM entry, enabling thus the data transfer between the host and guest running modes. Return switching from the host to the guest modes, executed by a VM exit instruction, can be triggered both by conditional and unconditional events. For example, the INVD instruction unconditionally triggers a VM exit, while a write to register or memory locations might depend on which bits are modified. What is important for the interaction between the hosts and guests is the virtual machine control structure (Intel's VMCS, which is equal to AMD's VMCB - Virtual Machine Control Block) which contains both guest and host states. On the VM entry, the guest processor state is loaded from the VMCS after storing the host processor state. The VM exit swaps these operations, saving the guest state and loading the host state. The processor state consists of segment registers, control register 3, and the interrupt descriptor table (IDT) register. CR3 holds the physical location of the page tables. By loading and storing this register on the VM entry and exit, guest virtual machines can run in a separate address space than the VMM. Since the VMCS does not contain any general purpose registers, the function can be provided by the VMM, whenever it is necessary, which improves VM entry and VM exit performance. On

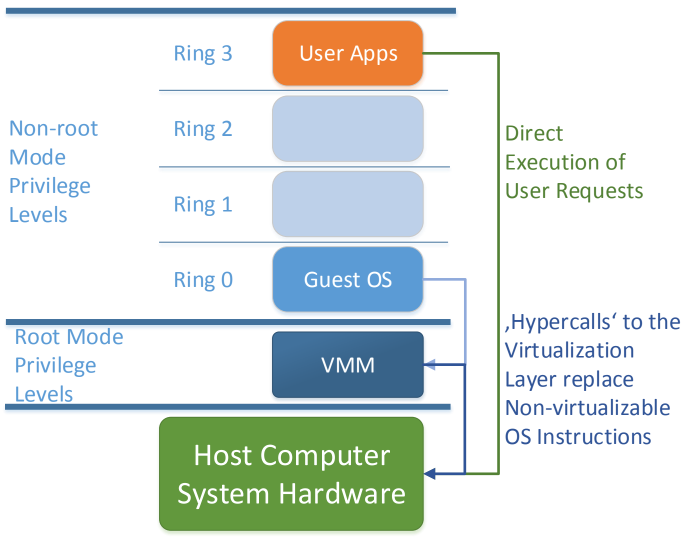

Ring 3
User Apps
Ring 2
Non-root Mode Privilege Levels
Ring 1
Ring 0
Guest OS
Direct Execution of User Requests
‚Hypercalls' to the Virtualization Layer replace Non-virtualizable OS Instructions
Root Mode Privilege Levels
VMM
Host Computer System Hardware

Figure 2.5: Hardware assisted virtualization. The hypervisor multiplexes the access of the virtualization guest to the real hardware.

a related node, a guest's VMCS is referenced with a physical address to avoid translating a guest virtual address. As mentioned above, the biggest difference between the host and guest modes (VMX root and non-root operations) is that many instructions in the guest mode will cause a VM exit.

AMD-V is functionally quite similar to Intel's VT-x. However, the two competing approaches are incompatible in terminology. A shared host-guest control structure is called: virtual machine control structure (VMCS) by Intel, and virtual machine control block (VMCB) by AMD. For more detailed comparison, see Shen [3].

2.2.6 Memory Virtualization

Another virtualization topic to discuss is the memory virtualization. It requires sharing the physical host memory among all executed virtual machines and allocating it to the virtual machines. Virtual machine memory virtualization is very similar to the virtual memory management provided by modern operating systems. Applications are able to write into/read a contiguous address space that is not necessarily bound by the underlying physical memory in the system. The operating system stores the mappings of virtual page numbers to physical page numbers in the page tables. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) for optimizing virtual memory performance. To run multiple virtual machines on a single system, one more level of the memory virtualization is required. It is necessary to virtualize the MMU to support the guest OS. The guest OS tries to control the mapping of virtual addresses to the guest memory physical addresses, but the guest OS cannot have a direct access to the physical host memory. The VMM is responsible for mapping host physical memory to the virtual machine memory and uses shadow page tables to accelerate the mappings. The VMM uses TLB hardware to map the virtual memory directly to the machine memory to avoid two levels of translation on every access. If the guest OS changes the virtual memory mapping to physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup. The MMU virtualization causes some overhead for all virtualization approaches. The principle is described in Figure 2.6. Intel introduced this technology under the name EPT (extended page tables), resp. AMD under the name NPT (nested page tables).

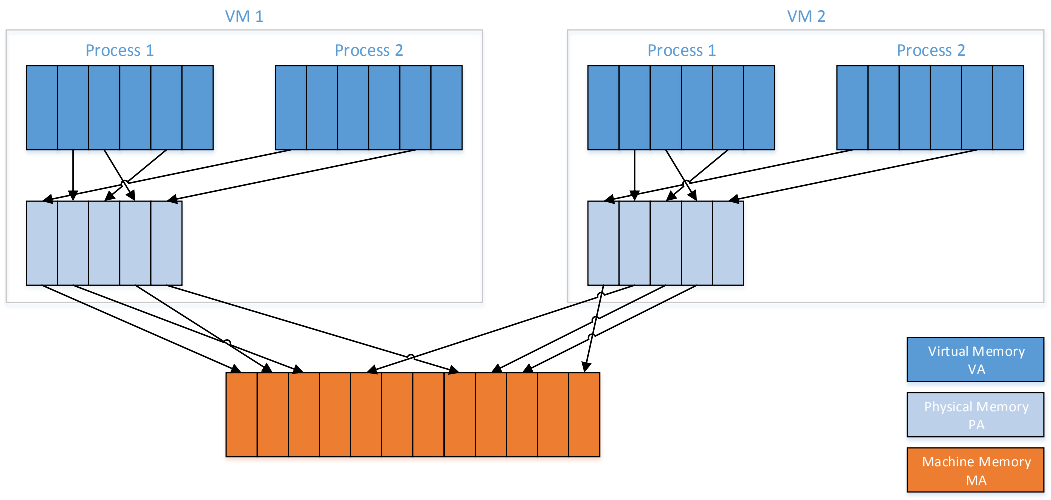


Figure 2.6: Virtual memory remapping by the VMM. This approach is similar to the virtual memory access in current operating systems.

2.2.7 Devices Direct Access and I/O Virtualization

In the system virtualization, peripheries and I/O virtualizations can be taken as bottlenecks. They require managing I/O requests between the virtual machine devices and physical hardware. Software-based I/O virtualization and management, in contrast to direct throughput to hardware, enables a rich set of features and simplified management. Virtual NICs and switches create virtual networks between virtual machines without a network traffic-consuming bandwidth on the physical network through the operating memory. NIC teaming allows multiple physical NICS to appear as one for virtual machines. The key to effective I/O virtualization is to preserve these virtualization benefits while keeping the CPU utilization to a minimum. The hypervisor virtualizes the physical hardware and presents each virtual machine with a standardized set of virtual devices. These virtual devices effectively emulate hardware and translate the virtual machine requests to the system hardware. This standardization on a consistent device drivers also helps with virtual machine standardization and portability across the platforms, as all virtual machines are configured to run on the same virtual hardware, disregarding the actual physical hardware in the system. These ideas are incorporated both in Intel VT-d/VT-i and AMD-DEV technologies. The commonly used term VT-c is used for the virtual network interface virtualization. A negative aspect of these technologies lies in the problems caused by the virtual machine migration, because the hardware mapping is unique for the concrete hardware machine. The principle is depicted in Figure 2.7.

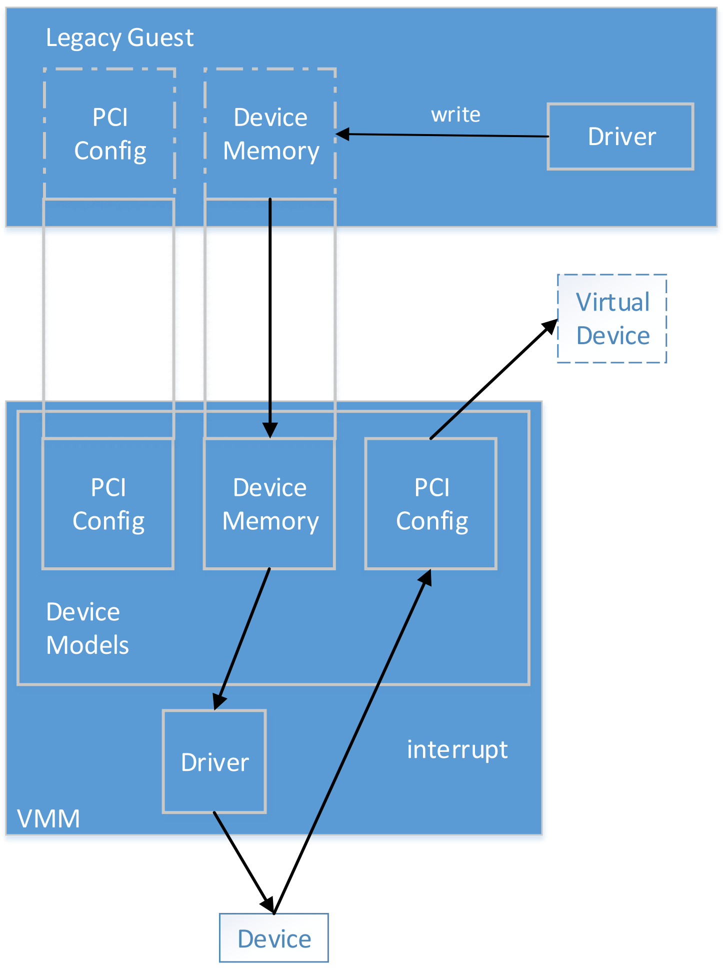


Figure 2.7: I/O virtualization and direct access of virtual machines to devices.

2.2.8 Network Virtualization

Nowadays, all virtualization software with network capabilities has built-in virtual switches, whereas most of them function as routers. Their aim is to connect multiple virtual machines into one or more networks, in the same way as real switches or routers. The idea is demonstrated in Figure 2.9. Intel VT-c significantly increases the speed of delivery and reduces the workloads of the VMM and server processor through these functions implemented in private network chips. VT-c includes a Virtual Machine Device Queue (VMDq) and Virtual Machine Direct Connection (VMDc). A universal model of the network virtualization architecture is illustrated in Figure 2.8. More detailed overview can be found in [3] or [5] and [62].

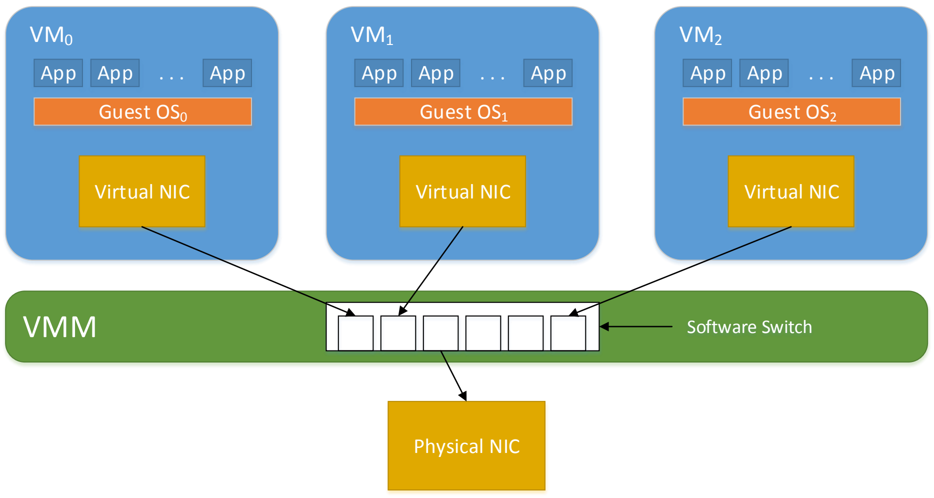


Figure 2.8: Universal model of the network virtualization.

2.2.8.1 Virtual Machine Device Queue

In traditional server virtualization environment, VMM must categorize every individual data packet, and deliver it to its assigned VM, which will take up a lot of processor cycles. And with VMDq, this function can be performed by specified hardware within network card, and VMM is only responsible to deliver presort data packet group to appropriate guest OS. This slows down I/O latency, and provides more available cycles for processor to deal with business applications. I/O throughput can be more than doubled by Intel VT-c, so that virtualized applications are able to reach the level of the host throughput. Every server will integrate more applications, but I/O overhead will be less, the principle is depicted in 2.9.

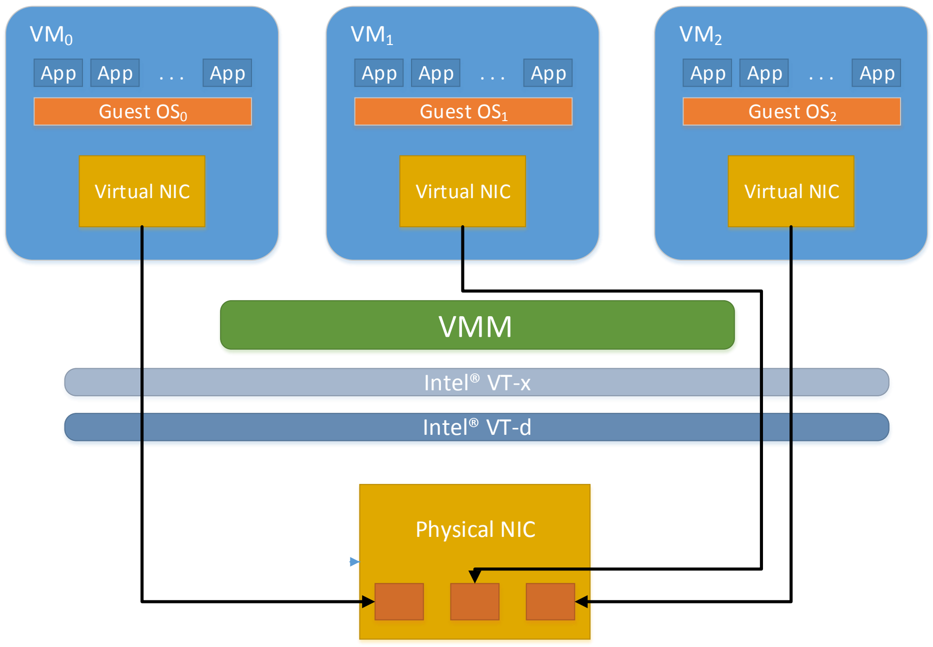


Figure 2.9: Machine Device Queue - the efficient way for data delivery.

2.2.8.2 Virtual Machine Direct Connection

With the aid of a single- root I/O virtualization (SR-IOV) standard in PCI-SI, a VM direct connection (VMDc) supports a VM direct access to network I/O hardware, and thus significantly improves the performance. As mentioned before, Intel VT-d supports a direct communication channel between the guest OS and I/O port. SR-IOV can be extended by the support of each of the I/O port multiple communication channels. For example, each of the 10 guest operating systems can be assigned to its isolated virtual 1Gb/s private link, i.e., just only one 10 Gigabit/s server network card can be used. These links bypass the VMM switch and can further enhance I/O in performance and reduce the workload of the server processors.

2.3 Virtualization in Data Centres

The current data centres allow enterprises and small businesses to transform, to manage and to optimize their IT systems infrastructure through the virtualization. A common large architecture overview is illustrated in Figure 2.10. More information about large data centres can be found in [9]. The architecture for the virtualization is fully distributed and has specific building blocks. The architecture for the virtualization has specific building blocks and is fully distributed. The main parts of such infrastructures are as follows:

1. *Management servers*: They manage all the processes between all building nodes of a distributed architecture. They manage the resource allocation, virtual machine placement, execution, etc.
2. *Virtualization servers*: Host-running virtual machines. Virtual machines run on their metal in real.
3. *Data storages*: They mostly store the contents of the hard drives belonging to virtual machines. The data storages can have various architectures and various types of connection to the virtualization servers.
4. *High throughput networks*: They connect all parts of the entire data centres together and allow managing and optimizing the architecture performance. Their throughput and latency have a big impact to the total performance, as well.

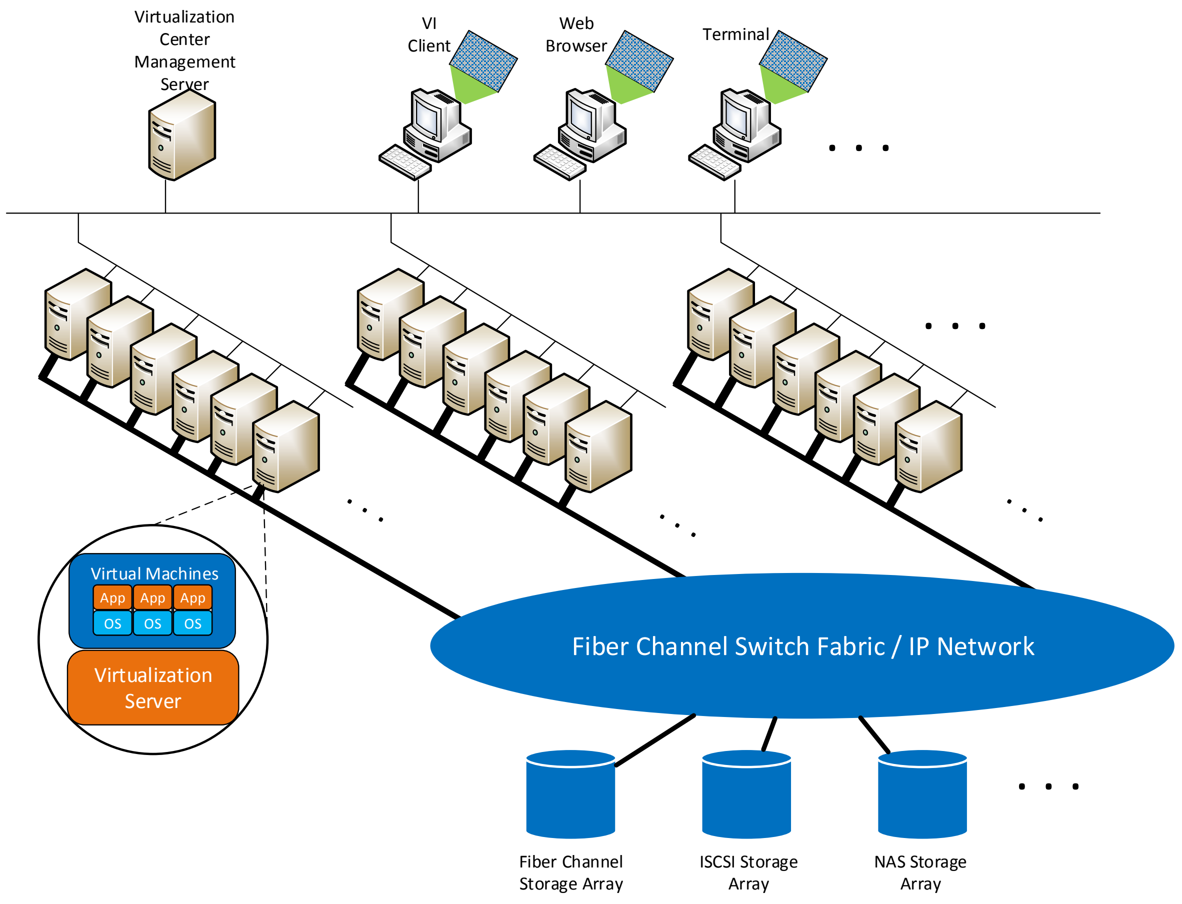


Figure 2.10: Overview of building blocks in a large distributed virtualization infrastructure.

2.3.1 Current Technologies Used in Data Centres

2.3.1.1 Virtualization Technologies

The most used technology today is hardware-assisted virtualization mentioned in Section 2.2.4. The world virtualization leaders in this field are large technology companies like VMware, Microsoft and Cytrix, whose distributed solutions for data centres will be discussed in Section 2.3.2. Another good source for the future study of data store technologies is presented in [6].

2.3.1.2 Data Storage Technologies

Data Storage Devices

Currently used data storage devices in commercial data centres are solid state drives (SSDs), magnetic drives (mostly SATA/SAS) and magnetic tapes. SSD drive technology is practically the same as in memory ash card technology, but it contains an interface for the magnetic drive emulation. The biggest and the most important difference between SSD and magnetic SATA III drives is in the number of input/output operations per second (IOPs) that can be performed by the device. The number of IOPs correlates with the physical principle of the applied technology. At magnetic hard drives, the number reaches the units of hundreds (smaller than 1000), and at SSD drives, the tens of thousands (typically over 50000). The IOPs value has a substantial impact on the number of virtual machines, which can simultaneously use such a device. The disadvantage of SSD drives is their price; at the moment it is cca 10 times higher than the price of magnetic drives. Magnetic types are used only for data archiving, their performance is for real time virtualization systems insufficient.

Magnetic Drives Raid Array with SSD Caching

This solution is very popular for its performance/price ratio. Magnetic drives with higher capacity are the members of some raid arrays (mostly of level 5 or 6) and SSD drives serve as a cache memory between the magnetic data storage and operating memory. The principle is depicted in Figure 2.11.

2.3.1.3 Network Technologies

Physical Layer

The optical networks with single/multimode transmitters are mostly used for data transmission. The network topologies are typically organized into n-dimensional hyper cubes. For the long-distance transmissions, DWDM systems are used.

