OULU UNIVERSITY OF
APPLIED SCIENCES

András Németh

Computing Resource Optimization Using Open Source Virtualization Technologies

Computing Resource Optimization Using Open Source Virtualization Technologies

András Németh

Master's Thesis
November, 2012
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Author:	András Németh
Title:	Computing Resource Optimization Using Open Source Virtualization Technologies
Supervisors:	Timo Räty, Dr. Kari Laitinen
Term and year of completition:	November, 2012
Number of pages:	68 + 18 appendices

Operating system virtualization techniques allow to decouple the operating system from the underlying physical hardware. This concept opens new views to software and system engineers to improve the current ways of working. Virtualization allows a more abstract and effective way of organizing computing resources. It has a great potential to reduce costs and provide more operational flexibility.

In the thesis, designing, building and configuring a low cost cluster of virtual servers is explained. Standard commodity desktop class computers and free open source software were used to build such a system. The aim of creating a virtual server cluster was to emphasize the importance of the deliberate computing resource allocation. With the help of a managed virtual cluster, a wide variety of tasks can be performed in a very flexible manner. The virtual server system is complemented by a distributed storage in which multiple disks are grouped and connected to form a fault tolerant high performance strorage repository.

The performance measurements and comparisons were made using various guest operating systems and desktop environments with regard to processor usage, memory and disk footprints. An example use case of a distributed software compiler is demonstrated as a subject of th measurements. The built system is proven to be a low cost yet well performing virtual environment.

Keywords:
virtualization, virtual machine, distributed computing, distributed storage, GlusterFS, performance comparison, XEN, XCP

CONTENTS

Abbreviations and acronyms

AMD	Advanced Micro Devices	13
AoE	ATA over Ethernet	25
APIC	Advanced Programmable Interrupt Controller	44
BIOS	Basic Input/Output System	13
CD	Compact Disc	39
CD/DVD-ROM	Compact Disc or Digital Versatile Disc Read-only Memory	44
CLR	Common Language Runtime	16
CP	Control Program	13
CPU	Central Processing Unit	13
CRUSH	Controlled Replication Under Scalable Hashing	24
DR	Disaster Recovery	46
EC2	Elastic Compute Cloud	29
ext4	Fourth extended filesystem	45
FLOSS	Free/Libre and Open Source Software	14
GB	Gigabyte = 10^9 bytes	22
GbE	Gigabit Ethernet	35
GPU	Graphics Processing Unit	30
GUI	Graphical User Interface	38
HBA	Host Bus Adapter	35
HDD	Hard Disk Drive	34
HDFS	Hadoop Distributed File System	23
HVM	Hardware Virtual Machine	18
HW	Hardware	13
IBM	International Business Machines	13
iSCSI	Internet Small Computer System Interface	25

ISO	International Organization for Standardization	44
IP	Internet Protocol	25
IT	Information Technology	38
I/O	Input and Output	18
IOPS	Input/Output Operations Per Second	34
JVM	Java Virtual Machine	16
KVM	Keyboard Video and Mouse	37
LAN	Local Area Network	26
LMDE	Linux Mint Debian Edition	31
LTS	Long Term Support	31
LV	Logical Volume	28
LVM	Logical Volume Manager	40
MiB/s	Mibibytes per second = 2^{20} bytes per second	34
MiB	Mibibyte = 2^{20} bytes	31
MDS	Management Data Server	24
NFS	Network File System	61
NIC	Network Interface Card	34
NTP	Network Time Protocol	44
OS	Operating System	16
OSS	Object Storage Server	24
OVD	Open Virtual Desktop	17
P2V	Physical to virtual	14
PBD	Physical Block Device	28
PC	Personal Computer	34
PVHVM	Paravirtual on Hardware Virtual Machine	19
RAID	Redundant Array of Independent Disks	25
RDP	Remote Desktop Protocol	38
Rx	Received	56

DDR-3 SDRAM	Double data rate type three synchronous dynamic random access memory	35
SAN	Storage Area Network	24
SATA	Serial Advanced Technology Attachment	25
SPOF	Single Point of Failure	31
SR	Storage Repository	28
SSD	Solid-state Drive	35
SW	Software	14
TCP	Transmission Control Protocol	41
TPS	Transactions Per Second	50
Tx	Transmit	56
UDP	User Datagram Protocol	41
UUID	Universally Unique Identifier	47
V2P	Virtual to physical	14
VBD	Virtual Block Device	28
VDI	Virtual Disk Image	28
VHD	Virtual Hard Disk	40
VIF	Virtual (network) Interface	45
VM	Virtual Machine	13
VMC	Virtual Machine Cluster	31
VMM	Virtual Machine Manager	18
VNC	Virtual Network Computing	38
XAPI	Xen Application Programming Interface	38
XCP	Xen Cloud Platform	17
XFCE	Xforms Cool Environment	31
XVA	Xen Virtual Appliance	17
XVP	Xen VNC Proxy	39

1 Introduction

In a turbulent economy, virtualization and cloud computing are becoming more and more attractive for enterprises because of the convenience and flexibility over traditional computing. Using a virtual machine is convenient compared to traditional computing, for example, when the user has to have a machine with a certain set of special applications. Today it is typical to distribute applications preinstalled on a virtual machine which the user can deploy without complicated installations and configurations. As an example, Kdenlive free and open-source video editor sofware can be downloaded in a complete virtual machine image format (26). A virtual machine is flexible because it can be easily equipped with a varying set of resources such as processing power, memory or storage. Virtualboximages[1] website provides thousands of preinstalled open source operating systems with given set of applications for download. For a quick evaluation it is more convenient to use these images instead of installations. If one looks around and seeks for virtualization trends in publications and in online media, one can find numerous reports written in this topic.

Zenoss[2] conducted a virtualization and cloud computing survey in 2010 to measure the usage trends. The number one stated goal with regards to virtual infrastructure was cost savings (64.7%) followed by deployment control (1). 43.3% of participants out of 204 individuals indicated flexibility as the main reason for using virtualization (see figure 1.1). It is undoubtedly visible from the data that the demand is highest for operating system virtualization and application level virtualization; however, storage virtualization plans are also significant. Knowing these trends results in a question. What can we learn and benefit from various virtualization models?

This thesis investigates how virtualization can be introduced starting off small-scale without massive investments. The focus is put on technical and practical aspects and on the sharing of experiences. The aim is to gather the knowledge for building and configuring a cluster of virtual servers. After a short overview of the virtualization history and the summary of the recent virtualization techniques, it is shown how a working prototype of a

[1]http://virtualboximages.com
[2]http://www.zenoss.com

serverless (without managing server) virtual cluster was designed and constructed using commodity hardware.

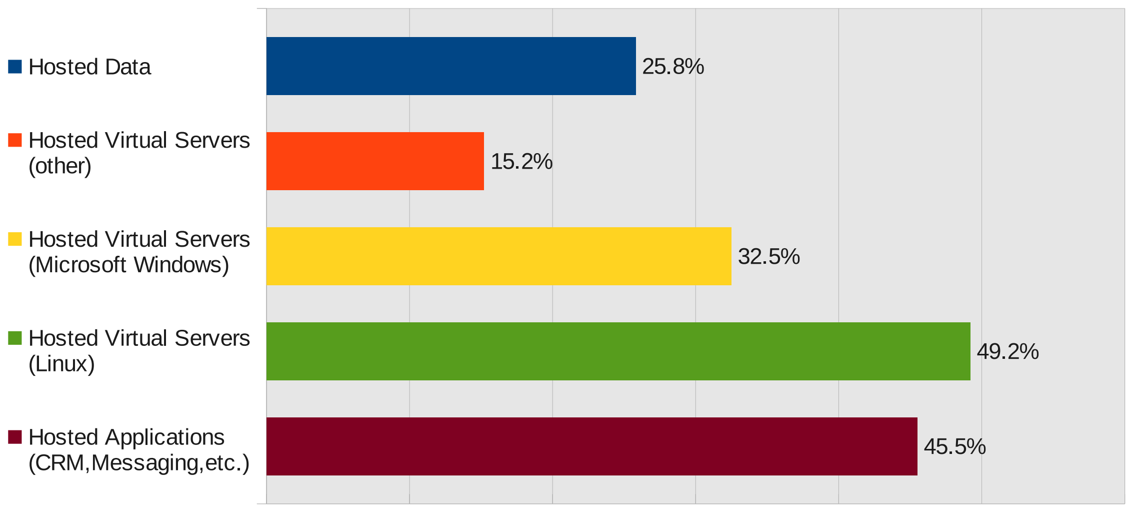


FIGURE 1.1: Survey results about the planned virtual deployments for the near future. The survey is dated second quarter of 2010. (1)

1.1 Virtualization: definition

Virtualization has an encompassing scope in manner. Numerous definitions can be found from various sources. The one that the author has selected is written by Amit Singh in 2004:

> Virtualization is a framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others. (24)

Virtualization does not always imply a division or partitioning of the resources but the opposite. Distributed computing grids appear as one logical entity that can also be interpreted as virtual.

1.2 History of virtualization

Virtualization was first implemented in the 60's by International Business Machines (IBM) corporation. CP-40, the first time-sharing virtual machine/virtual memory operating system provided a Virtual Machine (VM) environment in which multiple instances (up to 14) of client operating systems were running (36). The system consisted of a virtualizing Control Program (CP) which created multiple independent VMs. Each virtual machine had its own set of virtual devices, mapped to the real hardware of the system. CP helped in segregation of complex system problems from a single user application. Isolating users from each other improved system stability. A bug in one user's software could neither crash another user's application nor the underlying CP. The platform was made generally available to IBM customers in source code format later in 1968 (36).

