1 Introduction

1.2 Objective of the diploma thesis

After an introduction to the subject area of virtualization, this diploma thesis shall first provide an overview on the functionality of virtualization processes. By means of analyzing some selected Open Source methods for virtualization under Linux and their subsequent exemplary implementation, the following will provide an evaluation of these processes. By conducting benchmarks, the implementation is supposed to provide a comparison of the performance of the single systems. The objective is to show what forms of virtualization within a Linux server environment are best suited for different kinds of IT requirements.

2 Analysys of selected virtualization technologies
2.1 OpenVZ
2.2 Xen
(not translated as we intent to focus on benchmarking section of this thesis)

3 Exemplary implementation and benchmarks

Beforehand, the structure and functionality of the virtualization systems OpenVZ and Xen have been described in detail. In the following, an exemplary implementation of the respective system will be presented. For each implementation, the associated planning and concept will be described. Starting from there, the setup and configuration as well as the following launch of the respective environment will be presented. The implementation is meant to provide a performance analysis of both virtualization solutions by using benchmarks. These are supposed to deliver a final evaluation of both processes with respect to suitability, implementation effort and performance.

3.1 Basic principles of the implementation

As the implementation’s objective is to compare both virtualization processes as substantially as possible, first some basic conditions have to be set. In order to receive a utilizable statement by using benchmarks, it is mandatory to design the implementation for both procedures in a consistent way. Fedora Core 5 has been selected as an operating system base for the host as well as for the guest environment, because on the one hand it is fully supported by OpenVZ and Xen and on the other hand it is

used as the official testing and foundation platform for future implementations of the commercial [1]Red Hat Enterprise Linux.

For the installation of a test environment, the following hardware system has been made available:
• IBM xSeries 345
• 2x 2.8 Ghz Intel XEON Processor
• 2GB PC2100 DDR RAM
• Dual Ultra320 SCSI controller
• 2x 36GB HDD (RAID 1)
• Dual Gigabit Ethernet (100MBit utilized)

In order to create a test environment for an optimal utilization of resources as well as for the implementation effort required, a host system with three guests has been set up both for OpenVZ and for Xen. To be able to analyze the utilization of resources, the benchmark tests are supposed to reflect the general performance of a guest with respect to CPU performance, inter-process communication, hard disk access as well as the performance of the virtual network.

Accordingly, in addition to the program packages of the Fedora Core 5 Distribution, being installed as a standard, the guest systems both also have been provided with the GNU Compiler Collection[2] (gcc) and the Apache Webserver[3]. Chapter 3.4 will provide details on their exact reference to the benchmark tests. The host and the guests have been installed together on one hard disk within separate disk partitions, respectively. The disk space of the guests has been made available with the Logical Volume Manager (LVM) (see chapter 3.1.0.1), in order to enable the highest possible scalability of the disk space for the guest environments. Furthermore, the third extended filesystem (ext3) has been used within this reference. The ext3-filesystem does not provide the most powerful [4] file system, but it is used as the standard file system for several Linux distributions, including Fedora Core 5.

3.1.0.1 The Logical Volume Manager (LVM)

The choice of LVM for the guests partition, is, however, evident. As the amount of guests within a virtual environment often varies because of the addition of new guests and the removal of old ones, it is important to configure the available disk space for guests as dynamically as possible. Especially LVM2[5] allows for dynamically enlarging, reducing, adding and deleting disk space in the form of logical volumes during runtime. LVM provides a logical layer between the hard disk and file system. With LVM, one part of a hard disk -- or the entire hard disk as LVM drive -- is being

[1] http://www.redhat.com/rhel/
[2] http://gcc.gnu.org/
[3] http://spamassassin.apache.org/
[4] ref. http://linuxgazette.NET/102/piszcz.html and http://www.linux-magazin.de/Artikel/ausgabe/2002/01/jfs/jfs.html
[5] http://sourceware.org/lvm2/

designated as an LVM area by partitioning. This part is representing a so-called Physical Volume . One or more physical volumes can be combined as a Volume Group (see figure 3.1). It is thus possible to e.g. merge the entire disk space of several hard disks into one big partition. This merge to one partition is made possible by creating so called logical volumes within the volume group. These logical volumes now can be formatted as a normal partition with a file system, in the case of ext3 implementation, and installed under the operating system.

LVM is not only offering advantages. In particular, the error probability increases, as soon as multiple physical volumes are being deployed. It is recommended to secure LVM by using a network of Redundant Array of Inexpensive Disks (RAID), since whenever multiple physical volumes are used, data may be lost as soon as one hard disk in the network fails. Since logical volumes may potentially be distributed across two or more physical volumes, a loss of data would even be more disastrous.