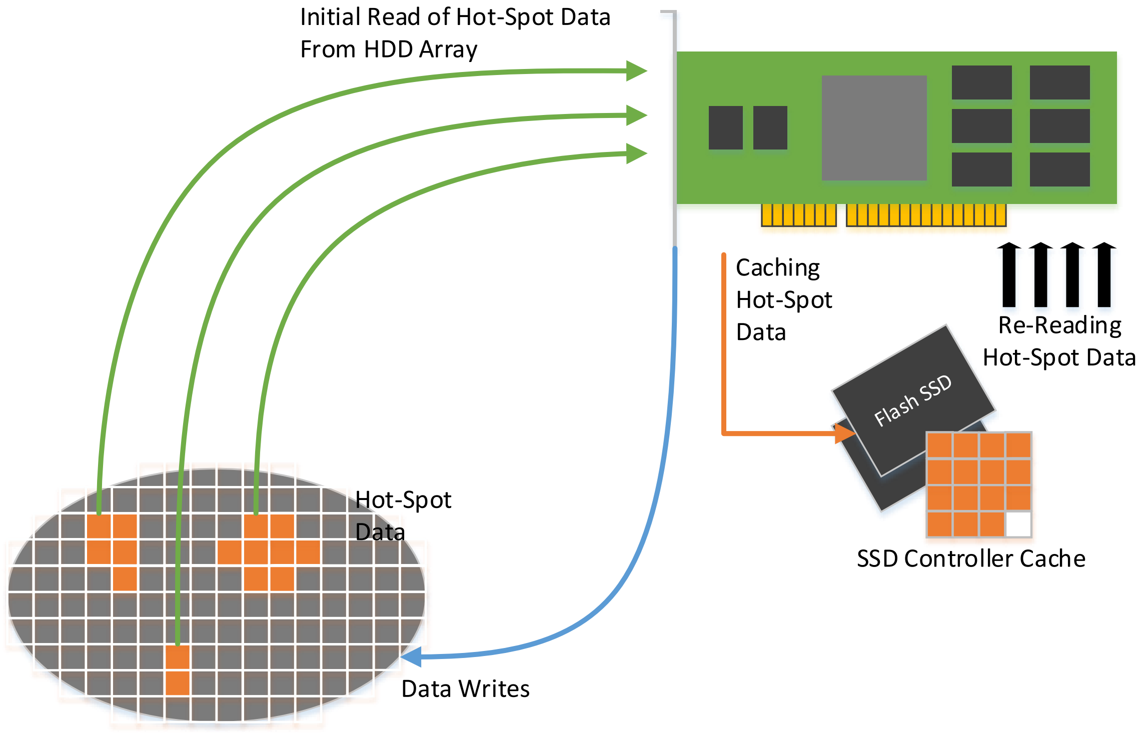


Figure 2.11: SSD caching principle. This approach leads to a rapid throughput increase.

Link Layer

Advanced technologies like InfiniBand, FibreChannel or Ethernet are required for their high throughput and small latency. The cheapest and widely spread technology is 10/40 Gbit Ethernet. IEEE 802.1Q VLAN technology is used for virtual network traffic isolation.

Network Interface Card Teaming

Two or more physical network adapters are connected to the NIC Teaming solution multiplexing unit, which then presents one or more virtual adapters to the operating system. There are several different algorithms that distribute inbound and outbound traffic between physical network adapters. This technology is directly supported by the Microsoft Hyper-V-based virtualization platform. The solution principle is depicted in Figure 2.12.

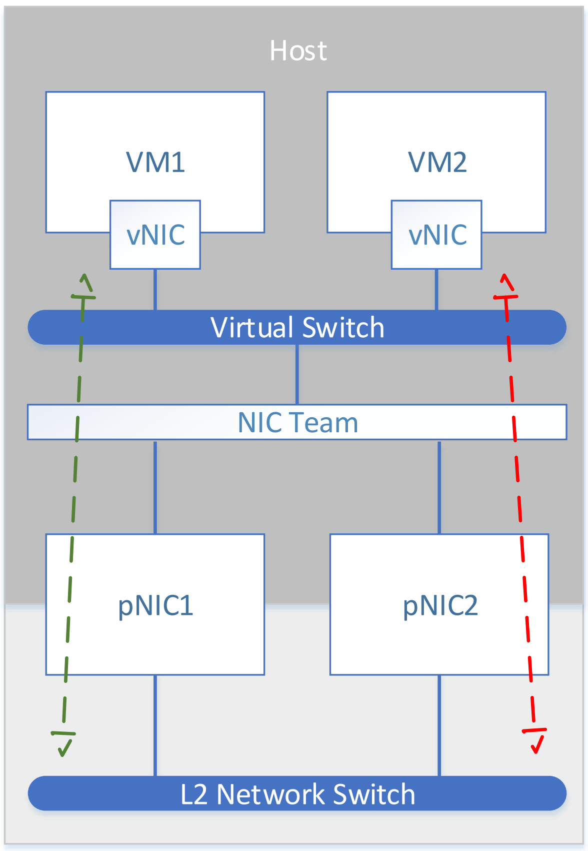


Figure 2.12: NIC teaming solution offers the higher throughput or the fault tolerant connection.

2.3.2 Systems for Large Distributed Virtualization Infrastructure Management

All solutions that will be discussed in the following subsection are the commercially-based products. There exists a common surface of the features implemented in various solutions. A complex overview of the management technologies is provided by Bouras in [7].

VMWare Sphere

At present, a leading virtualization software company is VMWare, whose products first appeared on the market in 1999. The first solutions were desktop-oriented, and the distributed platform with the central management was introduced under the name "Virtual infrastructure" in 2006. The system was than renamed to VMWare Spehere. The system was than renamed to VMWare vSpehere. The system supports only a proprietary hypervisor that runs on ESXi servers. The last version of VMWare vSphere is now 6.5.

The main features of this system are as follows:

1. *VMWare Vmotion:* This feature allows the migration between the physical hosts of the virtual machines at the runtime. The description of this technology is in the Section 2.4 contents. This technology is used for the runtime infrastructure optimization.
2. *VMWare DSR:* This tool dynamically monitors the workload of the running virtual machines and the resource utilization of the physical servers within an infrastructure. It checks the results against the resource assignment policies, i.e., whether there is a space for improvement. DSR uses VMotion and dynamically reassigns virtual machines to different physical servers.
3. *VMware HA:* If the hosting server fails, the module automatically enables a quick restart of virtual machines on different physical servers within the infrastructure. More complex overview of this feature can be found in [11].

Microsoft System Center

It is a Microsoft equivalent to VMware vSphere. The system natively supports only the Microsoft Hyper-V hypervisor.

1. *Live migration:* This technology is the equivalent to the VMware VMotion.
2. *Virtual Machine Cluster:* is the equivalent to the VMware HA, a replication of running servers in runtime. It allows fast switching in case of failure.

Citrix XenServer

1. *Xen Motion:* This technology is the equivalent to the VMware Vmotion.
2. *High availability:* Its automatic restart capability allows restarting VMs, if any failure occurs at the VM, hypervisor or server levels, and enables bonding network interfaces for network redundancy. Automatic restart helps to protect virtualized applications and to ensure higher levels of availability.
3. *Host power management:* It allows lowering the data centre electricity consumption by the dynamic consolidation of the VMs on less power-demanding systems, and later powering off underutilized servers, since the demand for the services fluctuates.
4. *Memory optimization:* It reduces costs and improves application performance and protection by sharing unused server memory among the VMs on the host server.

2.3.3 Open Source Systems for Virtualization Infrastructure Management

The open source solutions, e.g., OpenNebula, OpenStack, CloudStack, Eucalyptus etc., are free alternatives to the commercial systems. Such systems try to unify the access. They are independent only of the specific hypervisor. The functionality is similar to the solutions mentioned above, but the unified access to the hypervisors leads to the only partial support of all features available for the specific virtualization platform. Their implementation is mostly inhomogeneous; various programming languages are used and their possible adaptation for a specific, not fully compatible, type of usage is very complicated.

2.4 Virtual Machine Migration

Virtual machine is a concept that divides one physical computer into many virtual computers. Computing resources of this physical computer are shared by all virtual machines. Every virtual computer consists of three main parts: virtual (emulated) computer hardware, virtual memory and virtual data storage (hard disk). A virtual hard disk is mostly located on the network data storage device. The storage device contains a specific network file system that simply allows using the hard disk.

2.4.1 Types of Migration

There are two main strategies of the virtual machine migration-offline and online (live). The first strategy is much simpler, because the whole migration process consists only of three main steps. More detailed information can be found in [12].

2.4.2 Off-line Migration Algorithm

Three phases of the offline migration process are as follows.

Algorithm 2.1 Off-line virtual machine migration

1. Pushing and stopping the virtual machine (VM) on physical host A.
2. Copying the saved memory (M) and virtual hardware settings (H) of the VM to physical host B. Data storage (hard disk) at B is mostly only reloaded from a shared network device.
3. Executing the VM (with M, and H) on physical host B.

The technology is well- known and widely used for virtual machine migration. The main disadvantage is that some services provided by a virtual machine are not available in the migration process (lasting the tens of seconds or more [13]). This solution cannot be readily utilized for the services such as DNS, SMTP, LDAP, RADIUS because they require high availability. A similar situation appears in the distributed computing systems, because it is not factually possible to stop all the nodes at the same time.

2.4.3 Live Migration Algorithms

A live virtual machine migration is a more complicated problem, because the memory content of the migrating machine is under the permanent change all the time. There is another limitation caused by a different available data throughput into the virtual machine memory within the same physical environment and between different physical hosts connected within the network. A maximum theoretical data throughput to the virtual machine memory can be the hundreds of gigabits per second [14], unlike the throughput between two network-connected physical hosts, where it may be only the tens of gigabits per second, which are the typical values for InfiniBand, the latest Ethernet technology, and Fiber Channel. Frequent changes in the memory pages and a low network throughput can cause the impossibility to finish the migration process.

Live memory migration scheme is described in detail in [12] and [13]. The basic migration scheme is as follows:A virtual machine VM migrates from physical host A to physical host B. Gustaffson's work [15] tries to accelerate this phase using the ultra-fast network area storage. There are six phases of live migration algorithm 2.2; the entire process is depicted in Figure 2.13 .

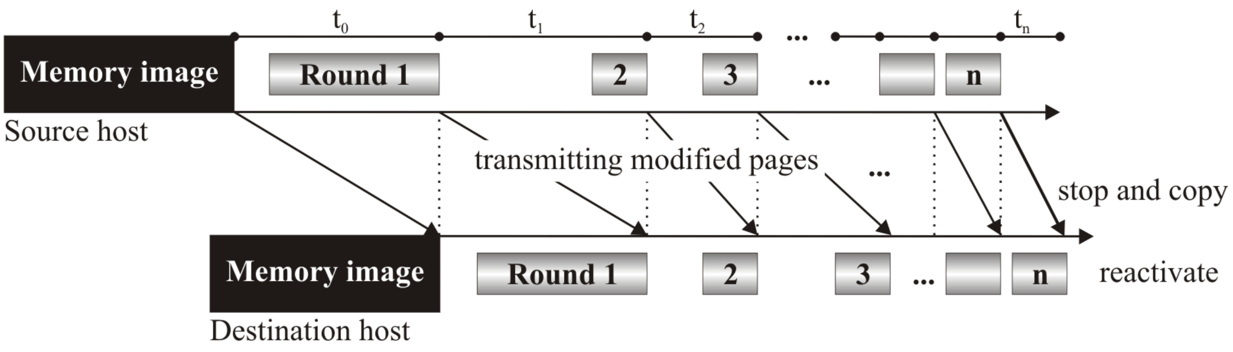


Figure 2.13: Live migration of the virtual machines has six various parts. It strongly depends on the memory changes and the network performance.

Algorithm 2.2 Live virtual machine migration

1. *Pre-Migration:* An active VM runs on physical host A. To speed up any future migration, a target host may be preselected, where the resources required to receive the migration will be guaranteed.
2. *Reservation:* A request is generated to migrate a given VM from host A to host B. B must confirm that it has the necessary resources and reserves a VM container of an appropriate size. The failure of a resource reservation here means that the VM simply continues to run on, staying unaffected.
3. *Iterative Pre-Copy:* During the first iteration, all memory pages are transferred from A to B. Subsequent iterations copy only those pages that are modified during the previous transfer phase.
4. *Stop-and-Copy:* A VM run is suspended at A and VM network traffic is redirected to B. The CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage, a consistent VM copy is suspended at both A and B. The copy at A is still considered to be primary and is reactivated in case of failure.
5. *Commitment:* Host B indicates to A that it has successfully received a consistent VM image. Host A acknowledges this message as a commitment of the migration transaction: host A may now discard the original VM, and host B becomes the primary host.
6. *Activation:* The moved VM on B is now activated. Post-migration code runs to reattach the device drivers to the new machine and reuses the IP addresses.

2.4.4 Memory Compression for Faster Migration

Memory compression is a technique that allows a radical decrease of the amount of necessary data to be transferred. This solution is described in detail in [16] and similar solution is actually used in Hyper-V Windows 2012 R2/2016 or VmWare vSphere 6.5 VMotion technologies. The solution described by Jin [16] and similar solution is actually used in Hyper-V Windows 2012 R2/2016 or VmWare Sphere 6.5 VMotion technology. Solution described by Jin [16] deals with the analysis of memory pages data granularity and the choice of the compression algorithm suitable for specific memory page. WKdm [18] and LZO [17] algorithms were were selected and tested for the memory compression. The memory compression scheme and the whole migration process are depicted in Figure 2.14.

An additional work published by Gustafsson [19] discusses the use of the universal common compression tools such as gzip, which is not suitable for this solution. There is a more advanced approach to the memory compression that uses a highly-sophisticated adaptive compression method [20].

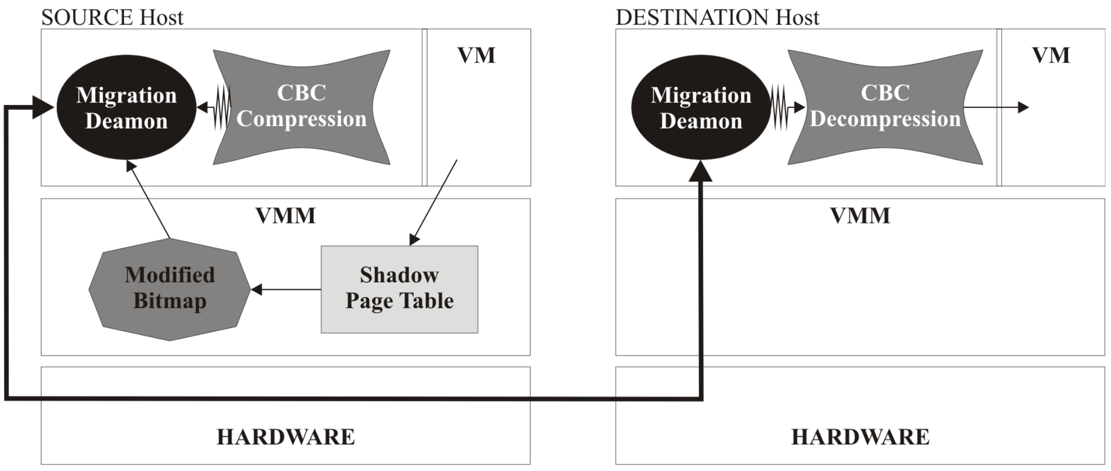


Figure 2.14: Memory compression improvement. The suitable algorithm selection can reduce the migration time about tens of percent.

Memory Compression Benefits

Jin [16] proved in practice that using an efficient compression mechanism, a live migration time can be reduced by about 30% and the amount of transmitted data can be reduced up to 60%. This time reduction also includes the necessary compression time. Gustafsson's work provides comparable results to the Jin's.

2.5 Energy Consumption Saving in Data Centres

The distributed virtualization systems, discussed in previous Section 2.3.1, are now used in all commercial data centres all over the world. The main blocks of such systems are formed by the virtualization nodes (VN), network area storages (NASes) and management nodes that are interconnected via a high-throughput communication network. The virtualization nodes are the high-performance computers, on which the virtual machines are executed. Network area storages serve for storing the virtual machines hard disk drive images. A separate storing of virtual images from the virtualization nodes is necessary for their efficient live migration **??**. The management node serves as a central coordinator of all actions in such a distributed system. A simultaneous poorly coordinated virtual machine execution or stopping can cause overloading, as some of the virtualization nodes can contain many running virtual machines. On the other hand, some virtualization nodes can contain only a small amount of running virtual machines and then they can be underloaded. Running such virtual machines can be much more expensive than it is acceptable, because they can run on other, not overloaded, virtualization nodes and the underloaded nodes can be further hibernated or switched off after the latest virtual machine migration. High energy consumption of the virtualization nodes has a direct impact on the amount of produced heat. The heat production also affects the activity of the data centre air conditioning system, which fundamentally participates on total energy consumption.

Many papers on this topic have been published recently. The first work was introduced by Nathui and Schwan [21] who proposed the virtual power management approach allowing setting various power management policies. Stoess [22] introduced the framework for the energy management for the modular operating systems, which contains another model for the virtual machines consolidation. Verma [23] proposed architecture of the power-aware application consolidation framework – pMapper. This framework allows setting the various scenarios for the virtual machine utilization. The main goal of this framework lies in power minimization by a fixed performance requirement. The relations between the energy consumption and resource utilization were studied by Shrikantaiah [24]. The direct relation between the power consumption and CPU utilization was in detail studied by Fan and more developed by Beloglazov [26]. Beloglazov further created a complex classification and the overview of various virtual machine consolidation algorithms [27] and provided their comparison. applied the reinforcement learning technique to improve the virtual machine consolidation efficiency. Cao [29] introduced a framework with specific heuristics for the minimum power and maximum utilization policy, which deals with the service level agreement (SLA) for the specific virtual machine deployment. Another similar solution was proposed by Dupont [30].

The optimization for virtual machine consolidation based on the improved genetic algorithm was published by Xu [31], another approach based on the ant colony optimization algorithm was proposed by Feller in [32]. Similar works were presented by Mills [33], Bobroff and Borgetto [35]. used the firefly optimization algorithm and introduced a novel migration technique. Real implementations of some algorithms exist for the Xen hypervisor - EnaCloud [37], the implementation for the hypervisor KVM is described in [38].

2.5.1 Energy consumption modelling

The total energy consumption of some virtualization servers is a sum of the consumption of independent parts of the server. The study [39] published by Intel labs tries to shows in detail the consumption ratio between all parts, which affects the entire consumption. The overview is depicted in Figure 2.15. The data shows that the highest consumption is apparent at the CPU, but the consumption at other devices like memory, NIC or disk drives is not lower.

The study published by Fan [25] proves that the increasing consumption is related to the CPU utilization, while the power at other devices (memory, NICs, hard drives) remains the same. The CPU supports various operation modes (low, middle, high) affecting the CPU frequency, which leads to the power increase. This fact can be expressed by the

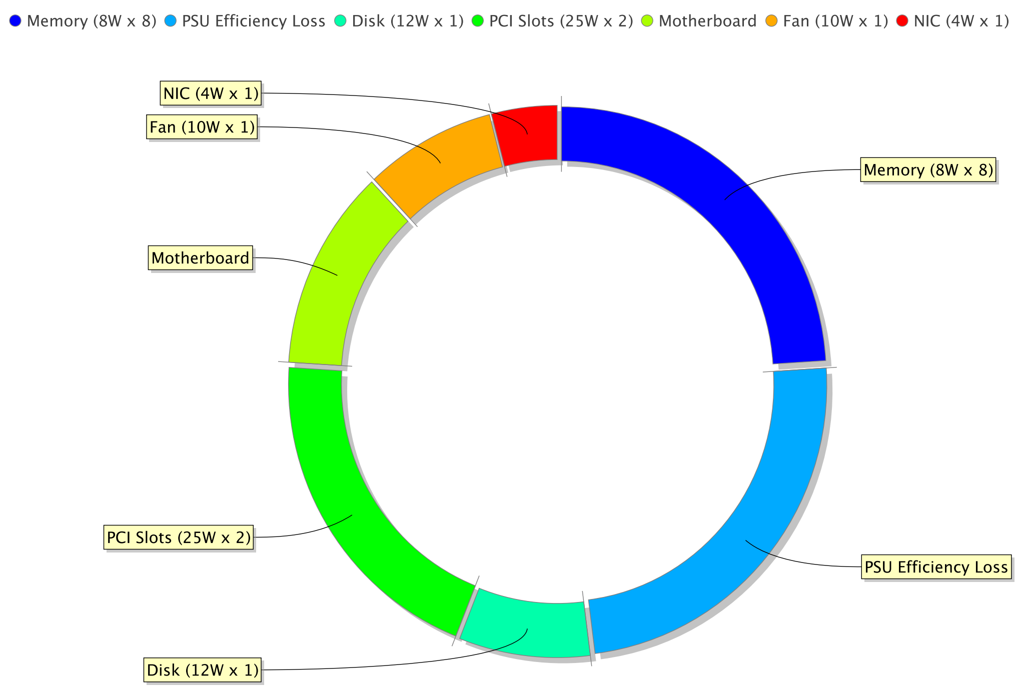


Figure 2.15: Detailed overview of consumption in a common virtualization server.

following equation:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) * u, u \in (0, 1) \tag{2.1}$$

where P is the estimated power consumption, P_{idle} is a power consumption by an idle server, P_{busy} is the power consumed by the server when it is fully utilized, and u is the current CPU utilization. Beloglazov and Buyya [27] reformulated this equation to another empirical model, which can be written as:

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) * (2u - u^r), u \in (0, 1) \tag{2.2}$$

where r is a calibration parameter that minimizes the square error and has to be obtained experimentally. For each class of machines of interest, a set of calibration experiments must be performed to tune the model. The accuracy of both models strongly depends on a concrete piece of hardware and the r parameter.

2.5.2 Virtual Machines Consolidation Problem

A large distributed system for the virtualization contains mostly hundreds or thousands of virtualization nodes. The solution implemented e.g. by VmWare or vSphere is depicted in Figure 2.16. The coordination is managed by the central management nodes. None of these solutions was described in detail. Beoglazov [56] implemented a new free solution OpenStack Neat, which was dedicated only for OpenStack clouds.

The reason for the infrastructure optimization consists in energy consumption. Every virtualization node needs a specific amount of energy for running, even though its utilization is near the zero. The basic intuitive idea behind is to consolidate all virtual machines running on this infrastructure into a smallest subset of virtualization nodes. The remaining nodes can be hibernated or switched off.