In the 70's and 80's improved versions of CP-40 virtualization platforms were introduced in numerous large IBM mainframes. Since mainframes were expensive resources at the time, they were designed for partitioning as a way to fully leverage the investment (22).

In the 90's inexpensive x86 server and desktop deployments led to new infrastructure changes and different kinds of challenges. More and more Linux and Windows server operating systems became available running on x86 processor architecture. In 1999, VMware introduced the first x86 virtualization product, VMware Virtual Platform. Unlike mainframes, x86 machines were not designed to support full virtualization. It was achieved by complex software techniques.

Since 2006 Intel and Advanced Micro Devices (AMD) processors have hardware virtualization capability. These hardware features differ between Intel and AMD processors. Intel named its technology *VT-x*; AMD calls theirs *AMD-V*. The hardware virtualization features first need to be enabled in the Basic Input/Output System (BIOS) before VM can use them on many systems. The user needs to look-up the corresponding Central Processing Unit (CPU) flag to determine whether the CPU supports Hardware (HW) virtualization. The name of the flag for *AMD-V* it is "svm" and for *VT-x* is "vmx". This can be displayed on Linux operating systems via */proc/cpuinfo* file. An example for checking the CPU flags can be found in appendix 1.

1.3 Free/Libre Open Source

As the author is Free/Libre and Open Source Software (FLOSS) advocate, the project uses only software solutions from this model. The author tries to highlight that FLOSS is not intrinsically higher or lower quality than the proprietary software. It is not inherently more or less secure than its closed source Software (SW) counterpart (19). The difference resides in the license under which it is made available and in the development scheme whether the end user is able to contribute to the product. The FLOSS model allows for any user to view and modify the source code of a product. It is often necessary to read the source to fully understand their working methods. Using FLOSS and open standards tends to improve interoperability (19) which is considered as a major advantage in this project. Further advantages cited by proponents are expressed in terms of trust, acceptance, teamwork, collaboration, and quality (18). Using an open source platform also means that the user is never locked to a proprietary vendor and can therefore stay more flexible.

1.4 Scope and limitations

The nature of the virtualization would allow a long discussion. Therefore, the scope of this writing has to be clearly defined. After studying the available and latest technologies, the author selected a likely working prototype configuration, and made a proposal of implementation for a project. All the planning and implementation phases are explained in the following chapters, together with the faced problems and their solutions (if found). The intention is to provide a practical reference for the reader. The list below shows some adjacent fields which are **not** discussed in this thesis work:

- Proprietary tools and services from providers such as Microsoft and VMware
- Physical to virtual (P2V) and Virtual to physical (V2P) transformations
- High performance and large production grids
- Cloud computing stacks and platforms such as Cloudstack, Openstack and OpenNebula

The prototype virtual cluster is close to minimum entry level configuration and suitable for small projects only. Building high availibility and large clusters require more extensive studying and careful planning for specific needs. Building up cloud services on the top of the designed architecture is feasible. This is also mentioned as one of the future improvements.

2 What can be virtualized?

When selecting a suitable method of implementing virtualization, it is essential to have a clear understanding of different (currently available) virtualization solutions. The following virtualization methods are studied and described in this chapter:

- Application level virtualization
- Operating system virtualization
  - Guest operating system virtualization
  - Hypervisor virtualization
  - Shared kernel virtualization
- Storage virtualization

2.1 Application level virtualization

In the application level virtualization VM runs as a single process inside host Operating System (OS). The purpose of the application virtual machine is to provide a platform-independent software environment that allows a program to execute in the same way on many platforms (40). This type of VM has become popular with the Java programming language, which is implemented using the Java Virtual Machine (JVM). In .NET framework a similar process VM is called Common Language Runtime (CLR). Process VMs are always implemented using an interpreter. Application level virtual architecure allows creating platform independent and highly portable applications. The applications which are using these kinds of VMs naturally have a lower performance than their compiled counterparts.

2.2 Presentation virtualization

Presentation virtualization is an application level virtualization model that delivers users desktops and applications from a shared server, also known as server based computing or virtual desktop (31). Presentation-layer virtualization makes it possible to run applications on one location while these applications can be controlled remotely from a distant client. With presentation virtualization, applications are installed and run on centralized servers in the datacenter with screen images being delivered to the machines of users. The first implementation of presentation virtualization was the X Window System, a graphical remote display standard that was introduced in the mid-1980s.

Ulteo[1] Open Virtual Desktop (OVD) 3 is a well-known open source implementation of presentation virtualization. Ulteo provides an application delivery method that delivers users desktops and applications from a shared server. A fully funtional demo version of Xen Virtual Appliance (XVA) image can be downloaded from their internet page. It includes the session manager and application server with example applications. It is possible to start an OVD session without installing anything on the client machine, only a web browser is required with Java 1.6 support (38). Native Ulteo client application can also be used to access session. The XVA image was downloaded and imported into Xen Cloud Platform (XCP) for trial purposes. A few example screenshots can be found in appendix 4.

2.3 Guest operating system virtualization

The guest operating system virtualization (see figure 2.1) is also called software-based virtualization. This is the most simple and the easiest concept of all OS level virtualization methods. It does not require CPU virtualization support. The physical host computer runs a standard unmodified operating system such as Windows, Linux, Unix or MacOS X. Guest operating systems are created and ran within a virtualization application. The virtualization application is responsible for starting, stopping and managing each virtual machine and controlling physical hardware access. The virtualization application reads the executing guest OS CPU operation calls and replaces each privileged instruction with

[1] http://www.ulteo.com

safe emulations. The most commonly known open source virtualization application is *Virtualbox*.

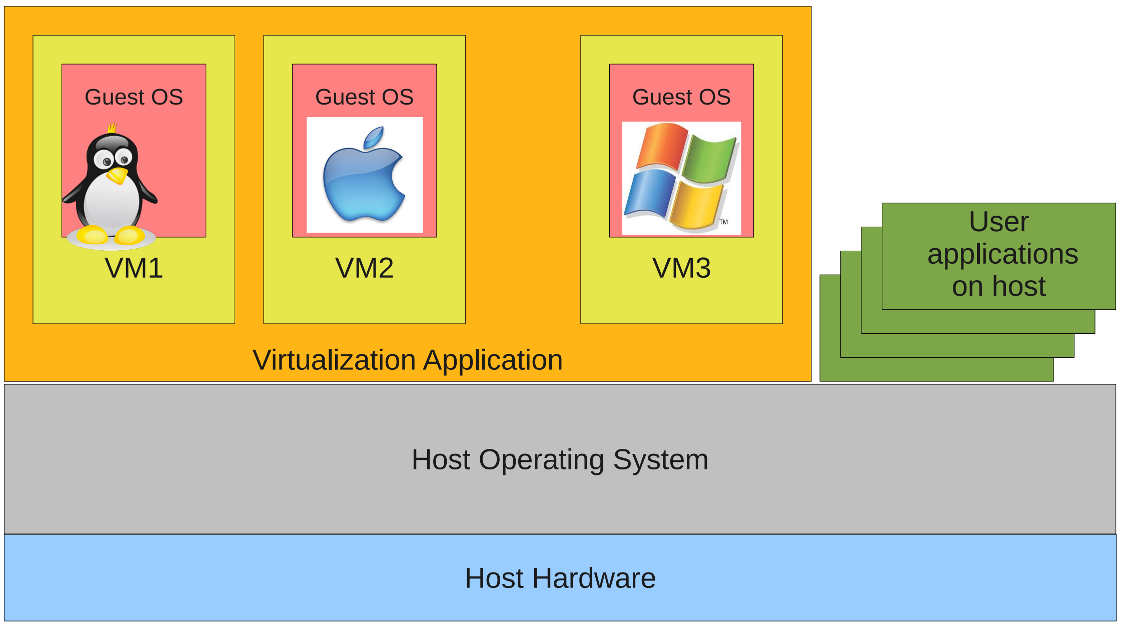


FIGURE 2.1: Guest operating system virtualization

2.4 Hypervisor virtualization

Hypervisor (also called as type 1 Virtual Machine Manager (VMM)) is a software layer that runs directly on host computer replacing the operating system (see figure 2.2). In this case, the VMM itself is the minimalistic OS. The hypervisor is the interface for all hardware request such as CPU, Input and Output (I/O), and disk for the guest operating systems (33). It is so named because it is conceptually one level higher than a supervisory program or operating system.

Hardware Virtual Machine (HVM) term is used do describe the guest operating system that is running in hardware-assisted virtualization environment. This technique requires CPU virtualization extensions e.g. Intel VT or AMD-V. HVM guests do not require special kernel, for example native windows operating systems can be used as HVM guests (45).