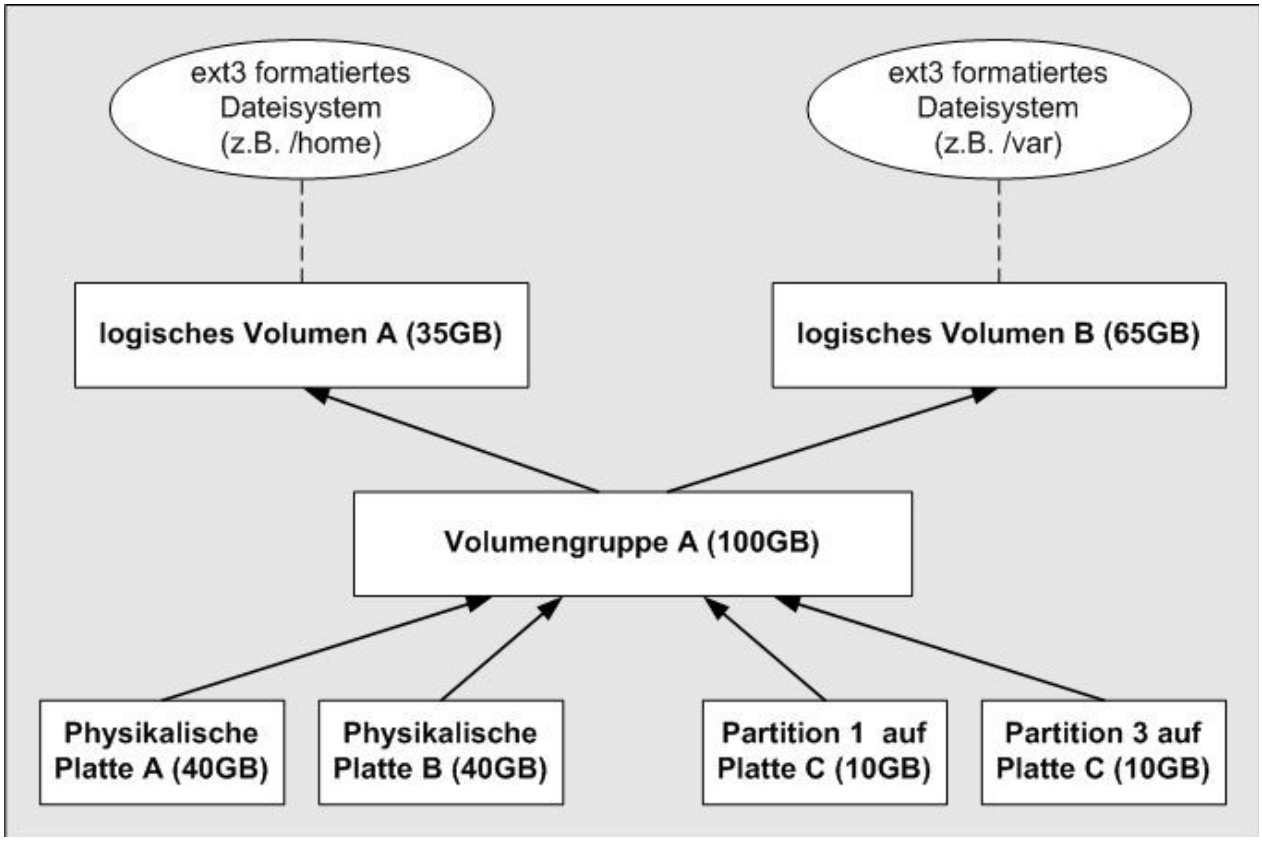

Figure 3.1: Logical Volume Management

But the advantages of LVM prevail and can be summarized as follows:
• nearly no loss of speed
• no partition constraints because of the size of single hard disks
• up to 256 logical volumes
• dynamically changing the size of the volume group by installing physical volumes

• dynamically creating/deleting partitions

The use of LVM is feasible both for OpenVZ as well as for Xen. Since within an OpenVZ environment, all guests are usually located below a directory (/vz/private/<vps-id>), it is evident to use LVM for this directory, since e.g. disk space can very easily be enlarged if necessary. LVM offers another, still bigger advantage for Xen, since for each virtual hard disk its own logical volume can be used. This significantly simplifies the maintenance effort, since new virtual hard disks can be created very quickly whenever necessary. Virtual hard disks that are no longer needed can be deleted.

3.2 The OpenVZ test environment

3.2.1 Installing the OpenVZ environment

The installation of the OpenVZ virtualization environment is divided into two steps. As the first step, the host system containing the OpenVZ kernel and the Open VZ tools has to be set up. Furthermore, the template cache enabling the set up of the Open VZ guests also needs to be built there. As the second step, the OpenVZ guests can be set up.