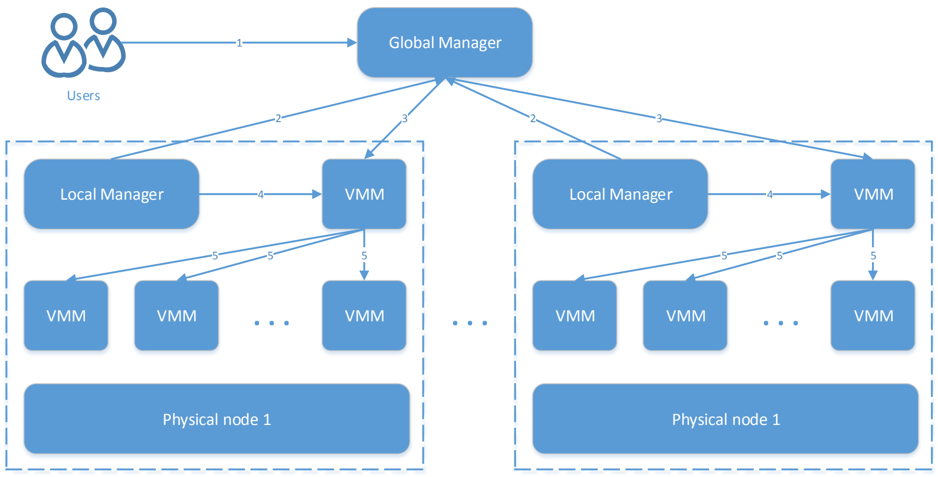


Figure 2.16: Consolidation principle used in modern virtualization systems.

Further aspect, which has been many times discussed in connection with the cloud computing, is the service level agreement (SLA) of the provided services. It guarantees a minimal availability time of service (e.g. web server). Its violation is penalized. The SLA can be affected by the migration of a virtual machine.

The consolidation problem can be divied into following parts:

1. *Underloaded nodes detection.* It means to identify all nodes that are not utilized very often. The mechanism for the system underload detection will be described further in Section 2.5.3.

2. *Overloaded nodes detection.* The situation is contrary to the underloaded node detection. The goal is to detect the excessively utilized nodes. The overloaded nodes can tend to slowdown some executed virtual machines.

3. *Virtual machines selection and placement.* 3. Virtual machine selection and placement. Once the underloaded/overloaded node has been detected, it is necessary to find the virtual machines that run on the underloaded/overloaded nodes and migrate them to other possible nodes.

The options to infrastructure optimization that can be used for the direct consumption decrease are as follows:

1. *Virtualization node switching on/off.*

2. *Virtualization node hibernation/wake up.*

3. *Virtual machine live migration.* , i.e., the transfer of some virtual machines to another virtualization node without stopping them.

2.5.3 Algorithms for Distributed Virtualization System Consolidation

2.5.3.1 Underloaded or Overloaded Node Detection

Reliable node state detection is a complicated process. All the methods, which will be discussed later, are based on some heuristic rules. The question, whether the node is underloaded or overloaded, is similar. The node state is related to the degree of the CPU utilization. Most algorithms are based on the previous collection and statistical evaluation of the CPU data utilization. These algorithms are called on-line algorithms, because their behaviour cannot be computed in advance.

Static CPU Utilization Threshold

The simplest state detection algorithms are based on setting a CPU utilization threshold distinguishing the normal, underload and overload states of the host. The executed algorithm compares the extent of current CPU utilization of the virtualization host to the previously experimentally detected value. When the value is reached, the algorithm recognizes whether the host is underloaded or overloaded.

Median Absolute Deviation (MAD)

The fixed ranges of a utilization threshold cannot be used for the system with dynamic or unpredictable workloads, in which different types of applications share the same resources of a virtualization node. This paragraph describes a heuristic algorithm used for the auto-adjustment of the utilization threshold, which is based on a statistical analysis of historical data collected during the lifetime of the virtualization node.

The algorithm uses a statistical method which is more effective than the commonly used methods for the data containing outliers or the data coming from non-normal distributions. The adaptive threshold algorithm adjusts the value of the CPU utilization threshold according to the strength of the deviation of the CPU utilization. It can be explained by an observation that a higher deviation increases the probability of the CPU utilization that reaches 100% and causes SLA violation.

MAD is a measure of statistical dispersion. It is more robust estimate of the scale than the sample variance or a standard deviation, as it behaves better with the distributions without a mean or variance, such as the Cauchy distribution. MAD is a robust statistics which is more resistant to the outliers in a data set than the standard deviation. In the standard deviation, the distances from the mean are squared, which leads to large deviations being, on average, weighted more heavily. It means that the outliers may significantly influence the value of the standard deviation.

For a data set $X = \{X_1, X_2, ..., X_n\}$, MAD is defined the median of the absolute deviations from the median of the data set.

$$MAD = median_i(|X_i - median(X)|) \tag{2.3}$$

An adaptive U threshold of the CPU utilization can be counted as follows:

$$U = 1 - a * MAD, a \in \mathbb{R} \tag{2.4}$$

where a is a system specific parameter which defines how strongly the system tolerates the host underloads/overloads.

Interquartile Range (IQR)

The interquartile range is also called the middle fifty and refers to the measure of statistical dispersion. It is equal to the difference between the third and first quartiles.

$$IQR = Q_3 - Q_1 \tag{2.5}$$

The interquartile range is a robust statistics, having a breakdown point of 25%, and thus, is often preferred to the total range. For a symmetric distribution (such that the median equals the average of the first and third quartiles), a half of IQR equals the median. The representative value of an IQR range can be calculated by using the median or average. Using IQR, similarly to the CPU utilization, U threshold is defined as follows:

$$U = 1 - a * IQR, a \in \mathbb{R} \tag{2.6}$$

where a is the same parameter as in the previous method 2.5.3.1.

Local Regression (LR)

The LR method simply models localized data subsets to build up a curve that approximates the original data. The weight function W(x) expresses neighbourhood weights using the tricube weight function. The detail description is presented by Cleveland [57].

$$W(x) = \begin{cases} (1-|x|^3)^3, & \text{if } |x| < 1 \\ 1, & \text{else} \end{cases} \tag{2.7}$$

Let $\Delta i(x) = |x_i - x|$ be the distance from x to x_i, and let $\Delta_{(i)}(x)$ be the distances ordered from the smallest to the largest. The neighbourhood weight is defined by the function $w_i(x)$, where q is the number of observations in the data subset localized around x.

$$w_i(x) = (\frac{\Delta i(x)}{\Delta i_{(q)}(x)}) \tag{2.8}$$

Let x_1 is the last observation and x_k is the kth observation from the right side. For the problem under consideration x_i satisfies $x_1 \leq x_i \leq x_k$. That means $\Delta_i(x_k) = x_k - x_i$ and $0 \leq \frac{\Delta_i(x_k)}{\Delta_1(x_k)}$. The weight function can be simplified as $W'(u) = (1-u^3)^3$ for $0 \leq u \leq 1$, and the weight function can be calculated as follows:

$$w_i'(x) = W' * \frac{\Delta_i(x)}{\Delta_1(x)} = (1 - (\frac{x_k - x_i}{x_k - x_1})^3)^3 \tag{2.9}$$

The algorithm uses a trend line function $g(x) = a + b * x$, which is found for each new observation. The trend of the line is used to estimate the next observation $g(x_{k+1})$.

The overloaded system is detected under the following conditions, where a is the parameter for a specific system.

$$a * g(x_{k+1}) \geq 1, a \in \mathbb{R}. \tag{2.10}$$

2.5.3.2 Virtual Machine Selection

Once an overloaded virtualization node has been detected, it is necessary to select the machine, which will be migrated to another node. The following section contains three basic approaches.

Random Selection (RS)

The simplest approach consists in the random selection of a virtual machine. If all virtual machines are homogeneous, this solution is fully acceptable.

Maximal System Utilization (MSU)

The previous solution cannot be effectively used in the systems, where virtual machines are not homogeneous. The optimization is possible by using the system utilization (SU) value. The SU of each virtual machine can be calculated according to Equation 2.11. The utilization is expressed by the sum of standalone performance components.

$$P = \alpha * P_{CPU} + \beta * P_{RAM} + \gamma * P_{HDD} + \delta * P_{NET}, \alpha, \beta\, \gamma, \delta \in \mathbb{R} \tag{2.11}$$

$\alpha, \beta\, \gamma, \delta$ are the real weights. All virtual machines are sorted descendent according to their P and the machine with the maximum P is selected to migration.

Minimum Migration Time (MMT)

The main goal of this approach is to select the virtual machine, whose time is the lowest. The migration time T_{mig} depends on the size of the virtual machine operating memory and an available network bandwidth N_{BW}.

$$T_{mig} = \frac{VM_{RAM}}{N_{BW}} \tag{2.12}$$

Maximum Correlation Similarity (MCS)

The method is based on the idea that the higher the correlation between the resource usage by virtual machines running on the overloaded server, the higher the probability of the server overloading. The solution was described first by Verma [23].

The correlation between the virtual machine system utilization is calculated across all virtualization nodes. The metric is represented by the Pearson correlation coefficient.

The correlation between a pair of applications with time series $X = \{x_1, x_2, ..., x_N\}$ and $Y = \{y_1, y_2, ..., y_N\}$ can be expressed as follows:

$$r_{xy} = \frac{N * \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N \sum x_i^2 - (\sum x_i)^2} * \sqrt{N \sum y_i^2 - (\sum y_i)^2}} \tag{2.13}$$

All virtual machines are sorted via their r_{xy} and the first is selected for the migration.

2.5.3.3 Virtual Machine Placement

This problem is formally known as the bin packing problem. When the virtual machines are packed in a virtualization host, their total memory requirement could decrease due to the pages shared by the VMs, which should be stored only once. If the items can share the space in arbitrary ways, the bin packing problem is hard to even approximate. However, if the space sharing fits the hierarchy, as in case of memory sharing in virtual machines, the bin packing problem can be efficiently approximated. This problem is a combinatorial NP-hard problem, therefore it can be solved by using heuristics.

Beoglazov in [27] introduced the new algorithm 2.3. All virtual machines are sorted descendent according to their utilization and allocated to one host that provides a minimum increase of the power consumption caused by the allocation.

Algorithm 2.3 Best fit decreasing algorithm (BFDA)

Input: hostList, vmList
Output: vmPlacement

```
 1. sort vmList desecendent to their utilization
 2. for vm in vmList do
 3.     minPower=MAX
 4.     allocatedHost=NULL
 5.     for host in hostList do
 6.         if (host has enough resources for vm == true) then
 7.             power ← estimatePower(host, vm)
 8.             if (power <minPower) then
 9.                 allocatedHost ← host
10.                 minPower ← power
11.     if allocatedHost != NULL then
12.         add (allocatedHost, vm) to vmPlacement
13. return vmPlacement
```

2.6 Conclusion

This chapter deals with the necessary introduction into the hardware virtualization principles, data centre architectures, virtual machine off-line and live migrations and possible energy consumption optimizations of the virtualization infrastructures. An appropriate selection of virtualization technologies and their optimum settings fundamentally affect the virtualization architecture performance. Understanding these principles and algorithms played a key role in the development of a new virtualization testing tool. A novel live migration algorithm was created and a new consolidation principle, described in detail in the following chapter, was developed.

Chapter 3

Overview of Our Approach

The following section is organized into four sections focused on virtualization technology efficiency, live migration of virtual distributed systems, virtual machine consolidation and a novel design of a distributed virtualization system and its implementation.

The organization structure respects the evolution of the CMU virtualization system. The first section, which is dedicated to the virtualization technology efficiency, discusses the size of virtualization technology overhead.

The second section introduces a new approach to the distributed virtualization system migration that tries to move the virtual machines among various physical virtualization nodes. The migration for all virtualized machines is fully transparent, i.e., no virtual machine should be affected by this procedure.

The third section deals with a new way how to effectively decrease the energy consumption of a large distributed virtualization system. The solution is based on the virtual machine consolidation, using only a minimum part of the virtualization infrastructure.

The fourth section presents the design and implementation of the virtualization system that utilizes the principles discussed in the previous chapters and sections.

3.1 Distributed Virtualization Systems Efficiency

The main aspect in the using and deployment of the virtualization technology is its efficiency. The substance of this technology is as follows. The resources of one powerful physical machine can be shared among many simultaneously executed virtual computers. The efficiency of the virtualization technology is an important factor, which determines the possible count of the simultaneously executable machines. Another aspect, also fundamentally affecting efficiency, is the type of the applied virtualization technique. The main virtualization types are full virtualization, hardware-assisted virtualization, paravirtualization, partial virtualization and emulation, all mentioned in Section 2.3.1. Nowadays, full virtualization technology is predominantly used. In the following text, the term "virtualization" will always mean the hardware-assisted virtualization technology.

Distributed System for Virtualization

A typical distributed system for virtualization consists of three main parts – management node(s), computing node(s) and shared data storage node(s). This architecture is graphically represented in Figure 3.1. The management node creates, executes, stops and migrates the virtual machines on the virtualization hosts. The virtual hard drives of the virtual machines are stored on the network area storage. The virtual machines are hosted on the nodes in the middle.

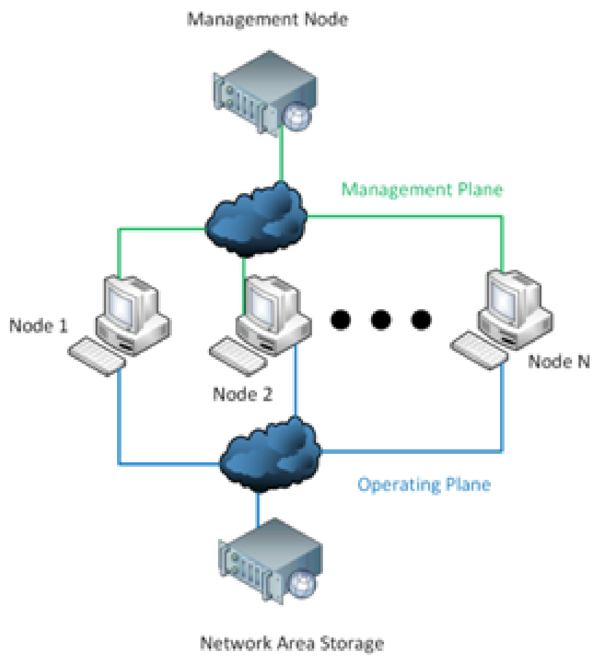


Figure 3.1: Distributed virtualization infrastructure scheme.

Overview of Existing Solutions

The common benchmarking techniques for standalone real computers were developed many years ago and are commonly used in many benchmarking tools. The benchmarking tools especially designed for the virtualization technology testing are used much less. The most famous commercial current benchmarking tools are SPECvirt_sc 2013 [40] or VMmark [41]. For academic or scientific purposes were developed tools such as vCosolidate [42], XenMon [43], XenoProf [44] and Perfctr-Xen [45] which were assigned only for the Xen hypervisor virtualization efficiency measurement [46]. Other tools such as GrenchMark [47] or C-Meter [48] were introduced for the grid/cloud architectures, but none of them is able to quantify the virtualization efficiency. A new complex tool for virtualization technology testing is called Virt-B [49].This tool measures the virtualization efficiency using a new approach – a three-layer benchmarking methodology. The biggest problem of this solution lies in its implementation. The framework is written in Java. It uses the Java Virtual Machine (JVM) sandbox for running. The implementation of JVM depends on the operating system and significantly affects the measurements. Besides the above mentioned tools, there were published many methodologies, see [50] [51], strategies [52] [53] or reviews for better virtualization overhead measurement understanding [54] [55].

3.1.1 CloudEvBench - Distributed Virtualization Efficiency Testing Tool

Our team has worked out a new testing tool which is able to measure the system efficiency. The solution was developed for the maximum measurement precision. There are many factors which can fundamentally affect the measurement accuracy. The proposed solution tries to eliminate these factors. The next advantage stands in the automated distributed system architecture, which allows finding out the virtualization efficiency through all virtualization system nodes. The solution has been optimized for the Linux and Windows operating system. The new command line style benchmarking tool Evbench have been developed, all tests in this tool have been designed for the maximal measurement accuracy. This tool it is possible to download from its GitHub repository.

3.1.2 Virtualization Efficiency Modelling

A simple, but realistic model of a common computer, valid for the performance modelling, consists of one or more single/multi-core CPU units, RAM operating memory, input/output devices (hard drive, CD/DVD drive, etc.) and a network card. The system performance (P) can be expressed as a four-dimensional vector of the values, see 3.1, which represent performances for specific virtual machine operations (CPU, RAM, I/O and NET).

$$P = (P_{CPU}, P_{RAM}, P_{IO}, P_{NET}) \tag{3.1}$$

P_{CPU} represents the CPU performance that is mostly measured in the CPU operations per second, P_{RAM} means the system memory read/write throughput measured in bytes per second. P_{IO} is the performance of the input/output devices, the measured value is device-dependent, e.g., the bytes per second of the read/write operations of the hard drive. P_{NET} performance represents the achievable throughput between two machines through the network medium, which is mostly measured in the bits per second.

3.1.3 Virtualization Efficiency Metrics

Virtualization efficiency E_{VT} can be computed as the ratio of the performance of a virtual computer (P_{VPC}) and the real computer (P_{RPC}).

$$E_{VT} = \frac{P_{VPC}}{P_{RPC}} \tag{3.2}$$

3.1.4 Overhead of Virtualization Technology

Overhead O_{VT} is the part of real computer performance necessary for running the virtual computer that can be counted according to this formula.

$$O_{VT} = 1 - E_{VT} = 1 - \frac{P_{VPC}}{P_{RPC}} \tag{3.3}$$

The performance of the real (P_{RPC}) 3.4 and virtual computer (P_{VPC}) 3.5

$$P_{RPC} = (P_{CPU_{RPC}}, P_{RAM_{RPC}}, P_{IO_{RPC}}, P_{NET_{RPC}}) \tag{3.4}$$

$$P_{VPC} = (P_{CPU_{VPC}}, P_{RAM_{VPC}}, P_{IO_{VPC}}, P_{NET_{VPC}}) \tag{3.5}$$

3.1.5 Performance of the Distributed Virtualization System

The architecture of the distributed system 3.1 consists of N computing nodes. The aggregate performance of such a distributed system (P_{DS}) can be counted according to the following equation. P_i is the performance of a real single node.

$$P_{DS} = \sum_{i=1}^{N} P_i \tag{3.6}$$

Because all computing nodes of the distributed system are physical, the equation can be rewritten into the following form:

$$P_{DS_{RPC}} = \sum_{i=1}^{N} P_{RPC_i} \tag{3.7}$$

where P_{RPC_i} is the performance of the single physical computer. $P_{DS(RPC)}$ is the total performance of the distributed system, which consists from the physical computing nodes only.

The set of virtual computers that can be executed on the distributed virtualization system contains K virtual computers, at maximum. All resources of the real computing nodes of the distributed system are assigned to the virtual computers. P_{VPC_i} is the performance of a single virtual computer. The performance of whole virtual computers $P_{DS_{VPC}}$ can be counted as follows:

$$P_{DS_{VPC}} = \sum_{i=1}^{K} P_{VPC_i} \tag{3.8}$$

The efficiency E_{DS} of the distributed virtualization system can be computed as the ratio of the performance of its real and virtualized infrastructures.

$$E_{DS} = \frac{P_{DS_{VPC}}}{P_{DS_{RPC}}} = \frac{\sum_{i=1}^{K} P_{VPC_i}}{\sum_{i=1}^{N} P_{RPC_i}} \tag{3.9}$$

3.1.6 Accurate Virtualization Measurement Aspects

Virtualization technology testing can be affected by various aspects that might cause high inaccuracies in the results. To obtain accurate results, it was necessary to comply with the measurement conditions that gradually appeared during our testing tool development. Let's discuss them in more detail.

Sameness (on the Instruction Level)

This requirement seems to be a bit strange, but it is very important. Using various compilers, power efficiency of the compiled program can differ. If the application, running in the native environment, is compiled by a different compiler than the application in the virtualized environment, there could arise inaccuracy in power of about 10-40%. It is caused by the settings or quality of the compiler code generator. The difference was observed between the compilers from different producers and between different versions of the same compiler, as well. Using the same benchmark binary code further means that the results (their absolute values) obtained from the Windows cannot be directly compared to the results obtained from the Linux without subsequent normalization. Another important factor is the same version of all additional system libraries, dynamically loaded by benchmark software, because their implementation can differ, too. This fact leads to using the exactly same system for the virtualized and native environments (with the same version of the operating system kernel and libraries). It can cause a performance difference varying between 5-10%, depending on the operating system. The last most important factor lies in the implementation of specific functions and libraries used by benchmarking software compilation. For example, the implementation of commonly used function memcpy() strongly differs in Linux glibc library, when compared to the implementation in MS Visual Studios. This factor may cause the differences in measurements in hundreds of percent.

Same Hardware Computing Performance

This requirement is logical, but it is not met every time we need. The problem lies in modern multi-core CPU architecture. The main measurement inaccuracy can be caused by the “Turbo boost” that allows a short-term power increase in some CPU cores. If the measurement, which is performed in the virtual environment, runs on the CPU core(s) with the increased computing power, the power of the virtual computer may seem greater than the physical one. This factor causes the measurement inaccuracy of above 50% (depending on a CPU type). The solution is to switch off the turbo boost feature during the measurement phase.

System Resource Utilization

If the measured system does not have enough free computing resources, i.e., free CPU cores, memory or unsaturated I/O devices, other background processes can affect benchmark performance. The inaccuracy depends on a concrete state.

Virtualizer Support by the Virtualized Operating System Kernel

Virtualized operating system performance is affected by its virtualizer support. This aspect does not directly cause the measurement inaccuracy, but affects the virtualization efficiency.

Native Execution of a Compiled Benchmark

The binary form of the executed benchmark must be directly executed by the operating system without the compatibility emulation. The problem can arise, for example, when a Win32 program is emulated in the Win64 environment.

3.1.7 System Utilization Modeling

The average utilization of the system is also the important factor in the virtualized system deployment, because the partial utilization allows multiple resource sharing among virtual computers. The lower utilization means higher available amount of deployable virtual computers. The entire system utilization is given by the partial utilization of the CPU, RAM, I/O bandwidth and network bandwidth. System snapshot (SS) parameters are as follows.