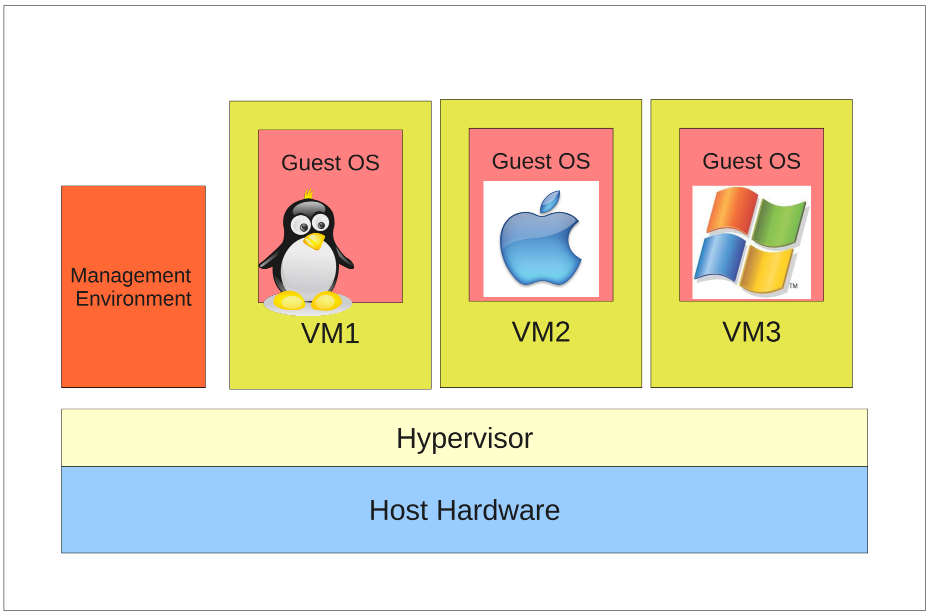


FIGURE 2.2: Hypervisor virtualization

Paravirtualization is another technique that relies on the hypervisor virtualization but does not require virtualization support from the host CPU. The guest OS is aware of the hypervisor therefore only modified guest operating systems can be loaded. Linux kernel version 2.6.24 and above have the Xen PV guest support and include all the necessary patches for use as PV guests (45). Microsoft Windows requires a HVM Guest and can not be used in paravirtualized environment. Paravirtualized guests are slightly faster than fully virtualized guests, but HVM guests can use special paravirtual device drivers to bypass the emulation for disk and network I/O. These Paravirtual on Hardware Virtual Machine (PVHVM) drivers provide better performing disk- and network I/O operations.

Hypervisor solutions are available from different vendors and sources such as: XenServer, VMware ESX/ESXi, and Microsoft Hyper-V hypervisor. The author has selected Xen open source hypervisor with hardware virtual machine solution for the prototype implementation. The following chapters explain Xen hypervisor in more detail with examples.

2.5 Shared kernel virtualization

Shared kernel virtualization is also known as system level or operating system virtualization that is available on Linux and Unix based operating systems (see figure 2.3). This type of virtualization is made possible by the ability of the kernel to dynamically change the current root file system. This concept requires the guest operating system to be compatible with the shared kernel version. For example, a 32bit guest operating system architecture will not be accessible by using a 64bit shared kernel version. With the help of the *chroot* command it is possible to change root file system from a host OS using a shared kernel. On many systems, only the super-user (a user with root privileges) can do this. *Chroot* can also be used to fix problems when the OS does not boot correctly because of problems in rootfs or in boot loader. Major web hosting providers have been using the shared kernel virtualization for years so that customers get their own virtual server for their web hosting needs. The customers do not know that the system is virtual, nor can they contact the host system through their VM (15). Unlike the above mentioned virtualization methods, the VMs only have their own root file system but not a kernel of their own.

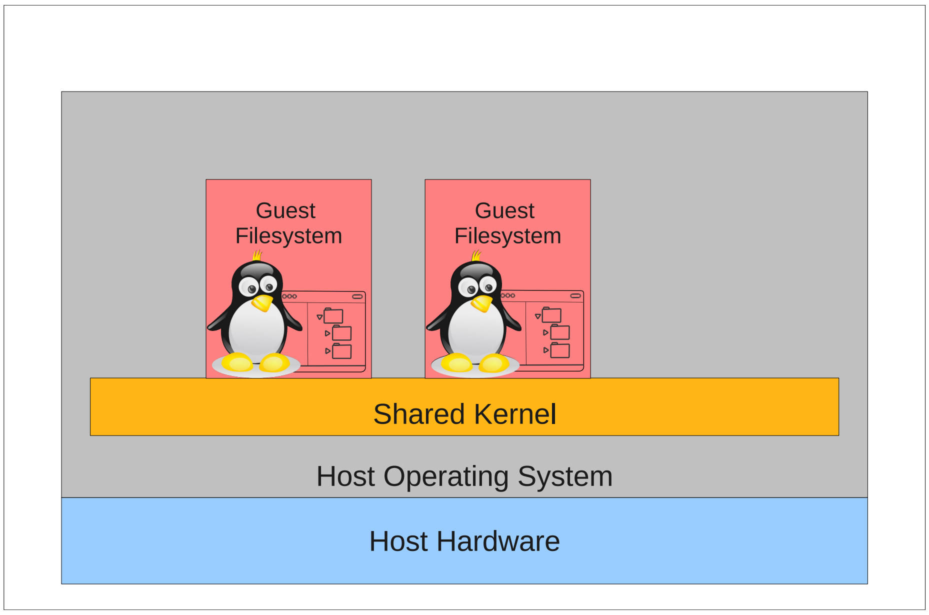


FIGURE 2.3: Shared kernel virtualization

2.6 Storage virtualization

Storage virtualization is a concept in which storage systems use special tools to enable a better functionality and more advanced features within a storage system (34). The main feature of storage virtualization is the abstraction of the logical and physical location of the data (see figure 2.4). One of the major benefits is the non-disruptive data migration when the data can be freely moved or replicated without affecting the operation of any client. Concurrently performed disk operations can significantly improve the I/O performance while the utilization of physical resources remain load-balanced. Different solutions are available based on the needs for availibility, I/O performance, search and indexing and for a combination of these.

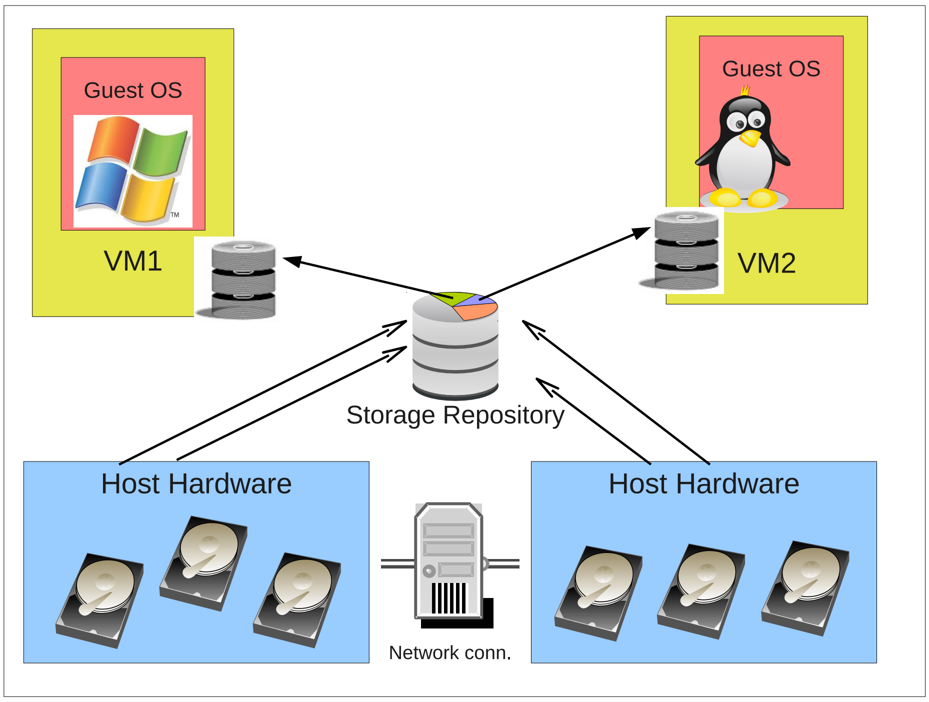


FIGURE 2.4: Storage virtualization

A simple version of the storage virtualization is considered in the design of the virtual machine cluster. Some of the most common practically used distributed disk arrays are explained in the following chapter.

2.7 Thin provisioning

Thin provisioning (over-allocation) is a mechanism that is widely utilized in virtual environments. It gives the appearance of a more physical resource than what is actually available. Most often it is associated and used with relation to the disk resources but it can also refer to an allocation scheme for any type of resource (CPU, memory). Thin provisioning is more efficient compared to the conventional allocation in cases where the amount of resource used is much smaller than the allocated amount (35). Figure 2.5 demonstrates an example of thin provisioning.

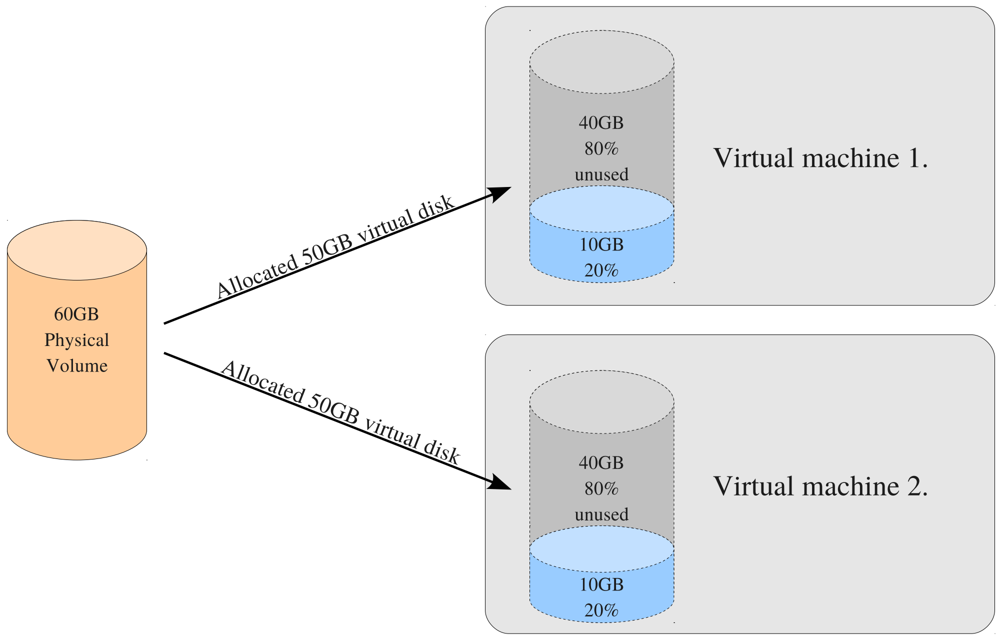


FIGURE 2.5: Thin provisioning example when a 60GB physical volume is over-allocated to 2x50GB virtual disks

3 Distributed disk arrays and file systems

As a response to the demand for data intensive file systems (systems which use large volumes of data, typically terabytes or petabytes in size), several distributed file systems have been developed in recent years. It is envisioned a further large scale increase in the use of parallel programming tools, scientific applications, data mining, etc. (52). The most commonly used open source implementations are:

- Hadoop distributed file system
- Lustre distributed file system
- Ceph distributed file system
- GlusterFS
- iSCSI
- ATA over Ethernet

These are described in more detail in the following sections.