3.2.1.1 Setting up the OpenVZ host

After the host has been installed with a standard installation of the Fedora Core 5 operating system, all required program packages first need to be loaded and installed for the set up of Open VZ. These are available in the source code as well as in prebuild packages for distributions like Fedora Core 5. and available directly on the home page of the OpenVZ project [6]. The packages are divided into three groups. The Kernel group contains the modified Linux Open VZ kernel with the extensions for virtualization. The Utilities and Templates groups each contain utilities, scripts and configuration files for setting up and operating the Open VZ environment and the template caches for creating virtual private servers. For these three groups, the following packages have been used for setting up the test environment:

• **Kernel**
kernel-smp-2.6.16-1.2111_FC5.026test012.i686.rpm

• **Utilities**
vzctl-3.0.11-1.i386.rpm
vzctl-lib-3.0.11-1.i386.rpm
vzquota-3.0.8-1.i386.rpm

• **Templates**
vzpkg-2.7.0-18.noarch.rpm
vzyum-2.4.0-11.noarch.rpm
vzrpm44-4.4.1-22.5.i386.rpm
vzrpm44-PYTHON-4.4.1-22.5.i386.rpm

[6] http://openvz.org/download/

vztmpl-fedora-core-5-2.0-2.i386.rpm

After installing the packages with the Red Hat Package Manager (RPM) via the subsequent call, the OpenVZ environment is already essentially prepared for the installation of guests:

rpm −ivh <package_file_name>

Listing 3.1 is meant to provide a short overview on the most important OpenVZ configuration files.[7]

/etc/sysconfig/vz −global OpenVZ configuration

/etc/sysconfig/vz-scripts/ −Configuration files of the guests

/etc/sysctl.conf −Kernel parameter

Listing 3.1: The most important OpenVZ file system paths

Although file /etc/sysconfig/vz at first does not require any mandatory adjustments, file /etc/sysctl.conf[8] needs to be added in listing 3.2 and needs to be adopted with the system call sysctl -p .

```
# Allow forwarding of network packages
net.ipv4.ip_forward=1
# Deactivating the Address Resolution Protocol (ARP)
# ARP is used to assign Internet addresses to hardware addresses
net.ipv4.conf.default.proxy_arp=0
# Activating "Source route verification"
net.ipv4.conf.all.rp_filter=1
# Activating the magic−sys rq key
# is used for a direct execution of kernel commands
kernel.sys_rq=1
# Deactivating the Internet Control Message Protocol (ICMP)
# redirect for all network adapters
net.ipv4.conf.default.send_redirects=1
net.ipv4.conf.all.send_redirects=0
```

Listing 3.2: Configuration of /etc/sysctl.conf

As soon as this basic configuration is completed, the system can be re-started with the OpenVZ kernel. Finally, the template cache (see chapter 2.1.3.5) should now be updated for Fedora Core 5. A call of the command vzpkgcache creates a template cache for all installed template meta data. In this case, the metadata for the standard installation of Fedora Core 5 has been installed with the package vztmpl-fedora-core-5-2.0-2.i386.rpm. As described in chapter 2.1.3.5, current packages for the respective distribution are being loaded via the package manager yum for the template cache and are installed within a temporary virtual private server. This temporary virtual private server finally is being archived in a template file with the name fedora-core-5-i386-default.tar.gz . Templates installed in the system can be viewed by using the call vzlist .

[7] Ref. [3], OpenVZ USER’S Guide, chapter 9 - Configuring OpenVZ
[8] Ref. [3], OpenVZ USER’S Guide, chapter 3 - Installing OpenVZ Software

3.2.1.2 Setting up OpenVZ guest

Within a few minutes, a new OpenVZ VPS can be set up and installed during the following steps from the template created:

```
vzctl create <vps-id> --ostemplate fedora-core-5-i386-default --config vps.basic
vzctl set <vps-id> --hostname <name> --save
vzctl set <vps-id> --ipadd <ip> --save
vzctl set <vps-id> --userpasswd root:<password>
vzctl set <vps-id> --onboot yes --save
```

The meaning of these calls is relatively simple. A new VPS with a preset ID (here 101, 102 and 103) is generated from the template created before fedora-core-5-i386-default with the pre-defined UBC parameters from emph/etc/sysconfig/vz/vps.basic.conf. During the same step, the entire configuration for the new VPS is saved under /etc/sysconfig/vz/<vps-id>.conf.

The server receives a name at one's choice --hostname <name> and an IP address <ip>. Furthermore, a password should be defined for the VPS system user root --userpasswd root:<password>. Finally, the start of the VPS whenever the host system is being re-started should be defined. The parameter --save is used to directly adopt changes. Further optional settings for creating an OpenVZ VPS can be found in the OpenVZ User's Guide[3]. The calls mentioned have been accordingly executed for each VPS server to be created.

3.2.2 Launch

By using the following commands, the VPS guests can finally be started:

```
vzctl start <vps-id>
```

A login to the guest should be made via OpenVZ with the following call, in order to possibly adjust server settings and to configure and activate services like the Apache Web server.

```
vzctl enter <vps-id>
```

Should the installation of further program packages be necessary, it can be done by using the following command:

```
vzyum <vps-id> <yum−parameter>
```

The exact syntax for possible parameters of yum can be found on the respective Linux Manual pages by using the call man yum.