SS = {CPU utilization [%], Memory usage [%], Hard drive usage, Network usage}

1. *CPU utilization [%]:* refers to the time of the total time in which the CPU works.
2. *Memory usage [%]:* refers to the amount of operating memory used for the data storage.

3. *Hard drive usage:* refers to the read/write data. A maximum value of the data throughput depends on a concrete drive.

4. *Network usage:* refers to the specific network bandwidth utilization. The relative usage is recalculated to the maximum link speed of the network card.

3.1.8 CloudEVBench Architecture

The above described parameters are measured periodically and further statistically interpreted by various statistical methods.

The benchmark system is determined for testing performance of the distributed virtualization system, whose architecture was presented in Figure 3.1. The testing tool divides the testing process into two parts. In the first part, real infrastructure performance is tested; in the second part the performance of the virtualized infrastructure is tested. All tests and evaluations are executed automatically. The main parts of the system are:

1. *benchmarkServer* is the tool that is installed on the management node. It has the capability of executing scripts remotely on all nodes. It enables completing the central control of the test preparation, operation and measured data collection for different testing configurations. The implementation of this tool is primarily intended for the Windows platform only, but it is fully functional on the Linux, using the Wine tool.

2. *benchmarkClient* is the tool that is installed both on every physical node of the distributed virtualization system and also on every virtualized computer. Its implementation was realized for the Linux and Windows. It is able to communicate with the benchmarkServer by using the standard BSD sockets.

3. *resultsAnalyzer* is a statistical tool that evaluates the data collected by the benchmarkServer.

Possible Benchmarking Requirements

The developed tool CloudEVBench was used for the result measurement. It has a code optimization for the Linux (g++) and Windows compilers (MS Visual Studio). For the development of these tools, it was necessary to exactly know the testing principles and their evaluation. Another requirement was that the benchmarking tool must be executable from the command line and its results must be machine-workable.

CloudEVBench Network Topology

The benchmarking process is carried out as follows. The first tool to start is the benchmarkServer. It runs on the management node and sends multicast datagrams with the settings for benchmarkClient connected to the network. The IP address of the management node can be assigned dynamically and the whole benchmarkClient must be able to obtain correct settings. The situation is depicted in Figure 3.2. There are two types of communication between the client and management node. The first communication type provides the client parameters received from the management node. The second one provides a standard communication from the client to the management node. The first type is realized by the multicast packet sending and receiving, while the second type is a simple unicast TCP connection between the client and management node.

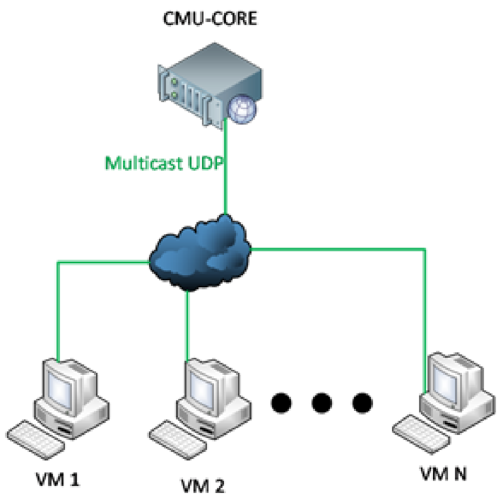


Figure 3.2: CloudEVBench supposed network topology.

The testing process starts after connecting all clients to the management node. Then the management node sends the testing commands for the given measurement simultaneously to all connected clients. The received commands are immediately executed by the client operating system.

3.1.9 Testing Scenarios

The CloudEVBench tests the virtualization infrastructure according to the following scheme. The physical infrastructure is tested as first – for obtaining the reference values, the CloudEVBenchClient must be installed on all physical nodes of the infrastructure. The configuration of the entire system must be stable and mustn't be modified during the test. The virtual infrastructure is tested in various system configurations (VPCC), which are defined as follows.

$$VPCC = \{cpuCounts, ramSize, diskDriveSize, netCardEnabled, OperatingSystem\}$$

- *cpuCounts* is the number of the physical CPU cores assigned to the specific virtual machine.
- *ramSize* is the size of the operating memory assigned to the specific virtual computer.
- *diskDriveSize* is hard drive capacity on the shared network area storage assigned to the specific virtual computer.
- *Operating system (OS)* used for testing can be determined by the following attributes.

The performance of an operating system can be affected by various factors, which is determined by the specific configuration. The operating system can be characterized by this configuration.

$$OS = \{Platform, architecture, usedfilesystem\}$$

- *Platform:* refers to the used operating system - Windows, Linux.
- *Architecture:* 64-bit system architecture.
- *Used File System:* is platform-dependent, can influence the efficiency of I/O disk operations.

Testing Configuration

The CloudEVBench tool was deployed on the CMU computing system. This system consists of four nodes now. The computing nodes contain 2x Intel Xeon 2660v2 CPUs, 128 GBs operating memory, 10 Gb Ethernet card and 2x raid mirror connected 1TB hard disks. These nodes physically execute and emulate the virtual computers. The network area storage node is based on the Intel i5-4440 CPU, with installed 32 GBs operating memory, an 18TBs Hardware Raid 5 hard drive (6x 3 GBs SATA III Seagate Constellation, NAS type drives). This node stores all hard drives of the virtualized computers. The management node is a computer with lower performance, which is sufficiently in compliance with our requirements - it contains only 1x Intel Xeon e5405 CPU and 6GBs operating memory. The node was called CMU-Core. The network architecture is divided into the management and operating planes. The management plane is used for the infrastructure administration; the operating plane is strictly used only for mapping the virtual computer hard drives and for the mapping and migration of the virtual machines. The situation is depicted in Figure 3.1. The used virtualization technology was the Microsoft Hyper-V running on the Microsoft Windows Server 2012 R2. Real testing efficiency was targeted to the CPU, RAM, hard drive throughput and the network interface throughput.

3.1.10 Testing Results

CPU Virtualization Efficiency and Overhead Measurement

Basic idea of this test is the CPU utilization. The task to solve by the CPU is to search for prime numbers in a given interval. This task can be divided into many computing threads. The evaluation metric was the time, which was necessary for the task completing. The testing process was provided by the Sysbench tool. The overhead value was calculated according to Equation 3.3; the reference time value was the execution time value of the task which was performed with the same parameters on the physical system. The results are graphically depicted in Figure 3.3. The average measured overhead value was about 1.1%, which means that the virtualization efficiency was 98.9%. This result can be achieved on the Intel platform only by using VT-x technology t rapidly increasing the virtualization efficiency.

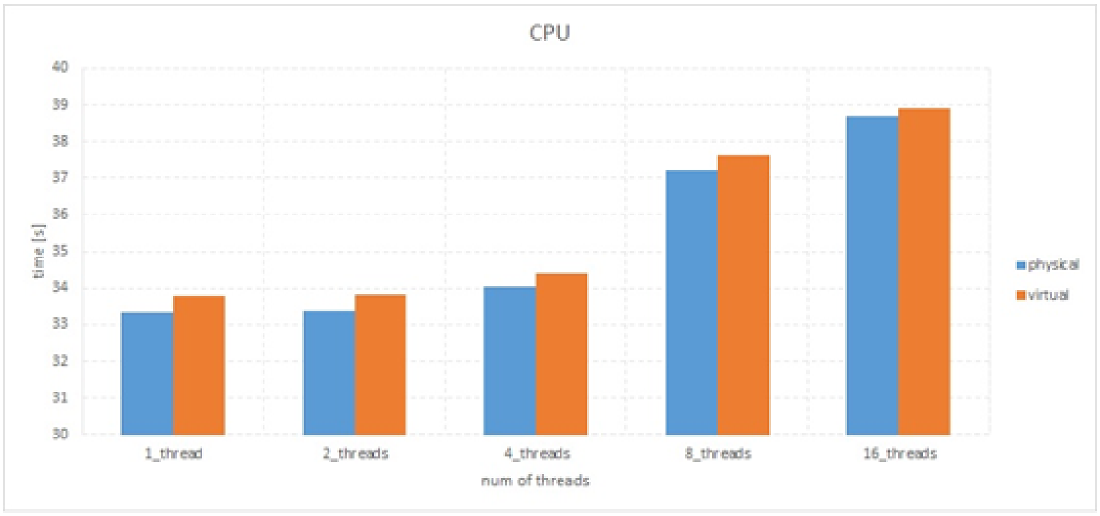


Figure 3.3: Virtualization efficiency testing (VT-x enabled). Measured efficiency was about 99%.

Operating Memory Virtualization Efficiency Measurement

This test was performed by reading/writing various large data blocks from/into the operating memory. The test proved that the maximum data throughput could be achieved by increasing the higher number of threads working in parallel. The evaluation metric is the memory bandwidth utilization in MB/s. The average measured overhead value was about 2.04%, which corresponds to 97.96% virtualization efficiency. This result is achievable by using the Intel VT-d technology allowing the virtual machine direct access to the physical operating memory. The results are graphically depicted in Figure 3.4.

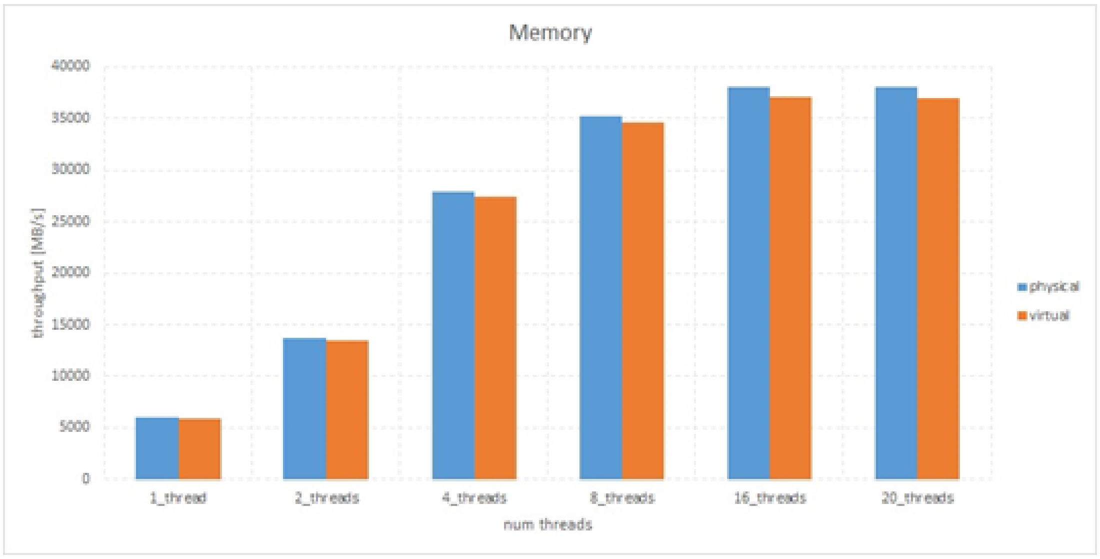


Figure 3.4: Memory virtualization efficiency testing (VT-d enabled), the measured efficiency was over 97%

I/O Virtualization Efficiency and Overhead Measurement

The I/O testing lies in measuring the reading/writing data from/to the hard drive. The metric is the reading/writing throughputs. The measurement was targeted on the single-thread sequential reading/ writing from/to the physical hard disk with the SATA III connection. The measured overhead value was about 4.42%, which means that the virtualization efficiency was better than 95%. The results are graphically depicted in Figure 3.5.

Network Virtualization Efficiency and Overhead Measurement

The network testing was realized as a bandwidth measurement between the local and remote network sides connected via the 10 Gbit/s Ethernet switch. The remote side was represented by a hardware router without any virtualization and the local side was first represented by the native computer and later by the virtual computer. The maximum bandwidth saturation is possible only by using simultaneous TCP connections. The measured average overhead was only 1.29%, which means the 98.71% virtualization efficiency. The results are graphically depicted in Figure 3.6.

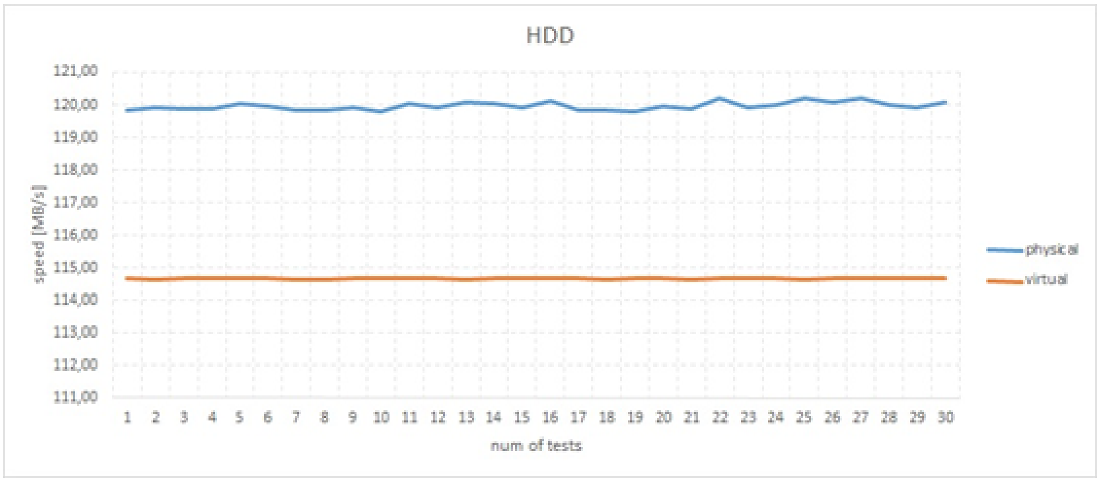


Figure 3.5: I/O devices (HDD) virtualization efficiency testing, measured efficiency was over 95%.

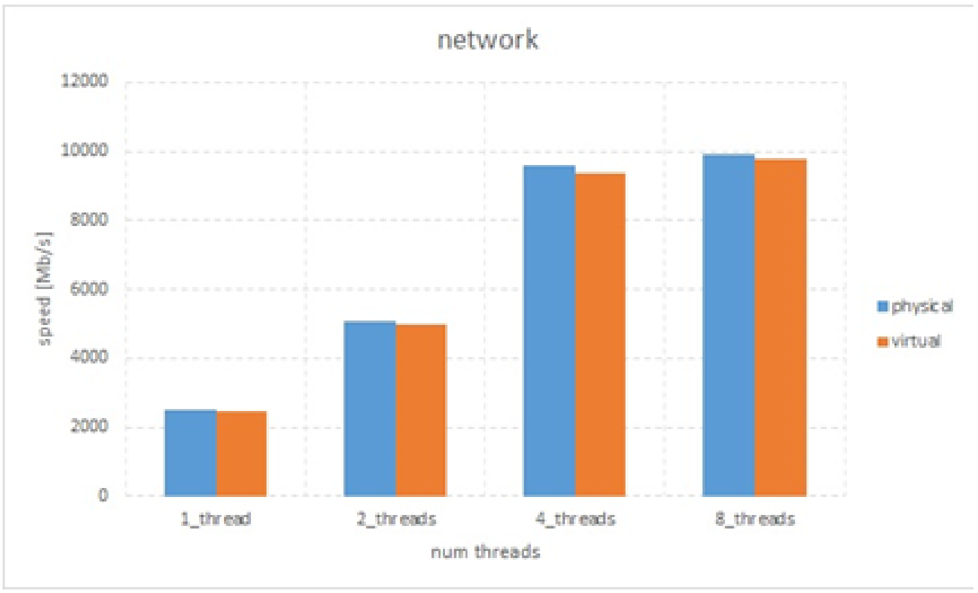


Figure 3.6: Network virtualization efficiency testing, measured efficiency was over 98%.

3.1.11 Conclusion

This work contains the description, development and deployment of the CloudEVBench tool, including a new command line benchmarking tool EVbench, which is dedicated to the accurate virtualization efficiency measurement. Its aspects were defined during the development. In case these aspects are not fulfilled, it will result in the measurement inaccuracy that will be higher than the virtualization efficiency. The testing was performed for the Linux and Windows operating systems and the relative overhead results are very similar for both environments.

The obtained results proved that the performance of the virtualized system could be similar (maximum measured overhead was only 4.4%) to the physical system that manages the virtualization process. The average system utilization was further measured on various systems. Both the extent of the system utilization and virtualization efficiency has high impact on the real number of simultaneously executable virtual computers. All results in this work were measured in real. They proved high performance of the advanced virtualization technologies.

3.2 Live Migration of Virtual Distributed Systems

The live migration technique is a very hot topic today in connection with the virtualization technology which is widely used in different computing environments from the single processor computers to the large cloud solutions and, at present, data centres. Live migration of the given virtual machines brings about some aspects which can cause troubles in comparison with much easier off-line migration. We have studied current architectures used for live migration and their optimizations for more efficient migration between various physical hosts. A specific part of our research involved the studies related to the live migration of the group of virtual machines representing either cooperating cluster nodes or a distributed computing system. From the point of view of their complete live migration, they have similar behaviour. Basic aspects and principles of migration functionality are discussed in the following section, where a new live migration algorithm, suitable for an automated live migration of various distributed architectures as a whole, is presented.

3.2.1 Distributed Computing Model

Distributed computing is a special science discipline dealing with processing complex computing problems that cannot be processed by a single standalone computer (for example processing very large data sets). Message communication between the cooperating nodes of the distributed system is achieved mainly by message passing. The state of a distributed computing system [58] is defined as a set of computer nodes with their own memory (containing computing processes) and a set of messages which were sent but were not delivered yet. The messages from the communication channels directly affect the computing processes by memory page changes on an appropriate computer node. For the live migration of a given distributed system, it is necessary to guarantee also the delivery of such communication messages.

Client Server Model Live Migration

The migration of the client-server model should ensure the migration of one virtual server node and many virtual client nodes from their own physical hosts to other physical hosts. This computing model is often used in many parallel software architectures [59], such as mpi, open mpi, mpich, lam/mpi and various computing cluster solutions like MOSIX, openMosix, Kerrighed or OpenSSI. All previously mentioned architectures can be run in the virtualized environment and can be used for distributed computing. Live migration of the server node is much more difficult in comparison to the client node because the server node memory content is modified by the messages from all client nodes. The migration of the client node is easier because only one server node sends messages to this client, i.e., fewer messages modify a client memory content.

Peer to Peer Model Live Migration

All individual computing nodes in peer to peer systems are equivalent. An average number of communication messages should be equal on all nodes. A set of algorithms for the group coordination exists in the distributed computing. Such a set also works on p2p topology, but it needs a coordinator [60]. Typical examples of these algorithms are the algorithms for entering critical sections, leader election or replication. This fact means that the number of communication messages of some nodes is higher.

3.2.2 Virtual Distributed System Live Migration Problems

In the following sections, we will discuss some problems that can have an impact on the migration process. Their elimination is important for the safe migration process and it represents the principle of the scheme, which will be introduced at the end of the work.

Free Resources of Physical Hosts Involved in Migration

The virtual distributed computing system is usually represented by tens, hundreds or thousands of virtual machines. Complete migration of such an amount of virtual machines requires successful allocation of all needed resources to the destination computing environment. This allocation is done during the pre-migration and reservation phases and must be done dynamically. Some aspects of the migration suitability will be discussed.

Low Throughput of the Communication Network

The network throughput between the source and the destination physical node can be very low in comparison with the throughput requirements for the necessary virtual machine memory updates. It can cause live migration impossible, because the difference in the memory content between the migrating and migrated machines remains the same, or even gets larger. It must be detected in the pre-migration phase. If is not possible to allocate the necessary bandwidth, the migration cannot be executed.

Message Loss during the Migration Phase

A current global state of a virtual distributed system is defined as a set of virtual machine memory contents and the contents of all communication channels (messages sent, but not delivered). If a message (packet) is lost during migration, a current global state is corrupted. The live migration technique should avoid the message loss, i.e., the migration is then transparent. None of currently used hypervisors contains the techniques for the message loss correction.

Network Bandwidth Saturation during the Migration Phase

The given virtual distributed system is represented by a set of virtual nodes. Virtual nodes can migrate in parallel or in series. Parallel migration is faster but, still, a migration of a large number of virtual nodes can easily cause network bandwidth saturation (typically gigabytes of memory content per one node migrate). This leads to a lack of availability of some services on the network and other problems. More information can be found in [61].

3.2.3 Live Distributed Architecture Migration Scheme

A new migration scheme, which is suitable for a very large data centres, proposed in this paper consists of three main parts: network topology optimization, destination host selection, and a live migration process. In our concept we assume that the migration process is managed only by one central coordinator.

Network Bandwidth Optimization

There are three types of network traffic in the virtual distributed system, see the situation in Figure 3.7. The first one is the communication traffic (CT) among all the nodes of the distributed system, typically produced by passing the messages. The second type of the traffic is generated by the migration of virtual machine memory content (VMMCT) from one physical host to another. The third type is generated by reading/writing from/to the virtual hard drive network area storage (NAST). CT and NAST are generated during the whole lifetime of the virtual distributed system. VMMCT is generated only during the migration process, but it is much bigger in comparison to CT or NAST. VMMCT can cause bandwidth saturation and has a negative influence on further traffic. A theoretical solution is to divide the network bandwidth into three sub-bandwidths, i.e., three separate (virtual or physical) network circuits, where each circuit is dedicated only to a specific traffic type. Another solution consists in sharing one band that can be divided and allocated through a particular attribute of the service-providing protocols.

3.2.4 Selection of Destination Physical Hosts

A typical distributed system can contain hundreds (or more) of virtual computing nodes distributed among the set of physical nodes. To determine a new distribution of the set of virtual machines among various sets of physical nodes is a very complicated task. A typical reason for the reallocation of the set of virtual machines is the necessity of releasing their allocated resources, e.g., the changes in the hardware equipment, etc. This reallocation problem is solved by the following algorithm 3.1. The simplest solution is to migrate all given virtual machines of the given computing system from their current physical host to the new one at once, but this is usually not possible, since the computing capacity of the selected destination physical node is not sufficient.