3.1 Hadoop distributed file system

The Hadoop Distributed File System (HDFS) is a highly fault-tolerant distributed file system. It is designed to be deployed on low-cost hardware (20). Hadoop is an open source Apache^{TM} project. It is used by a wide variety of companies and organizations including Amazon, Google and Yahoo (2). Configurations can vary from standalone mode to extremely large clusters with thousands of nodes. A HDFS cluster consists of two node types. NameNode is for managing the file system metadata and multiple DataNodes storing the actual data. Hadoop has a software framework (MapReduce) for writing applications which can process vast amounts of data (multi-terabyte data-sets) in parallel on clusters of thousands of nodes. Hadoop is not a drop-in replacement for storing database files or Storage Area Network (SAN) filesystem because of its significantly higher response time.

3.2 Lustre distributed file system

Similarly to Hadoop, Lustre also has two server node types, namely Management Data Server (MDS) and Object Storage Server (OSS). On the client side (after having the necessary kernel modules loaded) it is possible to mount the Lustre cluster. The system allows multiple clients to access the same files concurrently, and all the clients see consistent data at all times (39). Expanding the array is possible on the fly (without interrupting any operations). Several Linux distributions include Lustre file system libraries and utilities. Many of the users of Lustre do not recommend it to others because it easily breaks down in many kinds of situations (9).

3.3 Ceph distributed file system

Ceph is a distributed network storage and file system with distributed metadata management (11). In the file system there is a cluster of metadata servers. This cluster manages the namespace (file names and directories) and coordinates security, consistency, and coherence (55). The minimal system does not require metadata server and has at least two MDSs for data replication. The allocation list is predictably striped across storage devices using a distribution function called Controlled Replication Under Scalable Hashing (CRUSH). This method eliminates the need to maintain the object lists and look-up tables. The Ceph client is merged to Linux kernel since version 2.6.34 (14).

3.4 GlusterFS

GlusterFS is an open source, highly scalable clustered file system. It can be flexibly combined with commodity physical, virtual, and cloud resources to deliver highly available and performant storage with relatively low cost. GlusterFS is suitable for public and private cloud environments (14). Unlike other cluster file systems, GlusterFS does not use a centralized meta-data server. This feature makes it simple and easy to deploy in prototype environments. The author has selected this filesystem to be used in the current project.

The following two sections describe software solutions that can be used in connection with the above mentioned distributed file systems to access data:

3.5 iSCSI

There are several existing software solutions that were designed to help building a distributed disk array between computers using Ethernet interface. Internet Small Computer System Interface (iSCSI) is a network storage protocol over the Internet Protocol (IP). The protocol allows to use SCSI commands over an IP network. This was earlier possible only by using costly Fibre Channel (high-speed optical network technology developed for storage networking) technology. The clients (called initiators) can send SCSI commands to block storage devices (targets) on remote servers (32). iSCSI uses a client-server architecture and usually only one client has access to the remote block storage. A server that makes targets available only needs free partition to be available for export, and the partition does not need to be a SCSI disk. The technique may be combined with clustering solutions by exporting logical volume(s) and/or Redundant Array of Independent Disks (RAID). Open-iSCSI is an open source project that provides multi-platform implementation of iSCSI (28). The binaries are available for all major Linux distributions.

3.6 ATA over Ethernet

ATA over Ethernet (AoE) is a network protocol that was designed and optimized for high-performance access of Serial Advanced Technology Attachment (SATA) storage devices over Ethernet (5). AoE is a layer 2 protocol which makes it fast and lightweight. The protocol is non-routable, so the distributed disk array is not extendable over different sub-networks. Coraid provides a hardware AoE cluster implementation called EtherDrive^{TM}. Coraid claims that “AoE delivers a simple, high performance, low cost alternative to iSCSI and FibreChannel for networked block storage by eliminating the processing overhead of TCP/IP ” (54).

Ivan Pepelnjak (7) strongly criticises the design and simplicity of this protocol due to the lack of authentication, re-transmission and fragmentation. He disagrees with the existence of the inherent security in the non-routability of AoE.

On Linux it is relatively simple to implement AoE using aoetools (8) and vblade software packages. Vblade is a program that makes a seekable file available over an ethernet local area network (LAN) via the AoE protocol. The file is typically a block device or partition like */dev/md0* or */dev/md0p1*. When vblade exports the block storage over AoE it becomes a storage target. Another host on the same Local Area Network (LAN) can access the storage if it has a compatible AoE kernel driver. Similarly to iSCSI, only one client has access to the remote block storage. Figure 3.1 shows the differences between the iSCSI and AoE protocol stacks.

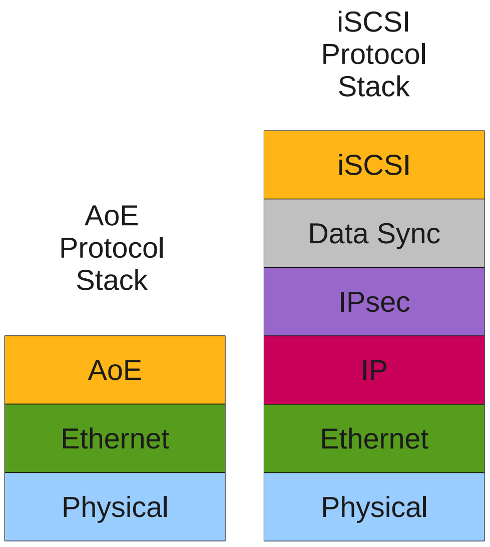


FIGURE 3.1: Comparison of the protocol stack complexity between AoE and iSCSI (6)

4 Virtual machine cluster

A cluster is a group of computers (virtual servers) bound together into a common resource pool. All the VMs in this pool can have access to the Gluster distributed volumes. The usage of distributed storage could provide high scalability with redundancy options. In this chapter the design of such a cluster is presented. Figure 4.1 explains the concept of the designed prototype.

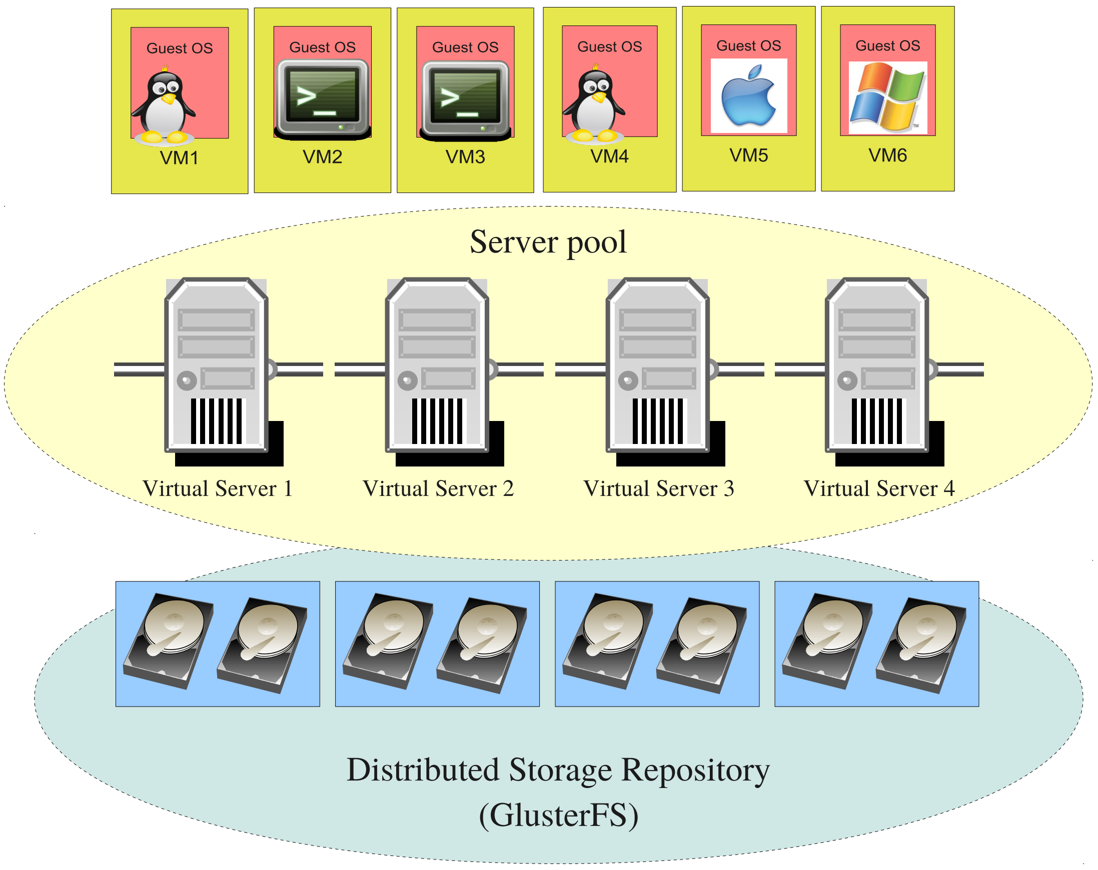


FIGURE 4.1: Virtual machine cluster prototype

4.1 Storage subsystem

The storage objects defined in XCP in a container called a Storage Repository (SR) to describe a particular storage target, in which Virtual Disk Images (VDIs) are stored. SRs can be local or shared. For instance, a shared SR is defined in resource pools. Physical Block Devices (PBDs) are the interfaces (software objects) through which the XCP server accesses SRs. In spite of its name (physical), in this case it is a Logical Volume (LV) device but can also be another (any) type of block device. A VDI is a disk abstraction which contains the contents of a virtual disk. This virtual disk is presented to a VM through Virtual Block Devices (VBDs) which are similar interfaces to previously mentioned PBDs.