3.3 The Xen test environment

3.3.1 Installing the Xen environment

As the test system provided does not contain CPU extensions for hardware virtualization, the Xen implementation has been operated in the mode of paravirtualization only. Accordingly, the guest systems could be operated with a modified kernel only. Since the Xen tools as well as the kernel tools are part of the Fedora Core 5 distribution, they have been used directly for the test environment.

Just as for setting up the OpenVZ virtualization environment, there are two essential steps for creating a Xen environment. Initially, the host operating system needs to be installed which constitutes the domain 0 in the end. In order to implement the test environment, a standard Fedora Core 5 operating system installation has been performed here as well. One needs to observe that sufficient hard disk memory is being reserved for the guests. This has been done by allocating memory within a dedicated volume group to the non-privileged guests.

3.3.1.1 Setting up domain 0

As soon as the operating system has been installed, one can begin to reconfigure it to a domain 0. Initially, the necessary effort is relatively low, since, as mentioned before, all necessary packages for Fedora Core 5 are directly made available. The following packages are meant which can be loaded and installed via the package manager yum. Yum also considers potential dependences.

Exemplary implementation and benchmarks

- **Kernel**

kernel-xen0-2.6.17-1.2174_FC5.i686.rpm
kernel-xenU-2.6.17-1.2174_FC5.i686.rpm

- **Tools and libraries**

xen-3.0.2-3.FC5.i386.rpm
bridge-utils-1.0.6-1.2.i386.rpm
libvirt-0.1.1-1.FC5.i386
libvirt-PYTHON-0.1.1-1.FC5.i386.rpm

The packages can be installed via the following call:

```
yum install <paketname>
```

The two kernel packages provide the kernel for domain 0 and for the non privileged domains. The xenU kernel should be installed both in each

domainU and in domain 0, because it is used in domain 0 to start a guest, but also needs to provide additional kernel systems libraries required in the non privileged domain.

The Xen package primarily contains the Xen Control Daemon, the Xen Management User Interface, configuration files and further tools. Furthermore, libvirt-packages are necessary to allow for a communication interface between Xen and the domains. The bridge-utils- packages ultimately deliver the components for network virtualization. Having accomplished the installation, the Xen Management Daemon should be [9] configured first. The respective configuration file can be found under:

/etc/xen/xend-config.sxp

Listing 3.3 shows a minimal configuration with the most important parameters. It should first be decided which CPUs within the system are to be used by domain 0 (0 means all available domains) and how much central memory is to be permanently allocated to domain 0. The configuration for the network virtualization is also defined here. It should be defined which physical network interface card (<device>) is used in the host system to create a virtual switch with the name xen-br0 (see chapter 2.2.4.4). An overview on all parameters available can be found in the Linux Manual-pages via the call man xend-config.sxp and in the Xen User Manual[2].

```
## Xend configuration file
# Assigned CPUs
(dom0-cpus 0)
# Minimal memory allocation to domain 0
(dom0-min-mem 256)
# Settings for Bridge network
(network-script 'network-bridge bridge=xen-br0 netdev=<device>')
(vif-script 'vif-bridge')
```

Listing 3.3: Configuration of xend for the test environment

As soon as the configuration of xend has been completed, the host system with the domain 0 kernel can be restarted.

3.3.1.2 Setting up domain U

If the host system has been set up and Xen has been started, one can start to create Xen guests. It is necessary to first create a configuration of the guest. The configuration files for the virtual machines are being stored -- just as those for the Xen Management Daemon - under /etc/xen/ . Here, one can also find some examples. The configuration, however, is releatively simple.[10]

The following listing 3.4 shows the configuration of a guest that has been set up for the test environment:

[9] Ref. [2], Xen USER Manual, Chapter 4.1.2 - Configuring Xend
[10] Ref. [2], Xen User Manual, Chapter 5.1 - Configuration Files

```
## Configuration for virtual machine xenU_1
kernel ="/boot/vmlinuz-2.6.17-1.2174_FC5xenU"
root = "/dev/sda1 ro"
memory = 256
Name = "xenU_1"
disk = [ 'phy:VolGroup00/xenU_1.1, sda1, w', 'phy:VolGroup00/xenU_1.2, sda2, w' ]
##Network configuration
if = [ 'mac=00:00:00:00:00:01, bridge=xen-br0 ' ]
ip="192.168.0.2"
netmask="255.255.255.0"
GATEWAY="192.168.0.1"
hostname="xenU_1"
```

Listing 3.4: Configuration of xend for test environment

The parameters of the first part of this configuration are defining the general configuration of the guest xenU_1 . It is first defined which kernel to use for the guest. One needs to observe, however, that the path indication refers to a kernel file in the host system being installed there before (see chapter 3.3.1.1). Here, one also defines which virtual hard disk will contain the root file system and how much memory will be allocated to the guest. Finally, one defines the guest's name to be used for management under Xen and which disk space will be made available to the guest in the form of virtual hard disks. In this case, the two logical volumes xenU_1.1 and xenU_1.2 from the volume group VolGroup00 are made available to the guest as a SCSI device sda1 and sda2 .