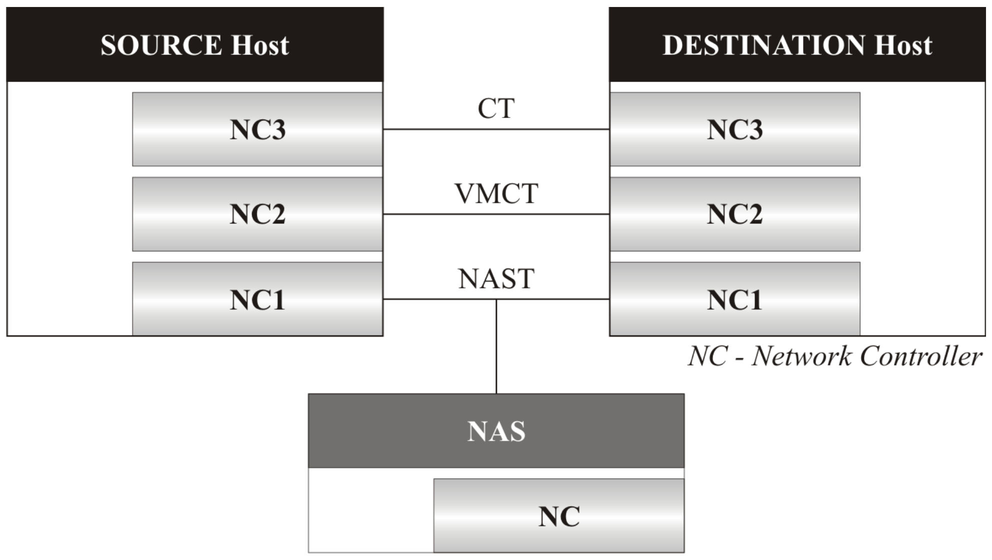


Figure 3.7: Traffic type in the distributed virtualization architecture.

A central coordinator C manages the migration process. The virtual distributed system uses the message passing for communication purposes. We assume that the probability p of the message being sent between any two nodes is the same and is independent of their physical location. S_1 is a set of source physical nodes, S_2 is a set of destination physical nodes. $DS(V, S_1, ID)$ is a distributed system with a set of virtual machines V, running on S_1 with a specific identification (ID). All virtual machines are the members of the same computing system. The ID is independent of the physical nodes.

S_1 may have same common items with S_2. S is a set of all physical nodes. $S = S1 \cup S2$. $M(S_1, S_2)$ is an operation migration which enables transfers of all virtual machines running on S_1 to S_2. $N(S_1)$ or $N(S2)$ is the number of physical nodes contained in S_1 or S_2. For a distributed system, let's make $M(S_1, S_2)$ with minimal $N(S_2)$. It is possible to achieve minimal $N(S_2)$ in three steps.

The algorithm 3.1 finds (if exists) the node-count optimal solution, which will be a part of the pre-migration and reservation phase. (If the solution doesn't exist, the set S_2 must be modified.). The destination node selection principle is depicted in Figure 3.8.

Algorithm 3.1 Destination selection algorithm (DSA)

1. *Discovery:* C sends message A with an ID of the virtual distributed computing system to all the nodes of S_2 and obtains back message B from each node. Message B contains free resources. If at node X, which sent back a message B to C, some virtual machines are running as a part of the distributed system with a specific ID, message B contains the number of free resources incremented by the number of virtual machines with the same ID running on X.

2. *Selection:* C makes a list of resources from the incoming messages and sorts it in a descending order according to the number of free resources – the nodes with the highest number of resources will be used first. Also the list of requests of resources for the members of S_1 is prepared and sorted in the same order. is prepared and sorted in the same order. Then the verification, relative to the existence of the intended migration solution, is performed from the point of view of the resource, mapping the possibility between the two lists.

3. *Recursion:* If the solution exists, it is produced by C recursive selections from the lists comparing the needed number of resources until all items of S_1 are mapped onto S_2.

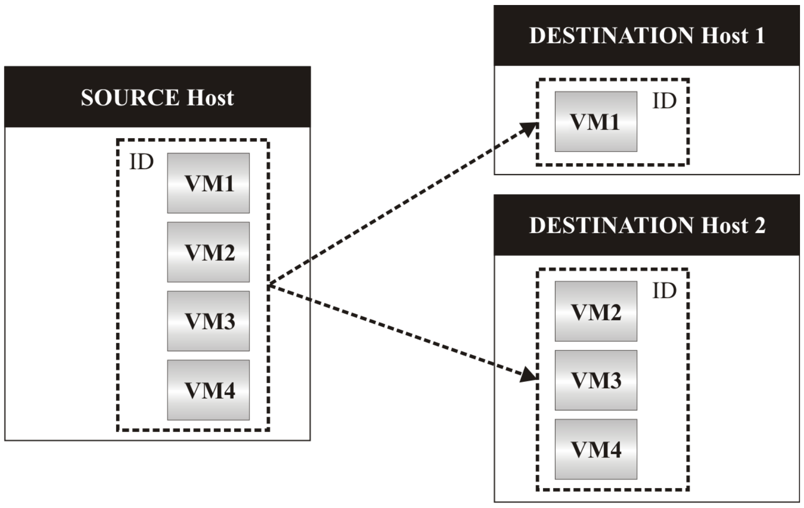


Figure 3.8: Node discovery for the possible migration. The items of the virtual distributed system can migrate to different physical nodes.

3.2.5 Distributed Virtual System Live Migration Process

In the previous section, we mentioned the resource allocation algorithm that serves as the input for the next phase. $DS(V, S_1, ID)$ is the current distributed system, running on the nodes from the set S_1 and $DS(V, S_2, ID))$, is a new distributed system which will be relocated from S_1 to S_2.

Virtual Remote Network Message Buffer

During the Stop and Copy phases, the currently migrating virtual machine can obtain one or more communication messages from other nodes. The virtual buffer stores these messages into the queue on the S_1 node and delivers them to the similar buffer assigned to the S_2 node. The main function of this buffer, The main function of this buffer, see Figure 3.9, is to save the messages which could be lost during the migration, and to enable the delivery of these messages to the destination node.

Algorithm for Migration of Entire Distributed System

The following method is used if we want to migrate the whole distributed system, part by part. The principle is depicted in Figure 3.9. V is a set of all virtual machines of a virtual distributed system. S_1 is a set of source physical nodes; S_2 is a set of destination physical nodes.

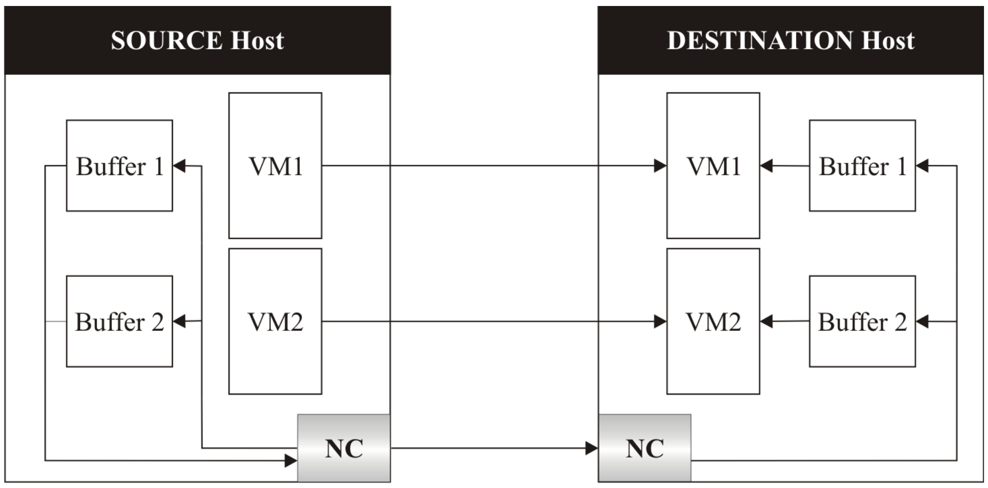


Figure 3.9: Distributed system live migration principle, where the solution uses the virtual network message buffers.

Algorithm 3.2 Distributed system live migration algorithm

```
1. Select k virtual machines from V, which are currently running on nodes from $S_1$,
   $0 < k < N(V)$, create a new empty set $U$, $U \in V$, insert all selected virtual machines
   from V into $U$ and remove these virtual machines from V.
2. For all items in $U$, start the pre-copy phase.
3. Before starting the migration process, create virtual network buffers on all nodes
   from $S_1$, where selected virtual machines are currently running. Message capturing
   on all nodes is disabled.
4. Create virtual network buffers on all corresponding nodes from $S_2$ for the virtual
   machines being migrated and interconnect this buffer to buffers created in stage 3.
5. Activate message capturing into virtual network buffers on all source nodes from S1
   and transmit this messages to virtual network buffers at destination nodes from $S_2$.
6. For all items in $U$, start the post-copy phase.
7. For all nodes in $U$, start the commit phase.
8. Start all virtual machines that have been migrated and deliver messages from their
   corresponding network message buffers.
9. Stop all virtual network buffers.
10. If V is empty, stop the migration process, else continue with the stage 1.
```

3.2.6 Proposed Algorithm Simulation Results

For the algorithm from the previous section, we upgraded a standard open source Xen hypervisor. The real simulation was done via the university multiprocessor cluster, consisting of four computing nodes, one network area storage and one management node. All these nodes were interconnected via the 10 Gb/s Ethernet network. Every computing node contained two physical 10-core Intel Xeon processors and 128 GB operating memory. On every physical node, the Xen (or the upgraded Xen) hypervisor starts up 10 dual-core virtual machines with the Linux Debian operating system.

System Transparency Testing

Migration transparency means the impossibility of the migrating distributed system to detect a migration process. The untreated migration can manifest itself by the increasing packet loss.

The first simulation for the transparency verification was as follows. The same computing process (tasks including the network communication) was executed on all virtual nodes. All executed processes communicated via a non-packet loss correcting a communication protocol. The graph in Figure 3.10 shows that our implementation significantly improved the packet loss during the migration phase. We tested the migration process for the minimum and maximum Ethernet MTU size to eliminate the network media influence.

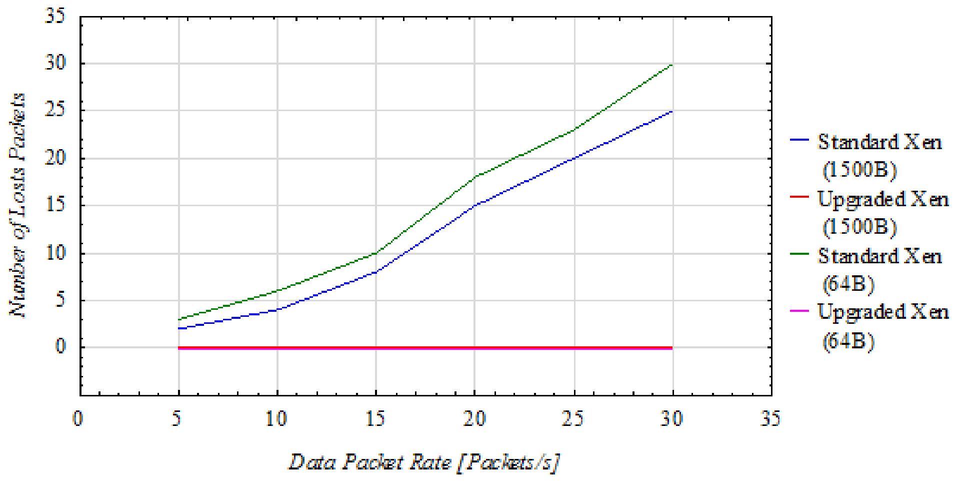


Figure 3.10: Packet loss, measurements for 1500 and 64 byte MTU.

System Downtime Measurement

The next simulation depicted in Figure 3.11 shows the average downtime for the migrating virtual nodes during the migration phase. In this case, our solution was a bit slower in comparison to the standard Xen solution due to the overhead of the virtual network message buffers.

Migration Time of the Distributed System

The measured migration time depends mainly on the number of migrating nodes and their operating memory size. For the migration of larger number of virtual nodes, it is possible to achieve network bandwidth saturation. Different colours of the lines in the graph depicted in Figure 3.12 represent various memory sizes of the migrating virtual machines.

Virtual Network Buffers Saturation

A full transparent migration process of the distributed system is realized by means of the virtual network buffers. Figure 3.13 shows the saturation process of the in/out buffers

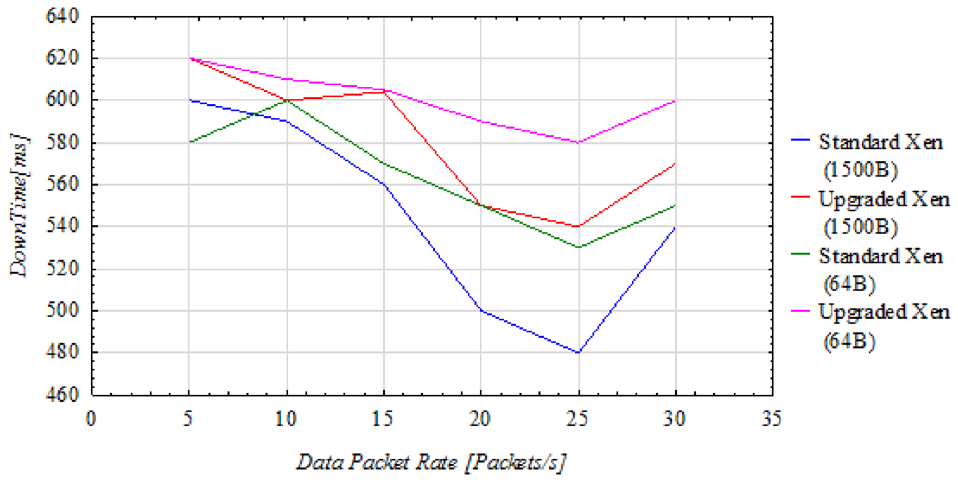


Figure 3.11: System downtime measurement for 1500 and 64-byte MTU shows that the proposed solution is only a bit slower than a standard solution.

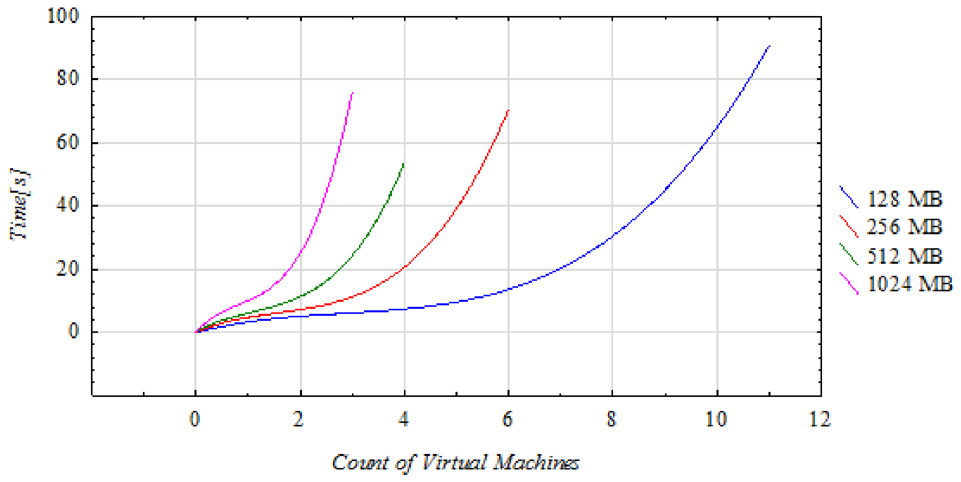


Figure 3.12: Count of virtual machines and their operating memory size influence the time of migration.

between the source and the destination node varying with time. This graph is taken as an average of values measured in Section 3.2.6 and shows a gradual change.

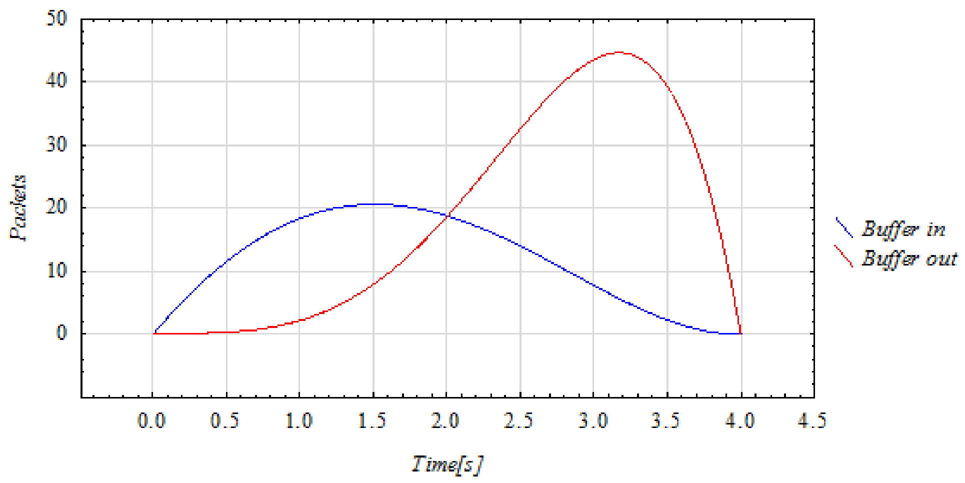


Figure 3.13: Network buffer saturation during the migration phase.

3.2.7 Conclusion

We dealt with basic aspects of distributed system migration, we proposed a scheme for their live migration, worked out a prototype implementation and made some evaluations. Our newly proposed migration scheme seems to be useful, i.e., in large data centres running many virtual machines that are interconnected with the virtual distributed systems.

3.3 New Approach for Virtual Machines Consolidation

The current state of the art of virtual machine consolidation was described in Section 2.5.3. Our newly proposed mechanism considers the heterogeneity of the virtualization nodes. The requirement for this feature is logical, because often happens that hardware of virtualization node groups differs, i.e., some virtualization nodes are more energy aware than the others. The main factor in the energy consumption is a global CPU utilization which is associated with the air condition system activity [26]. We introduced new metrics, which characterizes the node energy consumption efficiency. The total energy (TC) consumption value can be calculated as follows. The variable u is from the interval of 0-1 (0 means idle, 1 means fully utilized) and $P(u)$ is the global power consumption value for the concrete utilization.

$$TC = \int_0^1 P(u)du \tag{3.10}$$

The efficiency (E) of the virtualization node N, which has total K physical CPU cores, can be calculated as follows:

$$E_N = \frac{K}{TC} = \frac{K}{\int_0^1 P(u)du} \tag{3.11}$$

The function P(u) for the specific virtualization node is typically unknown. This can be experimentally measured and completed by the linear approximation. The CPU utilization can be increased between 0 and 100% by the 10% step. Equation 3.10 can be reformulated into this shape.

$$E_N = \frac{1}{0.1 * \sum_1^{10} P(\frac{i-1}{10}) + \frac{1}{2} * (P(\frac{i}{10} - P(\frac{i-1}{10})} \tag{3.12}$$

The consumption model is created for each virtualization node in the distributed system. Virtualization nodes (V) are then sorted descendent via their computed efficiency value. System utilization stress can be reached by using [63] or [64].

Algorithm Principle Description

The main goal of this idea is depicted in Figure 3.14. All nodes are sorted according to their efficiency. Every node contains the virtual machines consolidator (VMC). The VMC is a module directly cooperating with the local hypervisor and remote hypervisors. Another device interconnected to the VMC, is an electronic watt meter. The VMC periodically reads the current value of the power and knows its energy power consumption function course for its node. The value will be further used for the node state detection. The next steps are done simultaneously on all virtualization nodes.

Every virtualization node N knows its power consumption efficiency E value and knows all virtualization nodes that have worse E values. The connection to these nodes is depicted by the arrows between the virtualization nodes. If any node N with better E value is not overloaded, it can offer its resources to another node M with a worse E value. The offer from N to M contains the specification of N free resources - the number of free CPU cores and the size of free operating memory. If M has virtual machines that can run under the offered resources of N, all these machines are moved from M to N. N selects such an M node that has the worst possible E value. During the migration phase of the virtual machines, N and M must not accept the offers from other virtualization nodes. If M further contains no running virtual machines, it can be switched off or hibernated.

The solution can be further optimized. After the consolidation is finished, it could happen that some of the nodes with a high E value, but a low number of CPU cores, were fully utilized, in contrast to other nodes with a high number of CPU cores and lower E value. N therefore asks all other computing nodes M (with the worse E value and sufficient computing resources) to migrate all its virtual machines to M. M is selected from all nodes with the worse E value, but first are asked the nodes with better E value. If M has enough free resources, all virtual machines from N are moved to M and N can be switched off. During the migration phase of the virtual machines, N and M must not accept the offers from other virtualization nodes. This case can happen only if there are big resource differences between the virtualization nodes.

All steps of this solution are incorporated into the algorithm 3.3. The meaning of individual variables and the brief algorithm description are as follows.

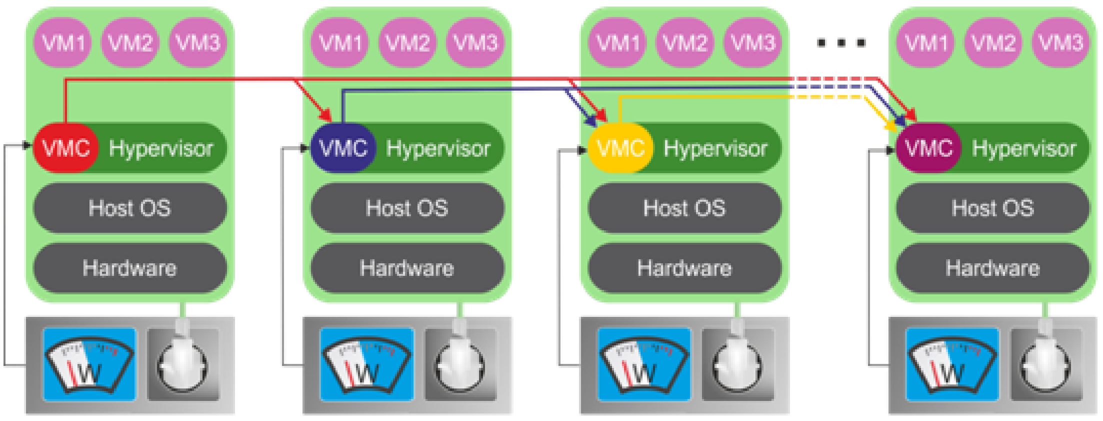


Figure 3.14: Consolidation principle of the distributed system virtual machines.