SRs can be attached (plugged) to multiple PBDs which might be located on one or several physical hosts. Also, VBDs can be attached to one or many VMs and one VM can have multiple VBDs attached (see figure 4.2).

As a simple example of typical PBD at the server can be a partition of pysical disk e.g. */dev/sda5*. The naming convention for VBDs differ from physical partitions, e.g. */dev/xvda5*.

4.2 Virtual machine instance types

There are various kinds of roles in which virtual machines can be applied, and there are some in which virtualization could not provide sufficient replacement of native installations. Each role can be mapped to a workload profile based on the processing, memory, disk I/O or network I/O need. Writing down a table of expected machine roles is often part of the design process. Table 4.1 is an example of such a table.

After characterising the expected workload model, the virtual machines can be combined into the same physical hardware to achieve the optimal utilization rate. For instance, a file server role with low CPU and high network I/O load can be combined with an SW compiler virtual machine. These two machines can then be run on the same hardware.

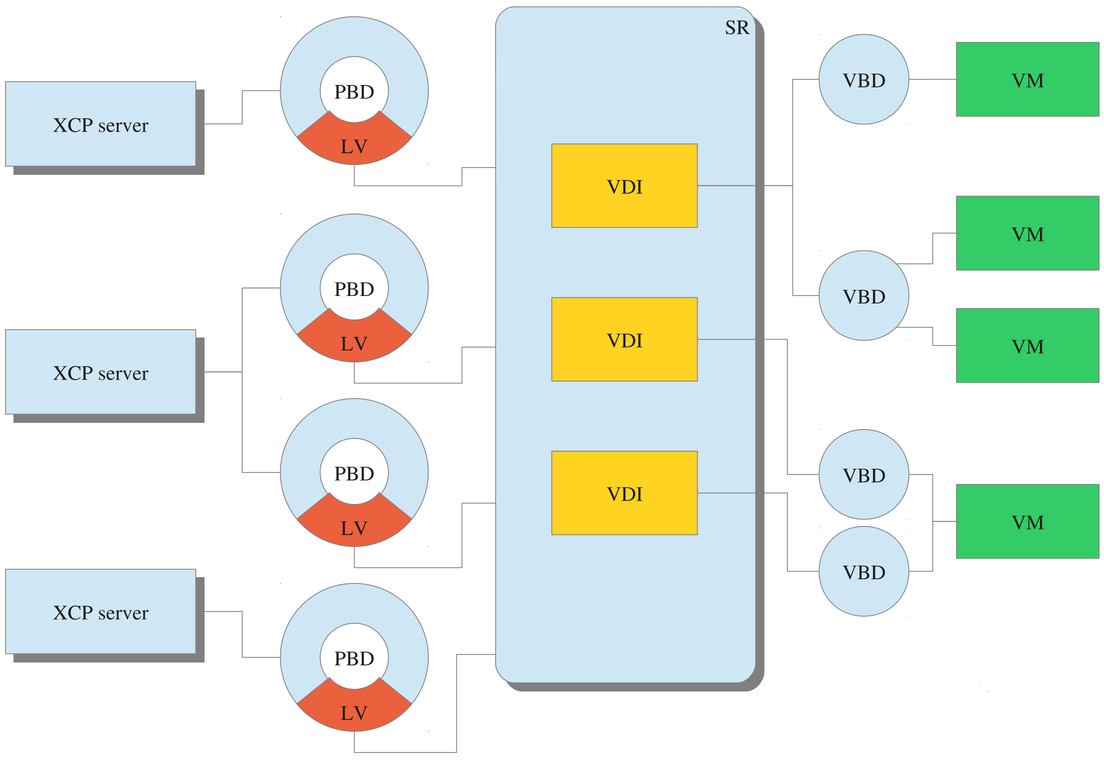


FIGURE 4.2: Virtual disk subsystem. (Similar configuration is used in the prototype.)

As a practical example, Amazon Elastic Compute Cloud (EC2) defines six families of VM instance types (3) such as:

- Standard
- Micro
- High-Memory
- High-CPU
- Cluster Compute
- Cluster GPU

One to four sub-instances are provided for each category to meet the customers' needs. In total there are 13 different VM types that can be selected based on the needs. The VM instances at Amazon are hourly priced (4).

Similarly to the role categorization, the time-sharing operation can also be applied. Typically, peak and off-peak periods can be easily predicted. Knowing these periods, the utilization can be improved by resource and priority re-allocation. For instance, on weekdays daytime the virtual instances of office desktop may have higher priority compared to a virtual machine doing a long simulation. On weekends, the simulation process may be adjusted to be run with higher priority.

TABLE 4.1: Virtual machine roles and their typical workload types

Role	CPU/GPU	Memory	Disk I/O	Network I/O
Sw compilation	●●●○○	●●●○○	●○○○○	●○○○○
Office Desktop	●●○○○	●●○○○	●●○○○	●○○○○
Web server	●●○○○	●○○○○	●●○○○	●●●○○
File server	●○○○○	●○○○○	●●●●○	●●●●●
Simulation	●●●●●	●●●●●	●●○○○	●●○○○

4.3 Requirements

The specification and requirements for the environment were not explicitly defined by any customer. A list of specified internal requirements is as follows:

1. The designed system must be capable of providing up to 30 virtual machine instances that can be used for executing different tasks.
2. The designed system must be easily integrated to existing SW and HW infrastructure and should be compatible with it.
3. The designed system must be easily extendible and scalable on demand.
4. The designed system must be easily reconfigurable for various kind of tasks e.g. SW build environment.

5. Assessment of potential Single Point of Failure (SPOF) must be made. The number of critical components should be reduced or eliminated. Fault tolerance should be provided.

4.4 Resource usage considerations

Due to that the Virtual Machine Cluster (VMC) servers are built using standard desktop class computers, special care should be taken when selecting the SW components. The limited memory and CPU processing capacity compel us to seek for a less resource intensive, lightweight virtual environment.

A simple desktop environment memory usage comparison was made by using the *gnome-system monitor*[1] to compare memory consumption of different Linux operating systems. The 64-bit version of installer ISO disk image was downloaded and booted in Virtualbox application. Only the default services were running and the *gnome-system monitor* application which displayed the actual resource usage. The comparison showed that the least resource hungry desktop environment is XFCE, using about 38% of memory compared to Unity which is the default environment in Ubuntu release 12.04. The results of the comparison can be seen in table 4.2. A similar comparison was made in 2010 by Phoronix with the similar results (30). The same study also showed that there were no significant power consumption differences between the tested desktop environments.

TABLE 4.2: Memory consumption comparison of Linux desktop environments

OS version (64-bit)	Desktop Environment	Used memory (MiB)
Linux Mint LMDE (12.04)	XFCE	191.9
Linux Mint LMDE (12.04)	MATE/Cinnamon	293.8
Ubuntu Desktop 12.04 LTS	Unity	502.4

[1] https://launchpad.net/gnome-system-monitor

4.5 Implementation plan

It was decided that the prototype will be integrated into the production environment gradually. The demand for the virtual environments is small at the beginning, and growing progressively. The implementation steps are scheduled synchronously with the needs. In the early phase there is no specific need for disk redundancy options and the HW utilization does not require performance adjustments. These can be added later without disturbing the existing virtual configurations.

The implementation work is divided into the following phases:

Phase 01: Collect and define internal requirements for the future system. Study existing practical virtualization technologies and tools to be used and select the most suitable one that can be integrated easily to the existing infrastructure.

Phase 02: Order the selected HW resources. Prepare the environment for quick deployment when the HW resources are available.

Phase 03: Implement a standalone virtual server for reference using only a local disk. Create first instance of general virtual machine for templating base. This VM can be used for demo purposes.

Phase 04: Use template VM to create a production version of VM and clone multiple instances according to the needs. At this phase it is still the standalone virtual server is in use with local disk repository.

Phase 05: Install virtual servers on all the physical machines. Build a distributed file system. Do performance measurements on distributed volumes. Tune configuration parameters if necessary.

Phase 06: Clone as many production VMs as needed using the distributed filesystem. Do performance measurements on production VMs and tune configuration parameters if necessary.

Phase 07: Finalize the environment, extend the VMC with new nodes on demand.

5 Prototype implementation and configuration

After reviewing the design plan, the VMC prototype was decided to be implemented. The building of this small-size cluster is suitable for gaining adequate level of knowledge for further extensions of the prototype. In this chapter the building of the four-machine VMC is presented.