The second configuration part exclusively focuses on setting up the virtual network. It is first defined how many adapters are made available to the host. Further settings always refer to one adapter respectively. One has to define which hardware address needs to be to assigned to the adapter and to which virtual switch the adapter should be connected. Furthermore, IP, subnet mask, gateway and host name may be defined here.

As soon as the configuration has been completed, one can start to install the guest. For the example, one first needs to create the LVM volume group with the two logical volumes.[11] As the next step, the two logical volumes need to be formatted with a file system. In this example,l xenU_1.1 is supposed to serve as a root file system with ext3 formatting and xenU_1.2 as a Swap-memory area for the guest:

```
mkfs.ext3 /dev/VolGroup00/xenU_1.1
mkswap /dev/VolGroup00/xenU_1.2
```

In order to set up a guest operating system on volume xenU_1.1, it has to be installed in the host system:

```
mkdir /tmp/xenU_1
mount /dev/VolGroup00/xenU_1.1 /tmp/xenU_1
```

Since the package manager yum can perform an installation in this installation directory, but for certain packages needs to gain access to devices, it is necessary to create or embed them manually:

[11] Ref. http://www.tldp.org/HOWTO/LVM-HOWTO/ - Chapter 11

```
mkdir /tmp/xenU_1.1 / dev
for i in console null zero random urandom; do
 /sbin/MAKEDEV −d /tmp/xenU_1.1/dev -x $i; done

mkdir /tmp/xenU_1.1/proc
mount −T proc none /tmp/xenU_1.1/proc
```

Furthermore, the virtual hard disks need to be made known to the guest via a so called fstab file. For the given example, this file needs to contain the following content and needs to be copied to the directory /tmp/xenU_1.1/etc :

```
/dev/ sda1 / ext3 defaults 1 1
/dev/ sda3 swap swap defaults 0 0
none /dev/pts devpts gid=5,mode=620 0 0
none /dev/shm tmpfs defaults 0 0
none / proc proc defaults 0 0
none / sys sysfsdefaults 0 0
```

The file defines the attributes of both virtual hard disks and provides further system specific parameters.

Now, one can start installing the actual operating system. With the following call, a complete Fedora Core 5 Web server environment is being installed in the directory /tmp/xenU_1.1, thus in the root file system of the guest. The second call is required to make the libraries of the non privileged kernel available within the virtual environment:

```
yum --installroot=/tmp/xenU_1.1 groupinstall -y web-server
yum --installroot=/tmp/xenU_1.1 install -y kernel-xenU-2.6.17-1.2174
```

Finally, it is possible to change to the guest system by using the chroot command, in order to determine potential settings like e.g. defining a password for the root user or defining the network settings:

```
chroot /tmp/xenU_1.1
```

The installation of the guest has been completed.

3.3.2 Launch

Finally, the launch of the guest is carried out by the call:

```
xm create -c xenU_1
```

The call starts the system and delivers a virtual console representing the starting procedure. From here, the system possesses the attributes of a complete operating system installation.

3.4 Benchmarks

Having completed the installation, it was possible to test the performance and distribution of resources of both test environments by using benchmark tests. The term benchmark in this respect designates a scale for comparing performance.

Within the context of the analysis presented in chapter 2, the following main points to be covered by a benchmark of the virtual machines have been identified:

• CPU utilization
• Inter-process communication
• Hard disk access
• Network performance

Two benchmark tests offering a comparison of these performance items have been selected for this context. A general overview on several benchmark tests available can be found under http://lbs.sourceforge.NET/.

3.4.1 Unixbench

The selection of the Unixbench 4.1.0[12] benchmark resulted from a range of tests with several benchmark programs available for Linux server environments. Unixbench is a so called benchmark suite containing a collection of test procedures in order to test multiple aspects of a server with respect to their performance. The tests to be performed are offering a comparison of the CPU performance as well as an analysis of the inter-process communication and provide an indication on the data throughput for copying, reading and writing files.

3.4.1.1 Unixbench benchmark results

In the following, the results of the Unixbench 4.1.0 benchmark test will be shown and explained. As a first note, one needs to take into account the possibility of Unixbench to use the results of one test run as a base value for the comparison with other test runs. This has been used for the tests described here to first perform a benchmark on the non-virtualized operating system environment. By inserting the results measured as a base value, these are being represented by the standardized value 10. All subsequent tests for guests in virtual environments are thus referring to the standardized value 10 determined for the non-virtualized host system. Since the execution of several test runs proved that the test results only show small variations, the results were normailized by rounding off.