N is the set of all virtualization nodes. For all nodes $N_i \in N$ are known their energy characteristics $E(N_i)$, that means every node, N_i contains a list of nodes M, which contains nodes with the worse consumption characteristics than N_i. The nodes M_i from M are sorted ascendent according to their E. This algorithm is executed in parallel on all nodes. If some receiver has obtained more offers from many senders, the offer from the node with the highest E is selected.

Algorithm 3.3 Distributed virtual machines consolidation algorithm (DVMCA)

```
Sender ($N_i$):

1. for all items $M_i$ in $M$
2. send an offer of available resources (AR) from $N_i$ to $M_i$ (nodes with worse E are
   asked earlier)
3. wait for answer from $M_i$
4. send migration request to $M_i$
5. update resources of $N_i$
6. if $N_i$ has available resources, continue 2
7. else break

Receiver ($N_j$):

$N_j$ is a node of sender's $M$ and contains a set of virtual machines $VM_j$, which are running
on it.

1. receive available resources AR from $N_i$
2. select virtual machines for migration ($VMSM_j$) from $VM_j$
3. send answer to $N_i$
4. wait for the migration request from $N_i$
5. migrate all machines from $VMSM_j$ to $N_i$
6. remove all $VMSM_j$ from $VM_j$
7. if VM is empty, switch $N_j$ off
8. else
9. continue with 1
```

Nodes States Detection Policies

The main aspect for the virtualization node state detection is the energy power consumption measured by a locally connected watt meter. A proper absolute power consumption value (for the state detection) differs for each virtualization node and depends on its hardware configuration. The data is collected continuously and evaluated periodically. The absolute value of under/over load power consumption (for the state detection) must be measured experimentally and depends on the specific system utilization. These evaluation metrics such as MAD 2.5.3.1, IQR 2.5.3.1, LR 2.5.3.1 can be used for the node state detection. Another used possible policy was MEAN - the mean of all values in the latest monitored time interval.

Virtual Machine Selection Policy for Migration

The best-fit (BF) selection policy was selected for the virtual machine selection, i.e., the group of virtual machines utilizing the offered free resources at maximum, is selected. This approach is similar to algorithm 2.5.3.3.

3.3.1 Algorithm Evaluation

Measurements Specification

System measurements were realized by using the university virtualization system. The system consists of four various groups of virtualization nodes. All servers contain the Microsoft Windows Server 2012 R2 operating system with the Hyper-V hypervisor.

The following server categories were used:

1. *SuperServers* – 2 physical CPUs Intel Xeon v2, with 20 CPU physical CPU cores, 128 GBs RAM, 10 Gbit/s Ethernet LAN connection, 2x Xeon Phi 5110P computing card, based on SuperMicro technology

2. *Servers* – 2 physical CPU Intel Xeon v3, 12 CPU cores, 32 GB RAM, , 10 Gbit/s Ethernet LAN connection, 2x Xeon Phi 5110P computing card, based on SuperMicro technology

3. *Old Servers* – 2 physical CPU Intel Xeon v1, 8 CPU cores, 16 GB RAM, Asus technology, 1 Gbit/s Ethernet LAN connection

4. *Swap Servers* – 1 physical Intel i5 CPU, 16 GB RAM, assembled from the parts of various manufactures

Such a set of computers was used to create heterogeneous environment. The images of all virtual machines were stored on the network area storage, with the 10 Gb/s LAN connection. The power consumption data was measured by the EnerGenie PWM-LAN electronic watt meter and the data was collected by using the in-house developed application EnergyMeter.

Energy Consumption Measurements

The energy consumption model was experimentally measured for all virtualization nodes in our system. Measurement results were consistent in all server categories and can be very well reproduced. Figures 3.15 - 3.18 illustrate the energy consumption of each server group. The system idle interval determines the minimum consumption power of the system without the stress. Virtualization nodes with Xeon Phi computing cards have a higher idle power consumption value, which can handicap these nodes by their efficiency value computation. It can be solved in Equation 3.12, where K variable is substituted by K+L and L is the number of additional CPU cores.

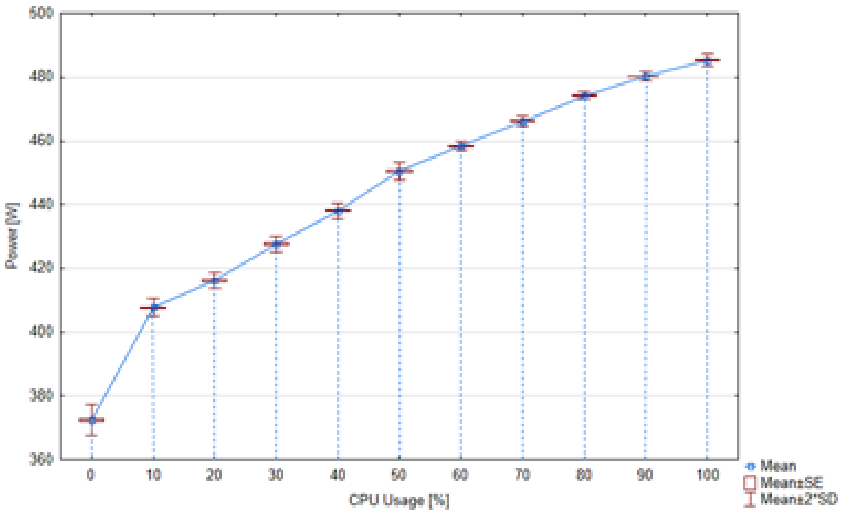


Figure 3.15: Super server consumption evaluation model.

The performance models showed the differences between all virtualization nodes. The Old Server Nodes have the same number of CPU core as the Server Nodes. The Server Nodes contain two additional Xeon Phi computing cards and the power consumption is similar. The air condition system of the Old Server has the similar power consumption as its physical CPUs. The comparison of all energy consumptions between all virtualization node groups is depicted in Figure 3.19. The minimum idle state consumptions were between 70-260 Watts. The consumption to performance ratio comparison among all server types is presented in Figure 3.20. The ratio is counted from the concrete server CPU consumption and CPUs result in the performance testing tool. For our purpose, the PassMark Software CPU benchmarking utility was used.

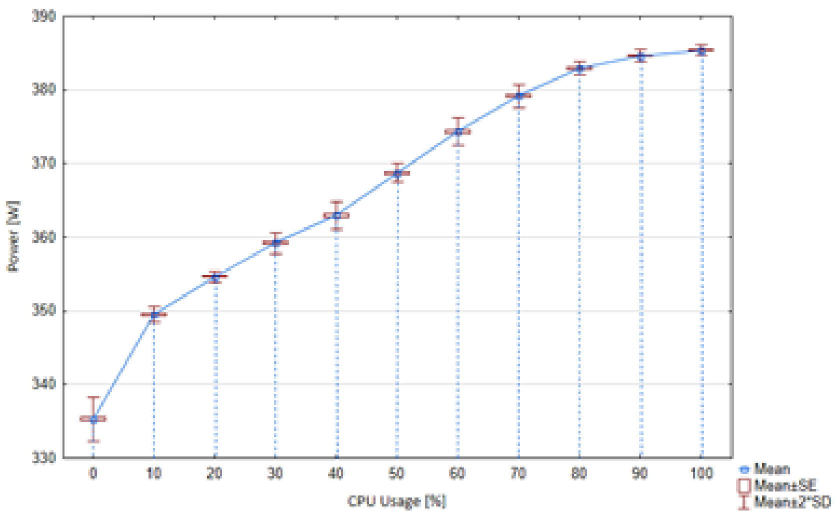


Figure 3.16: Common server consumption evaluation model.

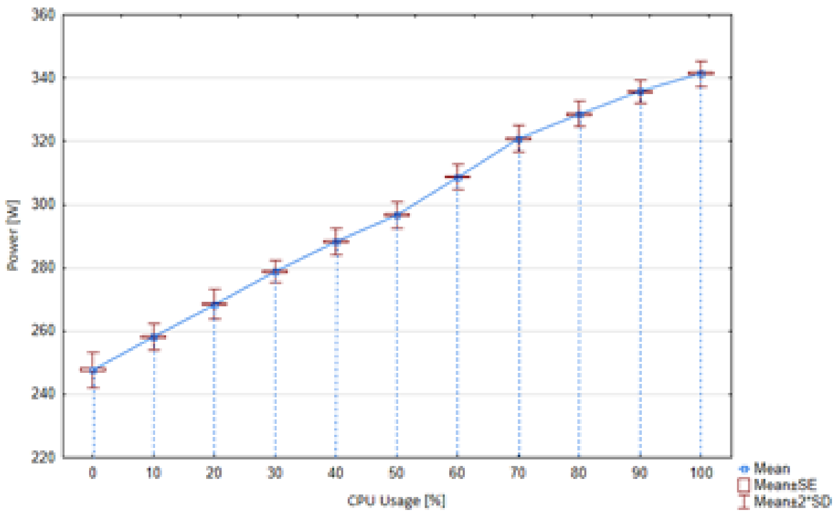


Figure 3.17: Old server consumption evaluation model.

Experimental Algorithm Evaluation

The evaluation testing scenario was performed as follows. We prepared the group of hundreds of virtual machines (with exactly the same virtual hardware configuration, 2 CPU cores and 4 GBs RAM, with the same 64bit operating system Windows 10 Education). The total energy consumption was computed as the sum of values that were measured by the standalone electronic watt meter. Every virtualization node energy input was monitored

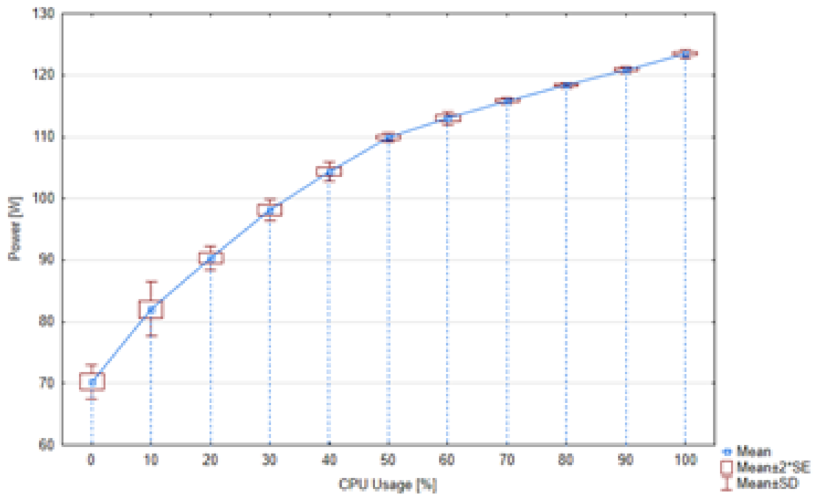


Figure 3.18: Swap server consumption evaluation model.

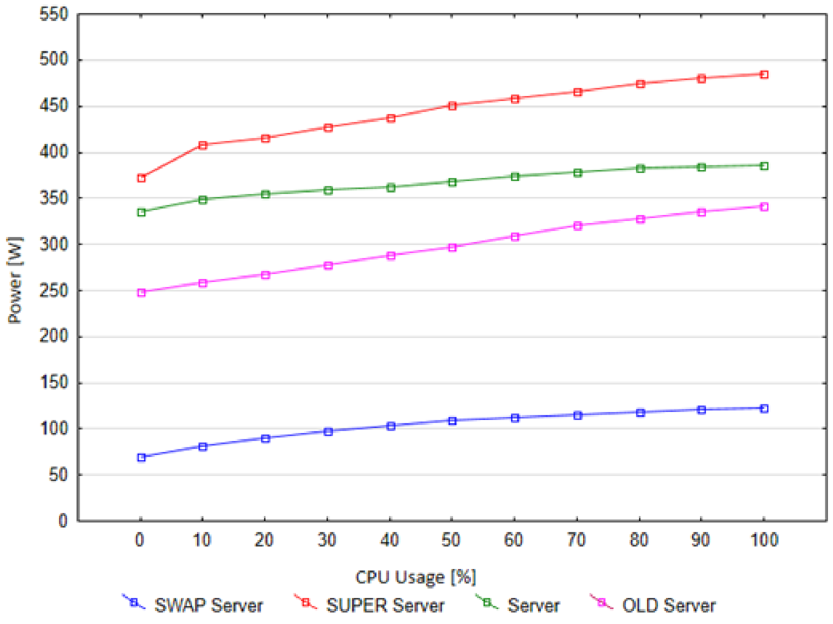


Figure 3.19: Power consumption comparison (all server groups).

by its own device. One additional device was connected to the central network switch. All virtual machines were 6x automatically executed or stopped in the specific order by the virtualization system. One measurement batch took about one hour. The virtual machines were placed equally between all running virtualization nodes. The reference energy

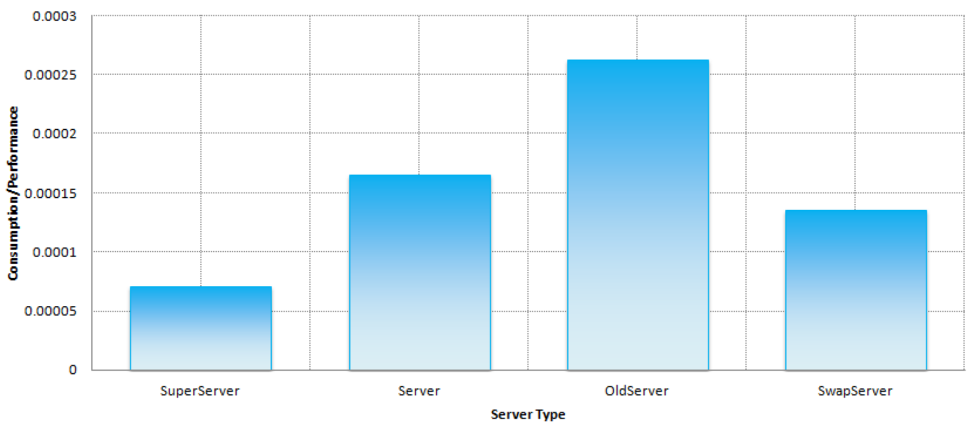


Figure 3.20: Consumption power ratio of all used server types.

consumption value was the total power consumption of the uncoordinated system, i.e., the virtual machines were still running on the same virtualization nodes and no virtualization node was switched off by its idle time.

The next measurement was similar, but the idle nodes were switched off. The last four measurements were performed by using the proposed algorithm. It means that the virtual machines were consolidated and idle nodes were switched off. Each state evaluation was performed periodically every five minutes. The data was processed by using the MEAN, MAD, IQR and LR evaluation metrics. The results are depicted in Figures 3.21 and 3.22. It is necessary to say that our results are relevant to our specific system configuration and depend on the number of changes of virtual machine states.

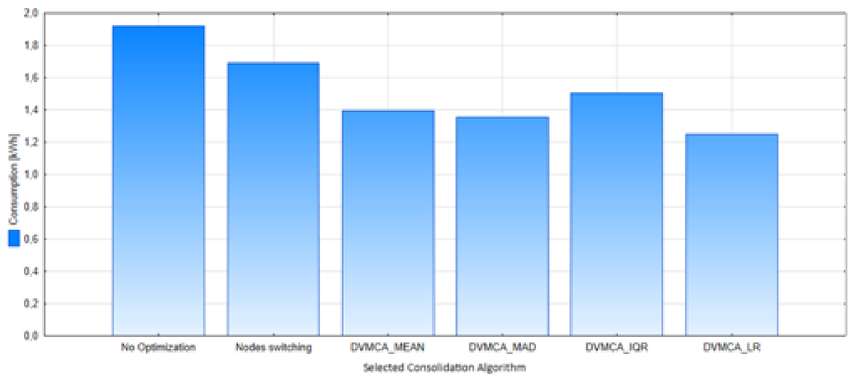


Figure 3.21: System energy consumption by the specific consolidation algorithm selection.

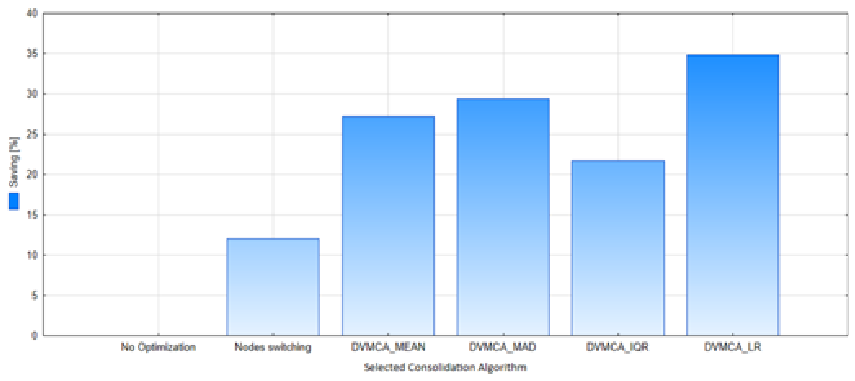


Figure 3.22: Energy consumption saving by the specific consolidation algorithm selection.

3.3.2 Conclusion

There was introduced a brand new algorithm for fully distributed virtual machine consolidation. It is able to consolidate the virtual machines by using the real measurements obtained from electronic watt meters. The power consumption value seems to be a more suitable factor than the CPU utilization used in other consolidation algorithms, because it has a smoother shape and changes fluently. Real energy consumption measurements respect various hidden additional aspects like the additional expanding cards, air condition system, etc., which further participate on the global system consumption. The testing was realized on a real distributed virtualization system and verified the abilities of the proposed solution.

3.4 Virtualization system CMU

This section provides a description of the virtualization system CMU, which was developed in the last three years. The system is able to manage the virtual machines which belong to the distributed virtualization infrastructure. The system is also able to provide the access to the end users. They can create, delete or execute the virtual machines via the web interface.

The system contains the executive and management parts, which together create the functionality. A concrete description of all parts will be presented in Section 3.4.2.

3.4.1 System Scheme and Motivation

The goal behind the system creation is as follows. Teaching some subjects requires specific software equipment and computing performance. Once the students start attending university, they obtain a personal virtual machine, which serves as a permanently available computer during their study.

Another reason to create the system lies in the utilization of supercomputer computing resources at a time, when they are not fully exploited. It happens mostly overnight, therefore it is advantageous to utilize the infrastructure for the scientific purposes.

The best to use is an automatic management scheduling module that automatically executes or stops the virtual machines with scientific applications.

3.4.2 System Architecture and the Main System Parts

The virtualization architecture scheme is illustrated in Figure 3.23. The system supports various numbers of virtualization nodes (NODES), multiple data storages and management nodes (CORES). The management nodes can communicate with each other within the infrastructure or in an adhoc mode; more can be read in Section 3.4.6.

System configuration

The current configuration of the virtualization cluster is presented in Tables 3.1 and 3.2. The first one depicts virtualization nodes and the second one deals with data storages. The CMU-CX is a group of computers that manages the entire infrastructure. More management nodes are necessary to ensure high availability. CMU-NODE-X and CMU-NODE-HX are two groups of virtualization hosts.

Both data storages contain hardware RAID 5 with the SSD caching option. The principle is described in Section 2.3.1.2. The current configuration allows executing tens of virtual machines per data storage (cca. 2 cores per virtual machine). The concrete number depends on the configuration of currently running virtual machines.

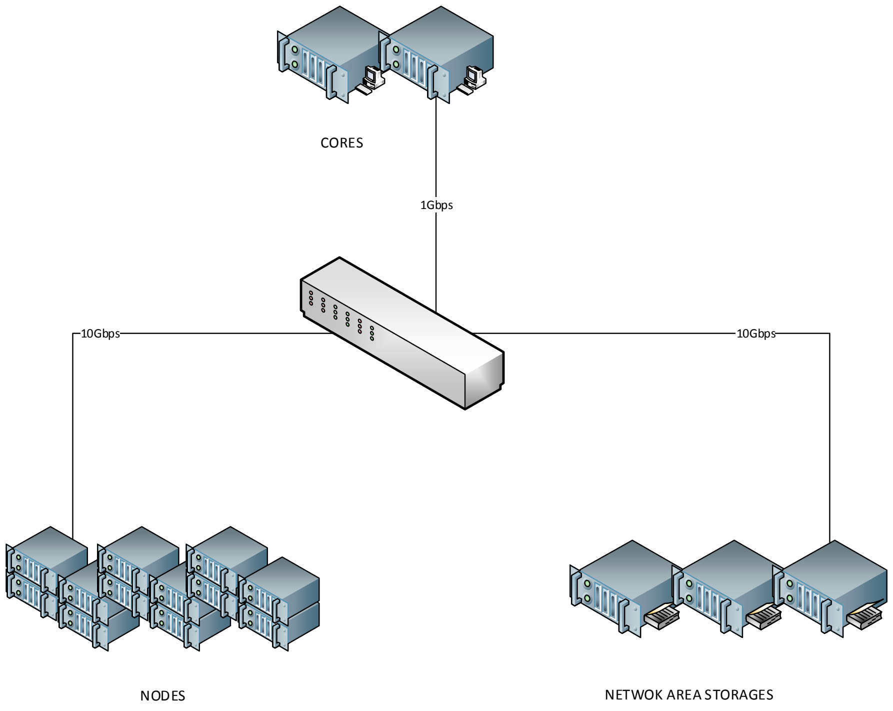


Figure 3.23: Virtualizaion system CMU, the principle scheme.