5.1 Selected hardware components

Before any parts are selected, one should clearly define the intended functions. There are multiple operating systems running simultaneously on a single computer. This drives the selection toward quad-core (single computing component with four independent central processing units) system instead of a dual-core alternative. The default configuration consisted of a single Network Interface Card (NIC) and only one Hard Disk Drive (HDD). For the good isolation of functions (virtual server and distributed disk) it was decided to extend the default configuration by adding two more disk drives and two more NICs. In a more isolated system, one can measure performance more accurately without the interference of parallel systems. For example, with a separate network one can ensure that other Ethernet traffic will not alter our measurements. For the data storage device within the distributed array a component with low latency (high Input/Output Operations Per Second (IOPS) parameter) is needed. OCZ[1] Vertex 4 family provides 85000-120000 maximum IOPS with more than 500MiB/s sequential reading performance (27).

The price trend and capacity of the commodity computer is following Moore's law. One can predict the future desktop-class computers will likely be equipped with better performing components.

As a result of hardware component selection, four pieces of Dell Optiplex 990 desktop Personal Computers (PCs) were ordered for building the VMC system with the following components:

[1] http://www.ocztechnology.com

- Intel® Core™ i7 quad core processor
- 8GB of Double data rate type three synchronous dynamic random access memory (DDR-3 SDRAM)
- 500GB SATA Hybrid Hard Drive
- One Gigabit Ethernet (GbE) network interface

In addition to the default desktop configuration, the following components were added:

- Two additional GbE network interface cards
- Two OCZ Vertex-4 Solid-state Drives (SSDs)

5.2 Hardware installation

The SSDs arrived separately with all the necessary screws and brackets. Installation went rapidly with the help of a single Phillips-head PH1 size screwdriver tool.

After installing the SSDs, a quick test was made of the new drives using Linux Disk Utility program[2]. It was observed that only one of the four port was supporting SATA-3 (6Gbps) transfer data rate. All the other three ports supported only SATA-2 (3Gbps). This limitation was not mentioned in the product specification of the computer (16). RAID0 configuration was tested using two similar SSDs which resulted 8.2% reading speed improvement. Configuring SSD and SATA hard disk into RAID0 array performed even worse than a single SSD drive. The buffered average read rates can be seen in table 5.1.

5.2.1 Experiences with RocketRaid SATA adapter

Due to the previously seen SATA port speed limitations, an alternate solution was tested to get the maximum performance out of the Vertex-4 SDDs. Two pieces of RocketRAID 640-6Gb/s PCI-E Gen2 RAID Host Bus Adapter (HBA) (21) were purchased and tested.

[2]http://git.gnome.org/browse/gnome-disk-utility

TABLE 5.1: Average disk reading rate comparison on Dell Optiplex 990 at different SATA ports with single and RAID0 configurations

Test 1		
Port number	Average Read Rate (MiB/sec)	Disk type
0	438.2	OCZ-Vertex4
1	210.6	OCZ-Vertex4
2	282.8	OCZ-Vertex4
3	215.4	OCZ-Vertex4
Test 2		
Port number	Average Read Rate (MiB/sec)	Disk type
0	515.3	OCZ-Vertex4
1	91.7	500 GB SATA HDD
2	282.5	OCZ-Vertex4
3	Not Tested	CDROM
Test 3 - RAID0 configuration		
Port number	Average Read Rate (MiB/sec)	Disk type
0	Total: 557.8	OCZ-Vertex4
2		OCZ-Vertex4
Test 4 - RAID0 configuration		
Port number	Average Read Rate (MiB/sec)	Disk type
0	Total: 318.0	OCZ-Vertex4
1		500 GB SATA HDD
2		OCZ-Vertex4

The HBA has 4 SATA ports and advertised to be capable of up to 6Gb/s transfer speed. It requires a kernel module to be compiled from source. The compiled kernel module worked with a generic kernel, but did not work with the Xen special kernel. It was loaded without any error message but the adapter could not connect to the disk. It printed the following error to the syslog:

```
rr64x:[0 0 f] failed to send 1st FIS
rr64x:[0 1  ] failed to hard reset.
rr64x:[0 1  ] failed to perform port hard reset.
```

With the generic kernel, the reading performance reached 222MiB/s on PCI express slot 1 and 345MiB/sec on PCI express slot 4. This achieved transfer speed with the generic kernel was less than expected. The HBAs were decided not to be used. The SSDs were decided to be be connected to the best performing SATA ports on the motherboard (port0,port2).

5.2.2 Physical placement of machines

All of the four desktop machines were placed on a working desk for the SW configuration and for the performance measurements. With the help of a four-port Keyboard Video and Mouse (KVM) switch, only one display and keyboard was needed. For the normal operation, the machines were relocated to a server room into a server cabinet without local console access.

5.3 Selected virtual server platform

The XCP[3] is an open source server virtualization and cloud computing platform. It is derived from Citrix Xenserver commercial product (44) and built on the top of Xen hypervisor. The functionality is so much similar to Citrix XenServer that all the XenServer manuals also apply to XCP. For instance XenServer 6.0 documents cover XCP 1.5 (42).

The first stable version (1.0) of XCP was announced in 2011. At the time of writing, the latest version is 1.5 (released in 2012). The hypervisor is built on the top of CentOS 5 operating system. Besides the ISO installer, the *xcp-xapi* standalone toolstack and server daemons can also be installed on the top of an existing Ubuntu or Debian installation by using the upstream Linux kernel (46). XCP was selected to be utilized for VMC realization because of its simplicity with clean installation.

The XCP was deployed on each of the four computers in the cluster. XCP is easily installable from ISO image and includes all of the necessary components for the target cluster.

[3]http://www.xen.org/products/cloudxen.html

It includes the Xen hypervisor, lightweight dom0 privileged domain and powerful command line control interface (xe).

5.3.1 XCP management interfaces

XCP includes Xen Application Programming Interface (XAPI) toolstack which allows managing the server either by using the command line, Graphical User Interface (GUI), or web management tools.

Xe (xe) is a powerful command-line interface which talks to both hosts and resource pools over https, invoking XenAPI operations. Commands can be executed both from within DOM0 and from remote hosts. Tab completion is available which is increasing usability and speed. *Xe* enables the writing of the scripts for automating tasks and allows the integration into an existing Information Technology (IT) infrastructure. *Xe* is installed by default on XCP server. A stand-alone remote version is also available for Linux (13).

XenWebManager (51) is an open source web based application written in Python CherryPy (12) web framework with graphical interface to manage XenServer / XCP hosts over the network. From the sourceforge[4] website it is possible to download a complete XVA image which can be directly imported and run from within XCP server. Xenwebmanager virtual image was loaded into one of the servers. It was tested and worked with small adjustments. The firewall (*iptables* service) must be configured or disabled to enable TCP ports for the web server. Corporate network proxy usage must be disabled in the client browser settings. Xenwebmanager provides the same functionality as XenCenter and Openxenmanager, except the VNC console which was not functional.

Citrix XenCenter (47) is a proprietary windows-native graphical user interface. It can be freely downloaded (after registration) from Citrix web page (48). It supports and autodetects (automatically identifies) both Remote Desktop Protocol (RDP) and Virtual Network Computing (VNC) protocol as a graphical console.

[4]http://sourceforge.net/

Openxenmanager (29) is a multiplatform tool written in Python with graphical interface to manage XenServer / XCP hosts over the network. OpenXenManager is an open-source clone of Citrix XenCenter windows application.

Xen VNC Proxy (XVP) (23) is a suite of open source programs for management of virtual machines running on XenServer and XCP developed by Colin Dean. The console allows to operate and access virtual machines through a web browser.

5.3.2 XCP resource pool

XCP resource pool consists of multiple XCP/XenServer host installations (up to a maximum of 16), merged into a single entity. The resource pool enables VMs to be started on an automatically selected server which has sufficient memory and the available resources. This feature requires a shared storage and each CPU from the same vendor and the same feature flags. AMD-V and Intel VT CPUs cannot be mixed. In the pool at least one physical node has to be selected as the master. Only the master node accepts commands through the administration interface. In case of a failure of the pool master, re-election takes place. The automatic re-election is only available when the *High Availability* feature is enabled (13).

5.3.3 XCP installation

The Xen Cloud Platform 1.5 Beta was available from the Xen internet page (43). The size of the disk image was 362MiB. The installation was carried out from Compact Disc (CD) media and took approximately half an hour. XCP uses the whole physical disk and does not allow to leave another operating system on a different partition. The dom0 physical disk footprint is less than 200MiB. This is enough to accomodate the hypervisor, necessary Xen services, command line tools and a minimalistic console user interface. The rest of the physical hard disk space can be used to store virtual disks and/or ISO images in a SR. A few screenshots were taken during XCP installation process (see appendix 3).

TABLE 5.2: Logical partition requirements and priorities

Logical partition requirements and priorities			
Purpose	Size approx.	Fault tolerance	R/W performance
Dom0 boot partition	4GB	YES	YES
ISO storage repository	100GB	NO	NO
VHD storage repository	100GB	YES	YES
Shared working read area	15GB	YES	YES
Shared working read/write area	15GB	NO	YES

5.4 Selected distributed disk solution

The option of having a distributed array of disks provides the most flexible and scalable solution for the virtual machine cluster. The Gluster[5] distributed filesystem has been selected for use as a core of the designed distributed disk array. The requirements for the logical volumes can be grouped into the following categories: size, fault tolerance and I/O performance. Table 5.2 shows an example arrangement of the logical volumes for the system. For the volumes on the gluster server, XFS formatted Logical Volume Manager (LVM) partitions (volumes) were used.