[12] http://www.tux.org/pub/tux/benchmarks/System/unixbench/

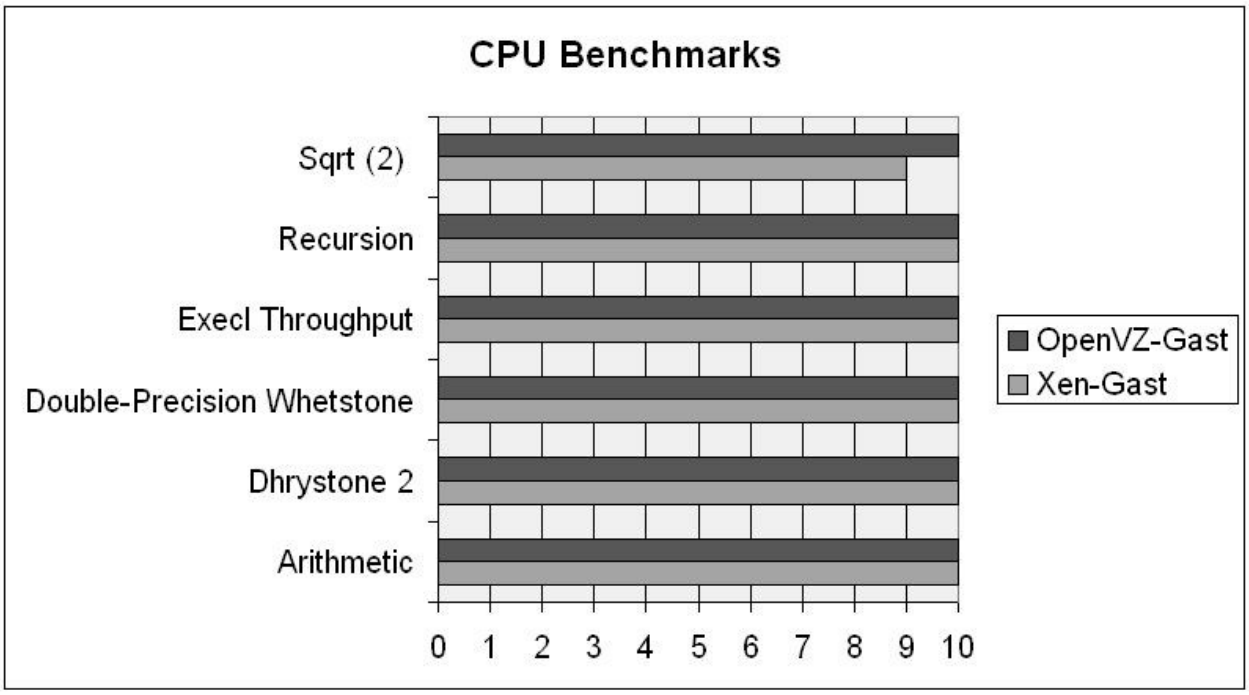

Figure 3.2 CPU Benchmarks

The results represented in figure 3.2 demonstrate that guests may be operated both under OpenVZ as well as under Xen with a nearly unreduced CPU performance. Both operating system virtualization and paravirtualization offer full CPU performance which is not reduced by the virtualization system.