Table 3.1: System CMU virtualization nodes configuration

Unit	Hw. Configuration	Count	Hypervisor	Comment
CMU-NODE-1	40 CPU cores, 192 GB RAM	1x	Hyper-V	2x Xeon Phi 5110P, 10 Gbit Ethernet
CMU-NODE-2	40 CPU cores, 192 GB RAM	1x	Hyper-V	2x Xeon Phi 5110P, 10 Gbit Ethernet
CMU-NODE-3	16 CPU cores, 48 GB RAM	1x	Hyper-V	
CMU-CX	8 CPU cores, 8 GB RAM	2x	Hyper-V	
CMU-NODE-HX	16 CPU cores, 16 GB RAM	10x	Hyper-V	
CMU-NODE-TESTY	16 CPU cores, 16 GB RAM	2x	Hyper-V	

Table 3.2: System CMU data stores configuration

Unit	Store Capacity	RAID	NET	Comment
CMU-NAS-1	12 TB	HW RAID 5	10 Gbit Ethernet	SSD Cache Cade 2.0 PRO
CMU-NAS-2	10 TB	HW RAID 5	1 Gbit Ethernet	SSD Cache Cade 2.0 PRO

3.4.3 System Scenarios

The following section deals with the virtualization entities (objects) that were implemented in the system and are accessible to the users. Most common users prefer a simple scenario with normal virtual machines. Nevertheless, for teaching some subjects, it was necessary to create more sophisticated solutions.

Virtualization Entities

Five different virtualization entities were implemented in the system. They can be managed by the elevated users only. The list of entities is as follows:

1. *Normal Virtual Machine:* This is the basic configuration of a virtual PC that serves for a common user. A typical application can be the computer with a programming IDE.

2. *High-Power Virtual Machine:* In contrast to the normal virtual machine, it works with more assigned computing resources. A typical number of CPU cores is 2-3 times higher than that at the normal machine.

3. *Computer Group:* It is mostly a group of the homogeneous virtual machines located on various virtualization nodes. All virtual machines communicate within the same network segment, which is physically isolated from other networks.

4. *Computer Network:* It is similar to the previous solution. The difference consists in the interconnection of all nodes. In this case, all items can be placed behind different virtual switches or routers, therefore the area is not one fully transparent segment.

5. *HPC Virtual Machine:* These machines are by the configuration similar to the normal virtual machines. They differ in the interconnection of the virtual machines NICs with the Xeon Phi computing cards, using virtual network bridges. More detailed description is presented in 3.4.7.

Virtualization hosts

During the system development, there were implemented two models of the virtualization hosts. The first one was the standalone virtualization server with a specific hypervisor (e.g. Hyper-V or VMWare), the second one was the virtualization data centre. The virtualization data centre is the platform that automatically manages its computing resources and perform the commands given by the main management node.

3.4.4 User Roles and Virtualization Groups

The system contains the four basic user roles {Administrator, User (student), Privileged user (teacher), Test}. The difference lies in specific features presented in Table 3.3.

Table 3.3: System CMU virtualization user groups

User	Max. Virtuals	Virtual Cluster	Virtual Network	Xeon Phi available
Normal User	2	no	no	no
Privileged User	10	yes	yes	yes
Administrator	Unlimited	yes	yes	yes
Test	5	yes	no	no

Virtualization Groups

The virtualization cluster, containing specific virtualization hosts, is separated into several groups, some of them overlapping, to avoid the system resource exhaustion by a specific user group. The current CMU virtualization hosts are separated into three basic virtualization groups - CORE, HERMES and TEST. The group CORE consists of the CMU-NODE-X with the Xeon Phi cards and is intended for education. It is accessible only to the privileged users (teachers). The group CMU-NODE-HX (H means HERMES, an old computing cluster) is available to the student virtualization machines and the CMU-NODE-TESTY group serves for the infrastructure debugging and testing.

3.4.5 System Modules

Distributed Modules Architecture

The entire system is created by the cooperation of standalone system modules. The basic system module is the Core module (CM) presented in 3.4.5 that serves for the interconnection of other modules (in the infrastructure mode). All system modules have the similar architecture, but their concrete behaviour differs. Considering the system reliability, the main requirement was the asynchronous behaviour of all modules. It means that no module should be blocked if it performs a specific command. The commands from all modules can come unpredictably and the response interval is not explicitly guaranteed. A system module is connected to the CM by opening two independent TCP connections. The first connection serves for sending commands and the second one for their receiving. This solution further allows implementing different modules in various programming languages. The only condition is that the communication protocol must be the same.

Table 3.4: System CMU command structure

DEST. MODULE	SRC. MODULE	ID	MESSAGE TYPE	DATA

The received commands are served simultaneously by a specific thread. All commands are sent asynchronously. The commands that require the synchronous behaviour are substituted by asynchronous commands that contain a waiting part for the response. The response is identified by a unique identifier (ID). The message type can be a request (RES) or response (RESP). The RESP is identified according to the source RES and message ID. The general system command structure is presented in Table 3.4. The first two items identify the destination and source module.

Core (CM)

It is a basic system module that serves all commands asynchronously. Core serves as a system module registrar and module authenticator. Every command that comes through the system is processed by this module and can be monitored. The communication with this module is possible only by utilizing an authorized system module.

Authentication

This module serves to verify user's identity. The user authentication credentials can be verified by local user database, or by a remote LDAP server. The current implemented solution uses both methods.

Database (DBS)

The database module maintains all important settings of virtualization infrastructure, user privileges, virtualization groups, etc. The module is based on the MS/SQL or MySQL database systems. The database failure protection is realized by the database replication.

Infrastructure Controller (IC)

This module monitors the availability of all virtualization nodes and data storages. A suitable monitoring mechanism depends on the concrete virtualization platform. . For instance, the Hyper-V domain registered virtualization host can be monitored by the domain controller. The universal simplest solution can be realized by the host pinging, but it is not fully reliable. Some specific errors (e.g. the virtualization service failure detection) are complicated, because they are caused by an abnormal state of the system.

Another task of this module is the system resource utilization monitoring. This data utilization serves for a new virtual machine execution or placement. The possible states of the virtualization nodes can be {On-line, Off-line, Error}.

Virtual Machines Management Module

It is the one of the most important modules. This module manages the creation, deletion, execution, dynamic consolidation, etc., of all virtual machines. It communicates directly with the virtualization nodes, data storages or virtualization data centres. The migration of clusters is based on the simplified version of the algorithm 3.1. The module periodically communicates with the infrastructure controller to get the states of the virtualization nodes.

Another function is the detection of virtual machine states, i.e., reading their running configurations. The communication with the virtualization nodes is established directly. The entire system is currently adapted for the Microsoft Windows (Win64) platform, therefore the communication is based on the PowerShell module. The communication with the VMWare virtualization hosts or data centres is provided by the PowerCli PowerShell extension.

Energy Saving Module

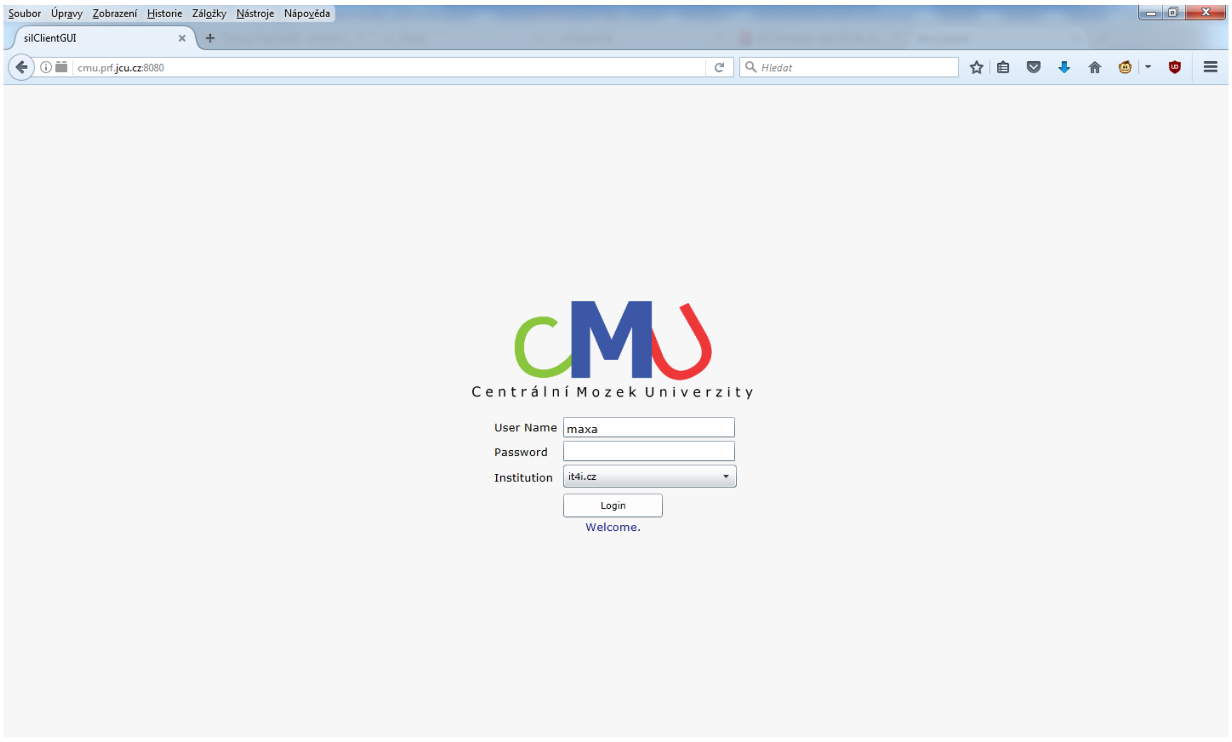
This module collects the data from the electronic watt meters connected to the virtualization hosts. The obtained data is forwarded to the virtual memory management module that provides the automatic consolidation of the virtual machines. The evaluation algorithm is analogical to the algorithm 3.3.

User Module

The user module serves as a provider between the system frontend and backend parts. It can connect all additional user modules to the CM. The web-based user graphical interface, allowing the interactive connection of the authorized users, is connected via this module, as well. The connection is provided via two parallel asynchronous TCP channels; one serves for the requests, the other one for responses. The communication between the user's browser and the module is encrypted by the SSL v3 protocol.

User Graphics Interface

The graphical user interface is the part that provides the system access to the authorized users. The solution is based on the web-based application working on the Microsoft Silverlight technology. The solution is fully interactive, i.e., every change that occurs is automatically delivered to the graphical user interface. The first step of the login process is the authentication, see Figure 3.24.


Soubor Úpravy Zobrazení Historie Záložky Nástroje Nápověda
silClientGUI
cmu.prf.jcu.cz:8080
Hledat
CMU
Centrální Mozek Univerzity
Login
Welcome.

User Name maxa
Password
Institution it4i.cz

Figure 3.24: Virtualization system CMU, a login page.

After the successful authentication, the users are allowed to manage their own virtualization entities. The limitation of the number s or types of the possible entities is presented in Table 3.3. The users can add, remove, start, stop, and restart, etc., on all their computers. The module supports the execution of the connection via the remote desktop protocol or SSH directly from the browser. All computers, which belong to the one concrete cluster, can be stopped/started simultaneously.

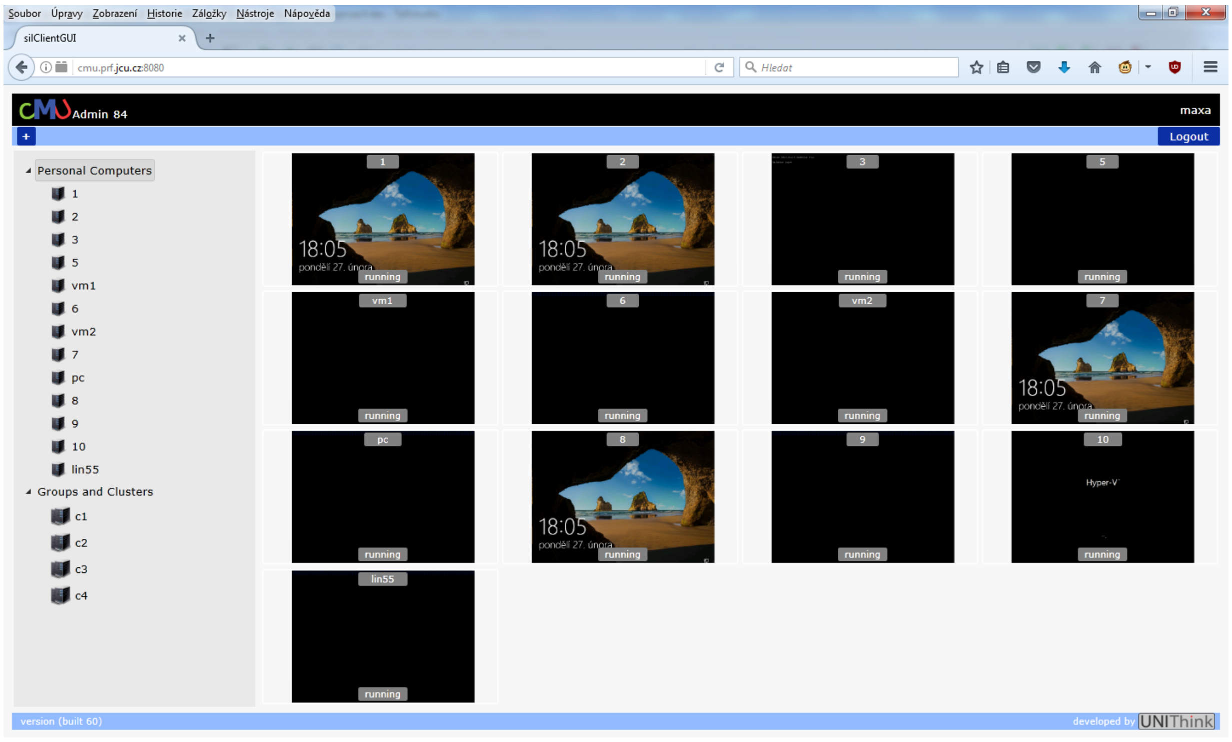


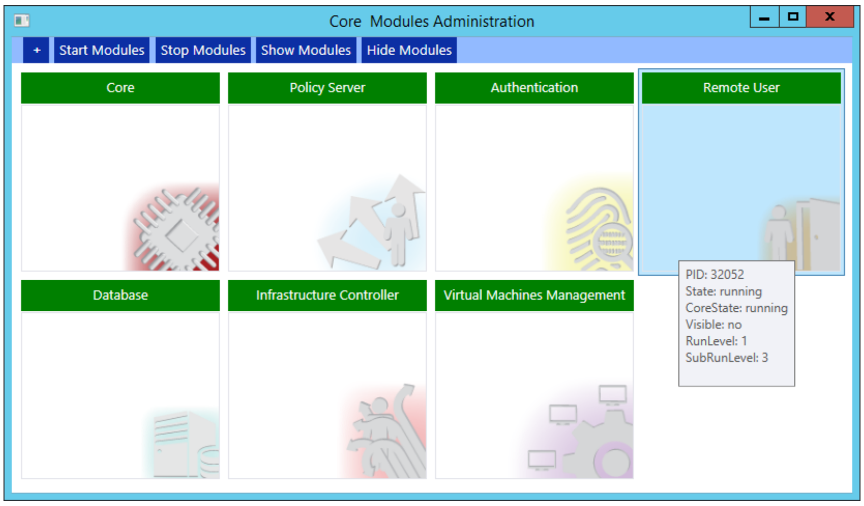
Figure 3.25: Virtualization system CMU, working environment.

System Monitoring Tool (CMU MON)

The monitoring interface is the main component that serves for the system modules management. The main functions are as follows:

1. *System Modules Monitoring:* To provide maximum reliability, it is necessary to monitor the system modules during their run time. It can sometimes happen that a module crashes or does not respond, i.e., other modules can be further affected. The CMU MON periodically monitors all modules and if a module failure is detected, then the module is automatically restarted. The module is shown in Figure 3.26.
2. *Infrastructure Monitoring:* It provides current runtime information about the infrastructure nodes, i.e., the administrator is able to see which virtualization nodes are on-line and what their current utilization is. The situation is depicted in Figure 3.27.
3. *Users and Virtual Machines Monitoring:* The administrators are able to see and manage all virtual machines according to a specific user name.

All system modules write their information both into the console output and the system log. The entire system starting/restarting can be realized by this module, as well.



PID: 32052
State: running
CoreState: running
Visible: no
RunLevel: 1
SubRunLevel: 3

Figure 3.26: Virtualization system CMU, system modules monitoring.

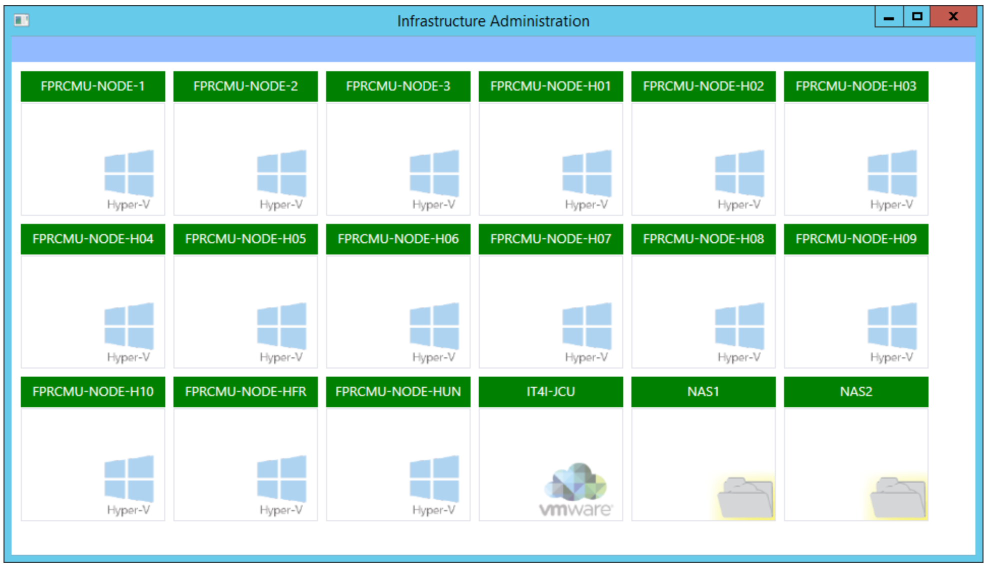


Figure 3.27: Virtualization system CMU, infrastructure nodes monitoring.

3.4.6 Alternative Operating Modes

Infrastructure Operating Mode

The above stated solution in Section 3.4.5 is implemented as a multiple client server solution, i.e., the core module serves as a module proxy. The solution is suitable for the application, where various modules run on different computers. The core-based solution is NAT traversal, where all activities can be easily monitored at the same time. The reliability of modules can be improved by multiple execution, e.g., on different computers. If one module crashes, another module can overtake its role. The architecture principle is depicted in Figure 3.28. The module with the orange dot is the one that is currently active, others serve as a backup. The solution explicitly does not solve CM crashing. Such crashing can be solved by the CMU MON.

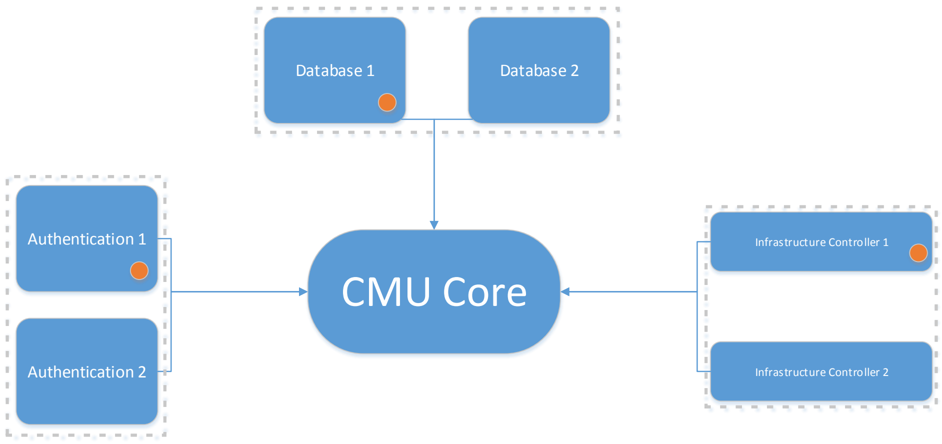


Figure 3.28: Virtualization system CMU, the infrastructure operating mode.

Adhoc Operating Mode

This mode is an alternative to the infrastructure mode. Direct communication proceeds between all modules without using the CM. This solution requires direct network transparency between all modules, which is not sometimes possible, e.g., due to NAT, firewalls, etc. In this solution, the similar modules create the module cluster, see the dashed circle in Figure 3.29). One module of the cluster is elected as a leader and communicates with other clusters (their leaders). If the leader crashes, another module of this cluster overtakes its role. This solution is more reliable than 3.4.6,but it contains more potential security problems. Action monitoring is more complicated, as well.

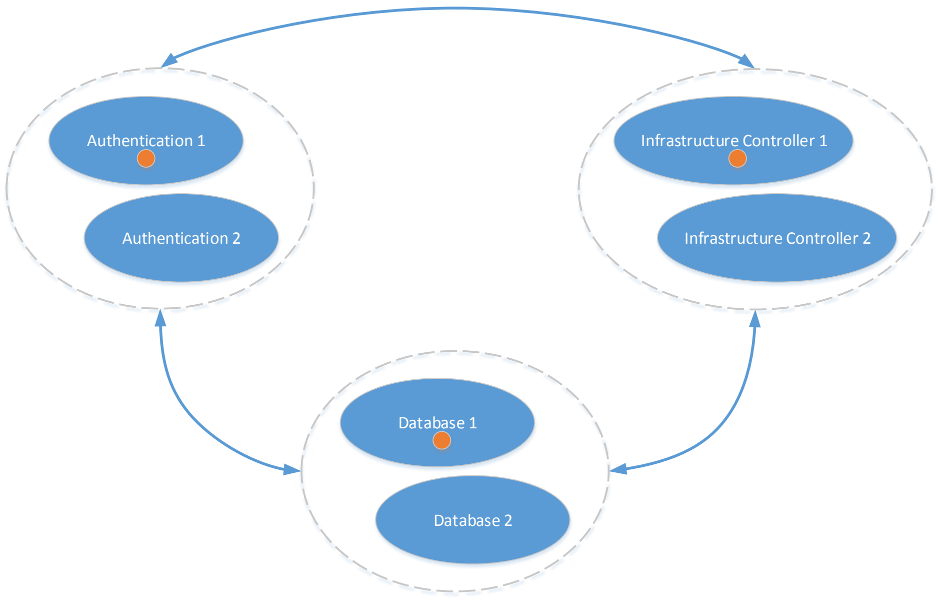


Figure 3.29: Virtualization system CMU, the adhoc operating mode.