5.4.1 Gluster installation

The installation of the Gluster server packages on the XCP servers was slightly more difficult than the installation on traditional Linux distributions. Because of the security reasons, the *yum* (package-management utility) manager does not allow to install any packages on the server by default. This can be enabled by editing the corresponding *yum* repository file: `/etc/yum.repos.d/CentOS-Base.repo`. The following steps should be performed on each GlusterFS server nodes.

```
root@xcp# sed -i -e "s/enabled=0/enabled=1/" /etc/yum.repos.d/
                                                CentOS-Base.repo
```

[5]http://www.gluster.org

In corporate networks, a proxy server is often used to connect to the Internet and protect their internal network(s). The proxy usage can be enabled (for host *proxyhost* with port *proxyport*) by writing:

```
root@xcp#  echo proxy=http://proxyhost:proxyport/ >> /etc/yum.conf
```

Once the installation is enabled, one can continue with installing development tools for the Gluster source compilation.

```
root@xcp# yum groupinstall 'Development Tools'
root@xcp# yum groupinstall --skip-broken 'Development Libraries'
root@xcp# yum install python-ctypes
```

Resolving software dependencies is followed by the downloading source code, compilation and installation. This can be done with the following commands:

```
root@xcp# wget http://download.gluster.org/pub/gluster/glusterfs/
                                    LATEST/glusterfs-3.3.0.tar.gz
root@xcp# tar -xvf glusterfs-3.3.0.tar.gz
root@xcp# cd glusterfs-3.3.0/
root@xcp# ./configure
root@xcp# make
root@xcp# make install
```

5.4.2 Firewall configuration for Gluster

By default the needed Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) ports are not open in the firewall (iptables) configuration. These can be enabled and verified by using the following commands as a superuser (root):

```
root@xcp# iptables -F
root@xcp# iptables -A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp
                                                 --dport 24007:24047 -j ACCEPT
```

```
root@xcp# iptables -A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp
                                                         --dport 111 -j ACCEPT
root@xcp# iptables -A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp
                                                         --dport 111 -j ACCEPT
root@xcp# iptables -A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp
                                                 --dport 38465:38467 -j ACCEPT
root@xcp# service iptables save
root@xcp# service iptables restart
root@xcp# service iptables --list
```

5.4.3 Building up the storage pool

A storage pool is a trusted network of several storage servers, called peers. Before configuring any GlusterFS volume, a storage pool must be set up. To add storage servers to the pool, the *gluster probe* command is used in the following way.

```
root@xcp# gluster peer probe 10.10.10.64
Probe successful

root@xcp# gluster peer probe 10.10.10.65
Probe successful
```

The server from which the commands are issued will automatically be part of the storage pool and does not have to be probed. The peer status can be checked by using the following command:

```
root@xcp# gluster peer status
 Number of Peers: 2

 Hostname: 10.10.10.64
 Uuid: ff073a59-56ca-47e2-8b5a-f50164a72aee
 State: Peer in Cluster (Connected)

 Hostname: 10.10.10.65
 Uuid: c64c0b22-2c8d-4163-a386-7e9e54169970
 State: Peer in Cluster (Connected)
```

When a storage server is not needed anymore in the pool, it can be removed using the *gluster detach* command. The command will only work if all the peers of the given pool are in *connected* state.

5.4.4 Setting up storage volumes

Gluster storage volumes can be created by using *gluster volume create* command. In the Gluster a *"brick"* is the basic unit of storage, represented by an export directory on a server in the trusted storage pool. A server can accomodate one or more bricks. Gluster storage volumes of the following types can be created:

- Distributed — Distributes files throughout the bricks in the volume.
- Replicated — Replicates files across bricks in the volume.
- Striped — Stripes data across bricks in the volume.
- Distributed Striped — Distributes data across striped bricks in the volume.
- Distributed Replicated — Distributes files across replicated bricks in the volume.

The first two volume types were tested in the VMC prototype.

In a distributed volume, the files are spread randomly across the bricks in the volume. Creating distributed volumes is recommended when the scalable storage is needed and redundancy is not important (17). Figure 5.1 shows how a simple distributed volume is built.

The replicated volumes create copies of the files across multiple bricks in the volume. Creating replicated volumes is recommended in the environments where high-availability and high-reliability are critical (17).

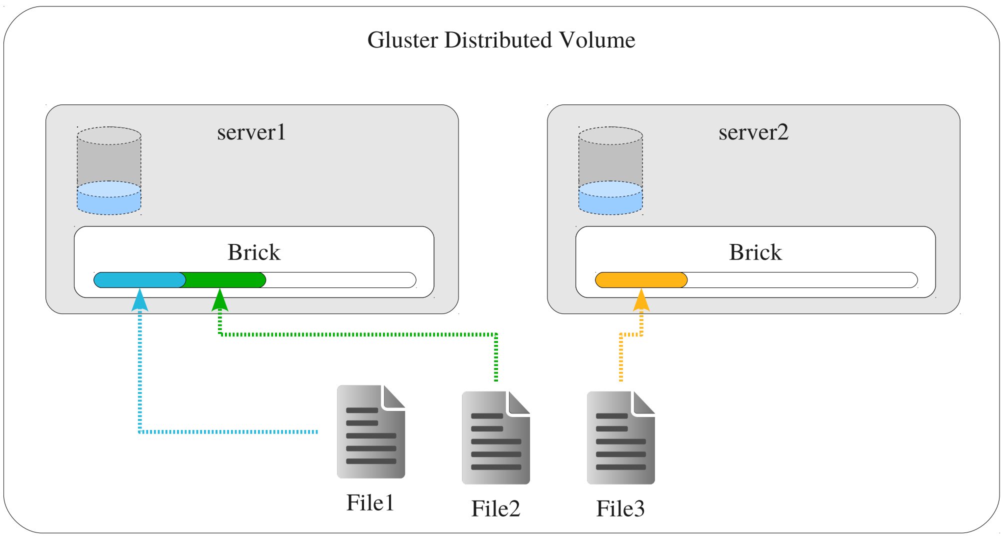


FIGURE 5.1: Illustration of a Gluster distributed volume

5.5 Fine tuning and configuration of virtual servers

After finishing all installation steps on the server, one can continue with setting up storage repositories and then installing the virtual machines.

Once a virtual machine instance is up and running, additional configurations can be made. These include configuring network (hostname, IP configuration), loading paravirtual drivers, setting up Network Time Protocol (NTP). The newly created, configured VM can be cloned and copied between the servers at will. In this phase, distributed disk volumes were not used yet.

5.5.1 Installing VM from ISO image

The VM installation can be performed by using an ISO 9600 formatted CD/DVD-ROM image located on a shared network location. Samba[6] network file share was used for storing installation media. The installer stopped at the bootloader due to a driver problem (incompatible or missing device driver). The same problem was noticed by testing several ISO images through various repositories. (NFS share, local repository, physical CD) The problem was solved by giving the following kernel parameters to the installer: `noapic nolapic`. The first one disables the Advanced Programmable Interrupt Controller (APIC) and the

[6]http://www.samba.org

second one disables the local APIC (10). Kerner parameters can be edited by pressing the *Tab* key when the bootloader lists the startup options.

5.5.2 Installing Xenserver tools

Installing XenServer Tools (PV Drivers) enhances the network performance and disk I/O operations without the overhead of traditional device emulation (50). XCP includes the xenserver tools ISO image (xs-tools.iso) which can be easily loaded and installed onto the guests. The Xenserver tool installation was performed on a Linux HVM guest (Linux Mint 13 MATE). Several additional VM features unveiled that were listed as `<not in database>` before (guest OS version, IP address). The VM *allowed operation* list now includes `pool_migrate`, `suspend` and `checkpoint`. The number of VBD and Virtual (network) Interface (VIF) devices has increased. For the full list of additional features, consult appendix 2. On the general tab in the properties pane within OpenXenmanager the message "Tools not installed" has been replaced by "Optimized (version 1.4 build 53341)" and the OS version is also displayed. The machine suspend and resume operations were tested and worked through OpenXenmanager.

5.5.3 Time synchronization between Dom0 and VMs

By default, the clocks in a paravirtualized Linux VM are synchronized to the clock running on the control domain and cannot be independently changed. The behaviour is controlled by the setting `/proc/sys/xen/independent_wallclock`. This mode is a convenient default, since only the control domain needs to be running the NTP service to keep accurate time across all VMs (49). With other types of VMs, running an NTP client daemon is recommended to keep the date and time synchronized.

5.5.4 Forced file sytem checking at every boot

During the experiments, the forced hard VM shutdown often resulted in errors in the journaling filesystems. The Fourth extended filesystem (ext4) has a parameter called *"Maximum mount count"* which defines the number of the mounts after which the filesystem is checked (by *e2fsck*). This parameter can be read by using *dumpe2fs* and set by using *tune2fs* Linux filesystem utility programs as the following lines show:

```
users@desktop ~ $ sudo dumpe2fs -h /dev/sda1 | grep -i 'mount count'
dumpe2fs 1.41.14 (22-Dec-2010)
Mount count:              13
Maximum mount count:      21
user@desktop ~ $ sudo tune2fs -c 1 /dev/sda1
tune2fs 1.41.14 (22-Dec-2010)
Setting maximal mount count to 1
```

The file system check at every boot can be forced by adjusting this parameter to *1* and can be disabled completely by using value *-1* (37). The forced filesystem check was set in each VM root filesystem to increase the reliability and recovery in case of failures. The execution time depends on the size of filesystem, the number of inodes and other file system parameters. In the case where the virtual disk size is relatively small ($\leq$ 20GB at maximum), the file system check took approximately 5-10 seconds to execute. In spite of frequent filesystem checks, a forced hard reset may cause irrecoverable file system errors that possibly prevent the OS boot. This behaviour was also observed a few times during the experiments. As a workaround solution a template of VM was created which is not in normal use but in recovery. The template VM is normally in shutdown state. It is used only as a security copy to overwrite the corrupted VM.