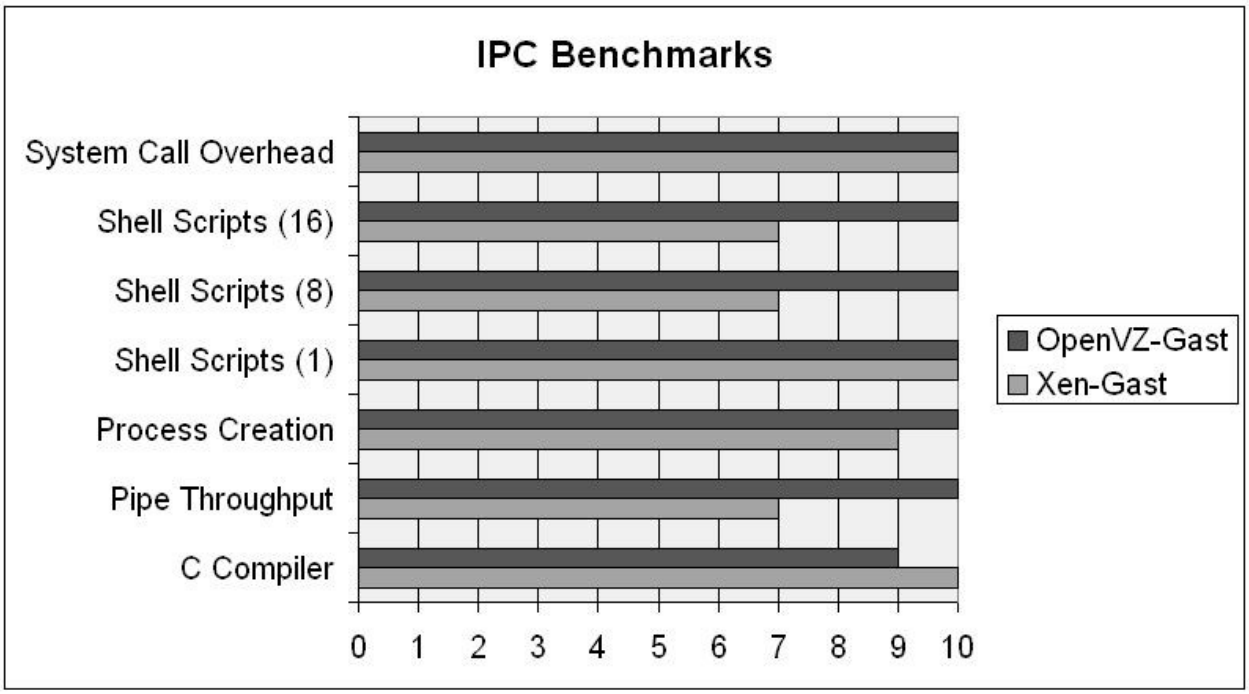

Figure 3.3 IPC Benchmarks

This, however, is not valid for the benchmark results of the inter-process communication tested, shown in figure 3.3. These results show that OpenVZ's IPC performance is significantly higher than Xen's performance in this context, since guests in OpenVZ may communicate directly with the host kernel. Xen's variations presumably result from the fact that processes in guests first communicate with the guest kernel offering a communication to other processes only afterwards via the virtualized Xen architecture. In contrast to that, OpenVZ shows nearly no performance losses.

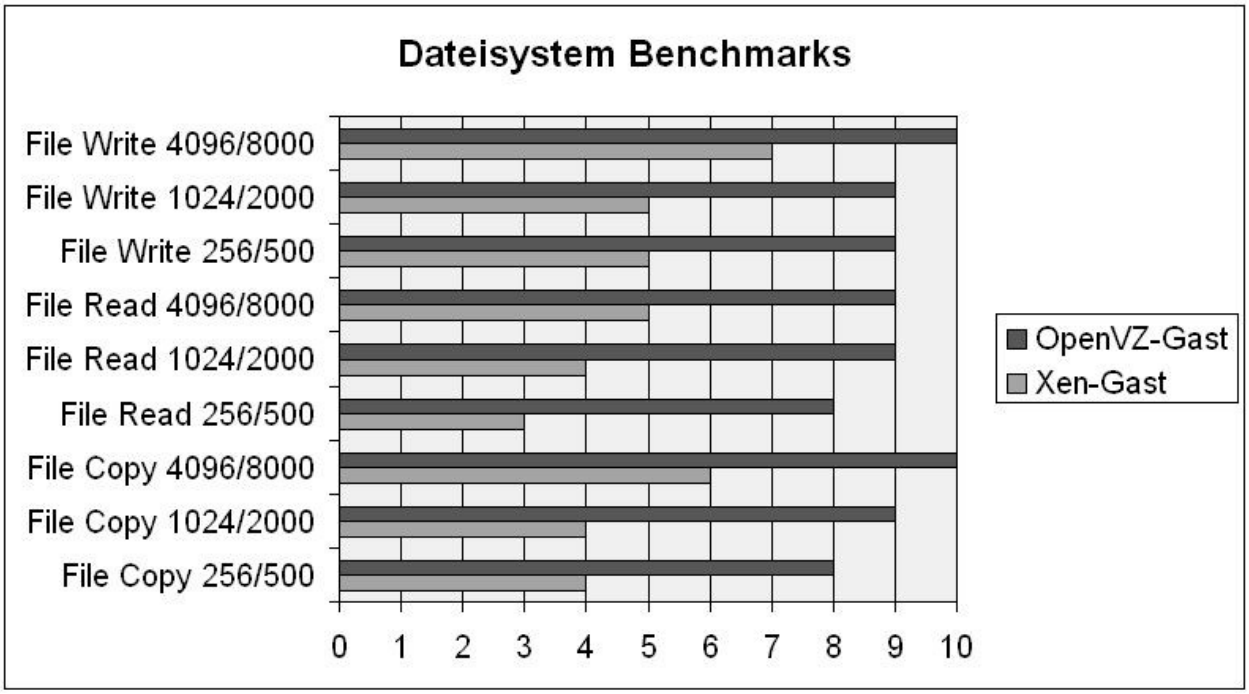

Figure 3.4 File system benchmarks

The final file system benchmark underlines these results, represented in figure 3.4. First, it becomes obvious that both Xen and OpenVZ show a performance loss in comparison with a pure host system. However, Xen's loss in some areas is much higher than OpenVZ's. While OpenVZ guests directly access the hard disk area which is directly embedded and managed by the host kernel, under Xen this area has to be accessed via the own kernel which obviously results in a higher loss of performance.