3.4.7 Integration with the Advanced Computing Technologies

The advanced multi-core technologies that contain many CPUs or GPUs can rapidly increase the system computing performance. The virtualization environment, which shares the computing resources, does not have a general support of these technologies for all virtualization platforms. It means that the virtual machines virtualized by a specific hypervisor needn't have the access to the computing cards. In the CMU system, the Intel Xeon Phi cards were chosen, because their programming model is very similar to the common x86 based CPUs and they are possible to be shared between various virtualization nodes.

Xeon Phi Computing Cards

The detailed description and good introduction into the Intel Xeon Phi card programming can be found in [65]. The detail description of their architecture is beyond this work, so we concentrate on the way of their integration only. The detail description of their architecture is beyond this work, therefore we concentrate only on the way of their integration. The card is not a typical PCI-E 16x device managed by the operating system driver. It is internally a standalone computer with its own independent architecture. The card is powered by the PCI-E bus and additional power source connectors. It contains the internal Linux-based operating system which includes the additional drivers and system services. The driver for the master system, where such a card is installed, internally creates a new virtual network PHI interface. This network interface is then used for the communication between the operating system and the computing card via the terminal service (mostly SSH).

The way how to share the cards between the various virtualization nodes is illustrated in Figure 3.30. Every card can be easily incorporated into the network bridge by an external Ethernet adapter. The Ethernet card can be connected to the internal or external network switches or VLAN, therefore all PHI cards are then easy reachable through this switch.

For the HPC (Xeon Phi containing) virtual machine, the system CMU creates an additional virtual network device which is also bridged by the PHI and Ethernet cards.

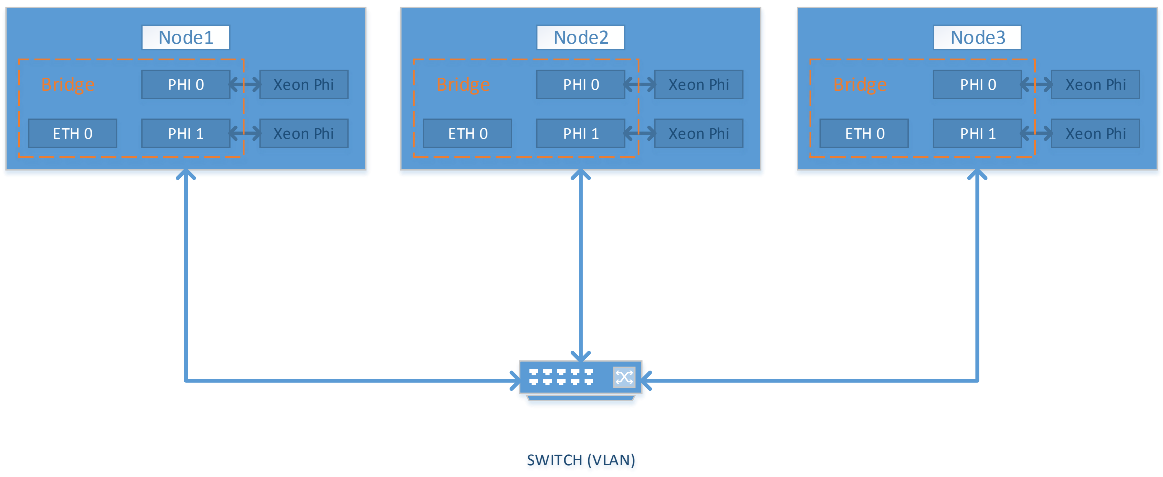


Figure 3.30: Xeon Phi cards cluster virtualization principle.

CMU in the World of Supercomputers

The Czech National Supercomputing Centre IT4Innovations (www.it4i.cz) started the project called Anselm CMU (ACMU) last year. The task of this project is to deploy this system on a part of the IT4I's virtualization infrastructure. The GUI interface module enables the access to this infrastructure for common end-users.

3.4.8 Conlusion

This section deals with the overview of the most important parts of the virtualization system CMU. The system is used at the University of South Bohemia, Faculty of Science, for teaching some computer-science oriented subjects (Distributed and Parallel Algorithms, Modern Computer Networks and Supercomputing). The solution is able to simultaneously run the tens of various virtual machines. It automatically optimizes both the utilization of infrastructure resources and the infrastructure power consumption. The system further contains the integration of the multi-core CPU computing cards that can be shared by various virtualization nodes and machines in the system.

The virtualization system is available online via the link **http://cmu.prf.jcu.cz**.

Chapter 4

Main Results

4.1 Efficiency of Advanced Virtualization Technologies

The first task of our research was to determine the virtualization technology efficiency and overhead. It was important to know if it is efficient enough to create a virtualization infrastructure for the specific purposes. The research and subsequent tests demonstrated that the virtualization efficiency was very high (over 90%). To reach such efficiency, it was necessary to consider all utilization aspects. Complete results for hardware-assisted virtualization were presented in Section 3.1.10.

4.2 New Approach to Virtual Distributed System Live Migration

The approach is described in Section 3.2. There was proposed a novel migration algorithm 3.2 that was tested during various simulations. A substantial part of this algorithm is used by the virtualization system CMU for the virtual machines migration. e.g., it is used by their consolidation.

4.3 Virtual Machine Consolidation Algorithm

The electrical power consumption is one of the factors that substantially affect the operational costs of the distributed computing platforms. The proposed algorithm, presented in Section 3.3, was able to eliminate the costs by about tens of percent. The algorithm was first verified by theoretical simulations and its further practical implementation showed the similar behaviour. Some results can be seen in Section 3.3.1.

4.4 Distributed Virtualization System CMU

The proposed algorithms from the previous Section were incorporated into the prototype of the distributed virtualization system, called CMU. The system natively supports the Microsoft the Hyper-V and VMWare ESx/Sphere virtualization platforms. The solution is used as a new generation educational system at the University of South Bohemia, the Faculty of Science. The system is used further by the Czech National Computing Centre for an access provided to the end-users. All important parts of the system were described in Section 3.1.10.

Chapter 5

Conclusions

The dissertation thesis contains complete practical and theoretical background standing behind the creation of the distributed virtualization system that is used for the educational and research activities.

The main motivation aspects were formed in Chapter 1. There was described the thesis organisation as well.

In the Chapter 2, main principles used both in current virtualization technologies and in live virtual machine migration are described and the current approaches of the virtual machine consolidation are introduced.

The Chapter 3 contains theoretical and practical testing of virtualization technology. A novel approach to the live migration of distributed virtualization systems, fully distributed solution for the virtual machine consolidation and finally the description, design and implementation of the CMU virtualization system are presented.

The important benefits of the thesis were summarized in Chapter 4.

5.1 Future Work

The author of the dissertation thesis suggests exploring the following tasks:

- It would be interesting to further improve the efficiency of the live migration by optimizing or selecting a memory compression algorithm according to the memory content. The memory size is the most important factor for virtual machine migration time. This factor substantially affects the network utilization.

- The implementation of the adhoc mode supported by the virtualization system CMU could be improved or optimized. The performance equivalent to the client-server mode including the ultra-high availability feature should be an ideal solution.
- The support of the devices that can be used for power consumption consolidation can be extended. The interesting solution could be the connection of the consolidation system with the air condition system control of the data centre on various levels.
- It seems to be interesting to use the time allocable (not permanently usable) virtualization nodes that can be available only in the specific interval of the day. This situation can easily happen, e.g., in university environment. Some educational computers can be used partly for education and the rest of the day they can serve, e.g., as virtualization hosts.

Bibliography

[1] D. Ruest, N. Ruest, *Virtualization*, pages 40-50, McGraw Hill press, 2010

[2] VMWare inc., *VMware Infrastructure Architecture Overview*, available online, 2006

[3] Q. She, *Hardware virtualization technology and its security*, lecture online, Peking University. 2010

[4] B. Golden, *Virtualization for Dummies*, pages 8-26, Wiley Publishing, Inc., 3th edition, 2011

[5] K. Gray, T. D. Nadeau, *Network Function Virtualization*, pages 64-82, Wiley Inc., 2016

[6] T.Clarc,k *Storage Virtualization: Technologies for Simplifying Data*, pages 16-30, Pearson Education, 2005

[7] L. Boursas, M. Carlson, *Systems and Virtualization Management. Standards and New Technologies*, Springer, Second International Workshop, SVM 2008 Munich, Germany, October, 21-22, 2008

[8] M. Portnoy, *Virtualization Essentials*, pages 57-62, Wiley Inc., 2012

[9] G. Blokdijk, *Data Center Virtualization - Simple Steps to Win, Insights and Opportunities for Maxing Out Success*, pages - 120-131, Emero Publishing, 2015

[10] M. Iqbal, *IT Virtualization Best Practices*, pages 170-194, MC Press, LLC., 2010

[11] S. Winter, *Concept for system virtualization in the field of high availability computing*, page 31-40, GRIN Verlag, 2009

[12] Clark, Fraser, Hand, Hansen, and Jul., *Live migration of virtual machines*, Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), pages: 273-286, 2005

[13] H. Jeong , S.H. Kim, H. Kim , Lee, Seo, *Analysis of virtual machine live-migration as a method for power-capping*, Journal of Supercomputing, 2013

[14] Oberheide, Cooke, Jahanian, *Empirical Exploitation of Live Virtual Machine Migration*, 2010

[15] C. Jo, E. Gustafsson, J. Son, B. Egger, *Efficient Live Migration of Virtual Machines Using Shared Storage*, ACM Sigplan Notices, pages: 41-50, 2013

[16] Jin, Deng, Wu, Shi, Pan, *Live Virtual Machine Migration with Adaptive Memory Compression*, In CLUSTER'09, pages 1-10, 2009

[17] *LZO real-time data compression library*, available online, http://www.oberhumer.com/ opensource/lzo/

[18] P. R. Wilson, S. F. Kaplan, and Y. Smaragdakis, "The case for compressed caching in virtual memory systems," in Proceedings of the USENIX Annual Technical Conference (USENIX'99), 1999, pp. 101–116.

[19] , Gustafsson, Optimizing Total Migration Time in Virtual Machine Live Migration, master thesis, University of Uppsala, 2013

[20] H. Jin, L. Deng, S. Wu, X. Shi, H. Chen, X. Pan, *MECOM: Live migration of virtual machines by adaptively compressing memory pages*, future generation computer systems-the international journal of grid computing and escience, pages: 23-35, 2014

[21] R. Nathuji , K. Schwan, *Virtualpower: coordinated power management in virtualized enterprise systems*, ACM SIGOPS Oper Syst Rev 41(6):265C278, 2007

[22] J. Stoess , C. Lang, F. Bellosa, "Energy management for hypervisor-based virtual machines", Proceeding ATC'07 USENIX annual technical conference on proceedings of the USENIX annual technical conference, 2007

[23] A.Verma , P. Ahuja , A. Neogi , *pMapper: power and migration cost aware application placement in virtualized systems* , Proceeding middleware '08 proceedings of the 9th ACM/IFIP/USENIX international conference on middleware. Springer, New York, pp 243–264, 2008

[24] S. Srikantaiah, A. Kansal, F. Zhao, *Energy aware consolidation for Cloud computing*, Proceedings of the 2008 conference on power aware computing and systems", San Diego, 2008

[25] X. Fan,W. D.Weber, and L. A. Barroso, *Power provisioning for a warehouse-sized computer*, in Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007, pp. 13–23., 2007

[26] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya , *A taxonomy and survey of energy efficient data centers and cloud computing systems*, Adv. Comput. 82(2), 47–111, 2011

[27] Beloglazov, A., *Energy-efficient management of virtual machines in data centers for cloud computing*, PhD thesis, Department of Computing and Information Systems, The University of Melbourne, 2013

[28] F. Farahnakian, P. Liljeberg and J. Plosila, *Energy-Efficient Virtual Machines Consolidation in Cloud Data Centers using Reinforcement Learning*, 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2014

[29] Z. Cao, S. Dong, *An energy-aware heuristic framework for virtual machine consolidation in Cloud computing*, The Journal of Supercomputing, Volume 69, Issue 1, pp 429–451, Springer, 2014

[30] C. Dupont, T. Schulze, GiulianiGet al., “An energy aware framework for virtual machine placement in cloud federated data centres“, e-Energy'12 proceedings of the 3rd international conference on future energy systems: where energy, computing and communication meet. ACM, New York, 2012

[31] J. Xu, J. Fortes, *Multi-objective virtual machine placement in virtualized data center environments*, 2010 IEEE/ACM international conference on green computing and communications and international conference on cyber, physical and social computing, Hangzhou, pp 179–188, 2010

[32] E. Feller, L. Rilling, C. Morin, *Energy-aware ant colony based workload placement in Clouds*, Proceeding GRID'11 proceedings of the 2011 IEEE/ACM 12th international conference on grid computing. IEEE Computer Society, Washington, DC, pp 26–33, 2011

[33] K. Mills, J. Filliben, C. Dabrowski , *Comparing VM-placement algorithms for on-demand Clouds*, 2011 IEEE third international conference on Cloud computing technology and science (CloudCom), Athens, pp 91–98, 2011

[34] N. Bobroff, A. Kochut, K. Beaty, *Dynamic placement of virtual machines for managing SLA violations*, 10th IFIP/IEEE international symposium on integrated network management, Munich, pp 119–128, 2007

[35] D. Borgetto, H. Casanova , A. Costa, *Energy-aware service allocation*, Future Generation Computer Systems (FGCS). Elsevier press, Amsterdam 28(5):769–C779, 2012

[36] N.Kansal, I. Chana, *Energy-aware Virtual Machine Migration for Cloud Computing - A Firefly Optimization Approach*, Journal of Grid Computing, Volume 14, Issue 2, pp 327–345, Springer, 2016

[37] B. Li, J. Li, J. Huai, T. Wo, Q. Li, L. Zhong, *EnaCloud: An Energy-saving Application Live Placement Approach for Cloud Computing Environments*, International Conference on Cloud Computing, IEEE, 2009

[38] S. Akiyama, T. Hirofuchi, R. Takano, S. Honiden, *MiyakoDori: A Memory Reusing Mechanism for Dynamic VM Consolidation*, Fifth International Conference on Cloud Computing, IEEE, 2012

[39] L. Minas and B. Ellison, *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*, Intel Press, 2009.

[40] "Standard Performance Evaluation Corporation", *SPECvirt_sc 2013*, available online, https://www.spec.org/virt_sc2013/, 2015.

[41] VMWare Inc., *VMmark*, available online, http://www.vmware.com/a/vmmark/, 2015.

[42] P. Apparao, I. Ravi, D. Newell, *Towards Modeling & Analysis of Consolidated CMP Servers*, ACM SIGARCH Computer Architecture News, Volume 36 Issue 2, pp. 38-45, 2008.

[43] D. Gupt, R. Gardner, L. Cherkasova, *XenMon: QoS Monitoring and Performance Profiling Tool*, available online, the technical report of the Hewlet Packard company, available online, http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html, 2005.

[44] A. Menon, J. Santos, Y. Turner,*Xenoprof - System wide profiler for Xen VM*, developed by the Hewlet Packard company, available online, http://xenoprof.sourceforge.net/, 2005.

[45] I.Nikolaev, R. Back, *Perfctr-Xen: A Framework for Performance Counter Virtualization*, Acm Sigplan Notices, Vol. 46, Issue 7, pp. 15-25, 2010.

[46] Xen Project, available online, http://www.xenproject.org/developers/teams/hypervisor.html, 2015.

[47] A. Iosup, R. Prodan, and D. Epema, *IaaS Cloud Benchmarking: Approaches*, Challenges, and Experience, Proc. Int'l Workshop on Hot Topics in Cloud Services, pp. 1-2, 2012.

[48] N. Yigitbasi et al., *C-Meter: A Framework for Performance Analysis of Computing Clouds*, Proc. 9th IEEE/ACM Int'l Symp. Cluster Computing and the Grid, pp. 472–477 2009.

[49] K. Ye, K. J., Wu, Z. H., Zhou, B. B., Jiang, X. H., Wang, C. Zomaya, A. Y., Virt-B: Toward Performance Benchmarking of Virtual Machine Systems, IEEE Internet Computing, Vol. 18, Issue 3, pp.64-72, 2014.

[50] Xu, X. G., Zhou, F., Wan, J., Jiang, *Quantifying Performance Properties of Virtual Machine*, Issue 2008: International Symposium on Information Science and Engineering, Vol 1, 2008.

[51] P. Xiao, Z. Hu, Z. Liu, D. Yan, X. Qu, Virtual machine power measuring technique with bounded error in cloud environments, Journal of Network and Computer Application, Vol. 36, Issue 2, pp. 18-28, 2013.

[52] C. Wen, L. Xiang, Y. Yang, F. Ni, Y. Mu, System Power Model and Virtual Machine Power, Metering for Cloud Computing Pricing, 2013 Third International Conference on Intelligent System Design and Engineering Applications (Isdea), pp. 1379-1382, 2013.

[53] Z. Shao, H. Jin, Y. Li, J. Huang, *XenMVM: Exploring Potential Performance of Virtualized Multicore Systems*, Information an International Interdisciplinary Journal, Vol. 14. Issue 7, p. 2315-2326, 2011.

[54] J.Q. Du, N. Sehrawat, W. Zwaenepoel, *Performance Profiling of Virtual Machines*, Acm Sigplan Notices, Vol. 46, Issue 7, pp. 3-14, 2011.

[55] O. Tickoo, *Modeling Virtual Machine Performance: Challenges and Approaches*, AC Sigmetrics Performance Evaluation Rev., Vol. 37, no. 3, pp. 55-60, 2010.

[56] A. Beloglazov, R. Buyya, *OpenStack Neat: A Framework for Dynamic and Energy-Efficient Consolidation of Virtual Machines in OpenStack Clouds*, Concurrency and Computation: Practice and Experience (CCPE), John Wiley & Sons, Ltd, USA, 2014

[57] W. S. Cleveland and C. Loader, *Smoothing by local regression: Principles and methods*, Statistical Theory and Computational Aspects of Smoothing, vol. 1049, pages 10–49, 1996.

[58] G. Colouris, J. Dollimore, T. Kindberg, *Distributed systems concept and design*, Fifth Edition, Addision-Wesley, 2012

[59] J. Morrison, *Architectures, Operating Systems, Parallel Processing & Programming Languages*, pages 12-27, 2004

[60] A. Kshemkalyani, M. Singhal, *Distributed Computing: Principles*, Algorithms and Systems Cambridge University Press, pages 121-233, 2011

[61] U. Deshpande, B. Schlinker, E. Adler, K. Gopalan, *Migration of Virtual Machines using Cluster-wide Deduplication*, Proceedings Of The 2013 13th IEEE/ACM International Symposium On Cluster, Cloud And Grid Computing (Ccgrid 2013), pages: 394-401, 2013

[62] K. Hwang, J. Dongarra, G. Fox *Distributed and Cloud Computing, 1st Edition, From Parallel Processing to the Internet of Things*, Morgan Kaufmann, pages: 420-454, 2011

[63] *The Spec Power Benchmark*, available online, http://www.spec.org/power_ssj2008/

[64] J. Fesl., *Exact Virtualization BenchMark (evbench)*, available online, http://https://github.com/UniThinkTeam/EVBench

[65] J. Jeffers, J. Reinders, A. Sodani, *Intel Xeon Phi Processor High Performance Programming, 2nd Edition*, Morgan Kaufmann, ISBN: 9780128091951, 662 pages, 2016

Reviewed Publications of the Author Relevant to the Thesis

[1] Fesl, J., et al. Live Migration of Virtual Distributed Computing Systems In: *International Journal of Innovative Computing Information and Control*. 2015, 11(3), 973-983. ISSN 1349-4198.

[2] Fesl, J., et al. CloudEVBench – Virtualization Technology Efficiency Testing Tool for the Distributed Infrastructures In: *International Journal of Grid and Distributed Computing*. 2016, 9(8), 249-260. ISSN 2005-4262.

[3] Fesl, J., et al. New Approach for Virtual Machines Consolidation In Heterogeneous Computing Systems In: *International Journal of Hybrid Information Technology*. 2016, 9(12), 321-332. ISSN 1738-9968.

Remaining Publications of the Author Relevant to the Thesis

[1] Fesl, J. *Virtual Distributed Computing Systems and Their Appplications.* Ph.D. Minimum Thesis, Faculty of Electrical Engeneering, Depart of Computer Science Prague, Czech Republic, 2015.

[2] Fesl, J. et al. *Virtuální paralelní infrastruktury pro velká data v metabolomice.* Konferenční sborník ENBIK 2016. ENBIK 2016 - Národní bioinformatická konference Loučeň, 2016-06-16/2016-06-15. Praha: Centrální laboratoře 2016.

[3] Fesl, J. et al. *New Processing-miningplatform for High Resolution Mass Spectral Metabolomic "Big Data".* Sborník příspěvků z 5. konference České společnosti pro hmotnostní spektrometrii. 5. konference České společnosti pro hmotnostní spektrometrii, České Budějovice 2016-04-13/2016-04-15. Česká společnost pro hmotnostní spektrometrii, 2016. pp. 28. ISBN 978-80-905045-6-1 2016.

Remaining Publications of the Author

[1] Fesl, J. et al. New Techniques of IEEE 802.11 Family Hotspots Attacks, Principles and Defense *Proceedings of the 14th European Conference on Cyber Warfare and Security.*14th European Conference on Cyber Warfare and Security, Hatfield, 2015-07-02/2015-07-03. Curtis Farm, Kidmore End, NR Reading: ACAD Conferences ltd., pp. 61-70. ISSN 2048-8602. ISBN 978-1-910810-28-6. 2015.