3.4.2 Siege

The Siege[13] benchmark is a tool which originally has been used for developing websites in order to measure the performance of Hypertext Transfer Protocol (HTTP) requests to a website. Siege provides an indication on the maximum speed of a web server for answering requests. It thus generates network utilization as well as workload for the web server.

3.4.2.1 Siege benchmark results

The following provides the results of the Siege benchmark test to determine the network performance of virtual machines. For this, a load test on 30 minutes has been performed respectively sending requests to an Apache Web server against a non-virtualized host and 3 virtual guests running in parallel.

[13] http://www.joedog.org/JoeDog/Siege

Table 3.1: Network benchmarks

	Host	Host Xen guests (3 parallel)	OpenVZ guests (3 parallel)
Transactions amount	1426657	1592317	1614994
Amount of transmitted data (MB)	122.45	136.77	137.07
Amount of average response time (sec)	0.02	0.02	0.05
Amount of transaction rate (trans/sec)	792.67	884.68	897.05
Amount of average transmission rate (MB/sec)	0.07	0.07	0.07

Table. 3.1 represents the results of the Siege benchmark and shows that the implementation of the virtual network structure is in no respect inferior to the physical one. It was even possible to show that the guest systems are delivering a higher transaction rate than the non-virtualized host system. This presumably results from the more efficient utilization of 3 web server processes operated in parallel.

4 Evaluation and conclusion

4.1 Evaluation

The exemplary implementation of a test environment derived from the analysis allowed for performance benchmark tests and offering an

evaluation of operating system virtualization and paravirtualization concepts. It turned out that OpenVZ and Xen both can offer significant advantages in their virtualization procedures for a Linux server environment. However, as a general conclusion it was observed that the complexity of the respective implementation and the implementation effort stands in contrast to the performance of a virtualization process deployed. OpenVZ allows for an optimal utilization of resources with nearly no loss by processes that need to manage the environment.

This, however, is in contrast to the lacking opportunity for virtual machines to use their own operating system kernels. This constrains the use of OpenVZ exclusively to a Linux server environment with a single kernel as the management instance. It is possible to dynamically assign ressources to a virtual environment, but it is not possible to exclusively assign physical resources to a guest, for example dedicate one CPU to one virtual environment or declare dedicated memory- except for network interface cards. As a huge advantage, however, one needs to mention the opportunity of a simple and quick installation of guests allowing to create a new guest within a few minutes. OpenVZ is thus suited for being deployed in environments with similar or identical structure of guests like e.g. a web server farm.

In contrast to this, Xen is providing the functionality to operate multiple operating systems on one host system. As Xen provides a complete virtual hardware architecture in the broadest sense, one can explicitly assign physical resources like a CPU or memory areas to a virtual machine. This allows for creating complex environments with a multitude of different guest systems. However, this also results in a higher administration effort for the installation of guests, as there are currently only a few tools available enabling a fully automated installation of guest systems. Another disadvantage is the fact that on the one hand, despite paravirtualization, many more management resources are being lost than in an OpenVZ implementation.

Furthermore, adjustments to the kernel of the guest systems are necessary if Xen is operated as a paravirtualization system. The deployment of Xen is thus limited to guest operating systems supporting Xen’s architecture or allowing for adjustments to the kernel. This becomes invalid for an implementation of Xen as the hypervisor of a hardware virtualization environment. However, it still remains to be seen to what extent the performance disadvantages will be compensated by this way of implementation.

4.2 Conclusion

In information technology, virtualization has gained more and more importance during the last years , since in many modern computing systems a resource pool is available which often remains unutilized. Server virtualization can help to efficiently use these resources. Because of the large range of different virtualization approaches, it is possible to

use suitable forms of virtualization for multiple IT areas. Within the frame of this diploma thesis, the different virtualization procedures first have been described in detail. An overview on available virtualization solutions has been created. As a result of these findings, the two Open Source implementations OpenVZ and Xen were selected to deliver a detailed analysis on their functionality as Linux virtualization systems.

The analysis demonstrated the exact implementation and operation mode of both processes and provided the base for a respective exemplary implementation. This implementation first delivered an overview on the implementation effort for the processes. A performance evaluation for both virtualization solutions was provided by performing benchmarks. The analysis in detail, the required implementation effort as well as the benchmark results allowed for an evaluation with respect to the suitability of OpenVZ and Xen for a deployment within a server environment.

The comparison of both processes led to the result that OpenVZ can deliver an extremely efficient virtualization approach for a deployment within a Linux operating system layer. This approach offers a level of performance similar to a standalone physical system without a perceptible loss through the implementation itself. On the other hand, Xen offers the functionality of a very comprehensive resource allocation offering the opportunity to exclusively make a large part of the virtual components available to guests, thus ensuring high scalability. In summary, one can state that the analysis and evaluation of the different implementations showed that no virtualization process is optimal for each kind of use case. It is important to first conduct an extensive survey for each application in order to select a solution which is most suitable for the respective requirements.