5.5.5 Automatic VM startup on server boot

It is often useful to have some of the virtual machines started up on server boot without manual interaction of the administrator. In the earlier versions of Xenserver there was an option to enable the VM autostart, but since Xencenter 6 the option has been removed. Bill Carovano explained the reason in Citrix forum:

> It was removed due to bad interactions with other features like High Availability, Rolling Pool Upgrade, and integrated DR. Auto-start settings made VMs start up in an uncontrolled fashion, at the wrong times or on the wrong hosts and basically broke these other features (41).

As a replacement for the missing feature we created a simple startup shell script that launches the selected VMs. This shell script (autostart.sh) is called at server boot by placing it to */etc/rc.local*.

In the simple script the *xe vm-start* command is called after a 40 seconds sleep time. This 40 seconds was proven to be enough for all the xen services to be started up. This amount of delay is an individually tested, non-deterministic parameter which may vary between different systems. The following shell script was used for automatic VM startup:

```
#!/bin/bash
export PATH=/opt/xensource/bin:/usr/local/sbin:
/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
sleep 40
xe vm-start uuid=UUID
```

The Universally Unique Identifier (UUID) of the VM can be figured out and copied from the output of the *xe vm-list* command.

6 Performance tests with VMC

In this chapter an example use case is demonstrated which is utilizing the newly built VMC. A distributed build system was chosen to be the first use case of the prototype virtual machine cluster.

During the sofware building process, source code files are converted to binary executable code. It is the speed that is the most important and critical goal to achieve by any building framework today. This demand is particularly critical when someone is considering agile and continuous integration based software development where it is common to have frequent builds. In the tested scenario the building process workflow steps are sent to separate (virtual) machines for execution. In the current research the author seeks for an answer whether any significant speed improvement can be achieved compared to single machine execution. The definition of the term *significant* is when the improved build execution time (reciprocal of building speed) is at least 20% less than the single machine execution time. Mathematically expressed in 6.1:

$$t_1 = \frac{1}{n} t_0 \text{ and } n \geqslant 1.25 \tag{6.1}$$

where: t_0 and t_1 are the compilation execution times before and after improvement.

6.1 Research approach

The applied research approach is a quantitative, experimental approach. With quantitative experiments, a sufficient amount of data is collected. The hypothesis is either justified (proved) or rejected based on the statistical analysis of the data. Even if there are indefinitely many (unknown) complex functions explaining the set of data, experiments may be repeated and theories can be verified by using this method.

In the current case there were two alternative models compared. The first one is a single virtual machine mode. It has a similar CPU and memory configuration in the VM as in the native physical installation. The second one is using multiple VMs working parallel orchestrated by *distcc*, a distributed C compiler. [1] The preliminary expectations on the compilation speed is that the distributed build is the fastest followed by the single physical installation and the single virtual machine. If the single physical and single virtual machines are very similar, one can expect only minor (2%..5%) difference because Xen paravirtual driver is installed in VM. The applied research method can be seen in figure 6.1.

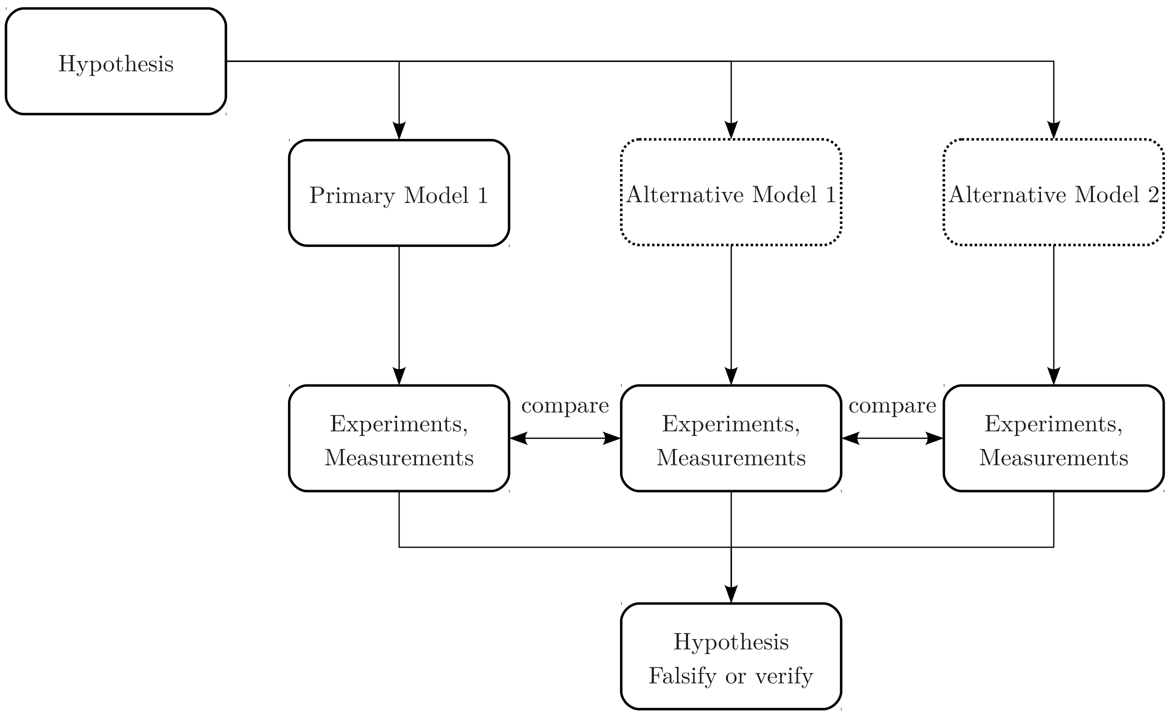

Hypothesis
Primary Model 1
Alternative Model 1
Alternative Model 2
Experiments, Measurements
compare
Experiments, Measurements
compare
Experiments, Measurements
Hypothesis Falsify or verify

FIGURE 6.1: Research approach for verifying the hypothesis

The hypotheses is that the equation (6.1) is valid for at least one of the alternative models against the single machine execution (primary) model.

6.2 Building task

In order to determine and compare the performances, Linux 3.4.6 kernel compilation was selected as the task to be executed. All the source code files, temporary artifacts and

[1] https://code.google.com/p/distcc

result binaries were stored on the local filesystem (ext4) on Vertex-4 SSD. The machine was not restarted between the measurements because it was seen that the repeated execution time is nearly identical in both cases.

6.3 The build speed of the single machine native installation

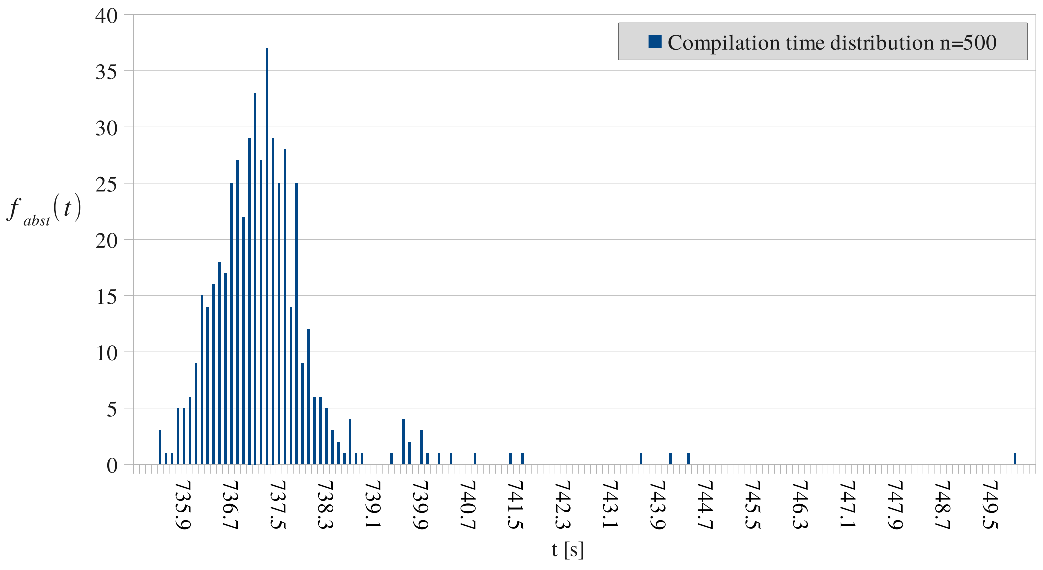
The primary model to which the alternative model improvements are compared to is based on a single native physical installation of Linux machine with the following parameters:

CPU type	Intel 4-core (8 threads)
Allocated memory size	8 GB
Operating system	Linux Mint 13 (64bit)
Storage type	Vertex-4 SSD
Executed command	make -j10

The compilation task was executed n=500 times using j=10 parallel jobs. The execution time samples were recorded into output files. The population was later processed and analysed. The number n=500 is chosen because it is not only providing a good statistical amount of samples but also tests the stability of the system. The minimum and maximum execution time of all samples were $t_{min} = 735.83s$ and $t_{max} = 750.27s$. The calculated sample mean $\bar{a} = 737.63$ with standard deviation $s^2 = 1.124$. The absolute frequency of execution time samples can be seen in figure 6.2. The performance measurement loop and the postprocessing scripts can be found in appendix 5.

The machine load from one compilation cycle is presented in figure 6.3 where CPU utilization and disk Transactions Per Second (TPS) (read and write) are shown. The *Iostat*[2] program was used to collect samples in two-second intervals for the whole duration of the compilation task. In the stacked area chart the user level application is marked in blue (user) and the kernel level executions (system) marked in red. The percentage of the time that the CPUs were idle during which the system had an outstanding disk I/O request is

[2]http://linux.die.net/man/1/iostat



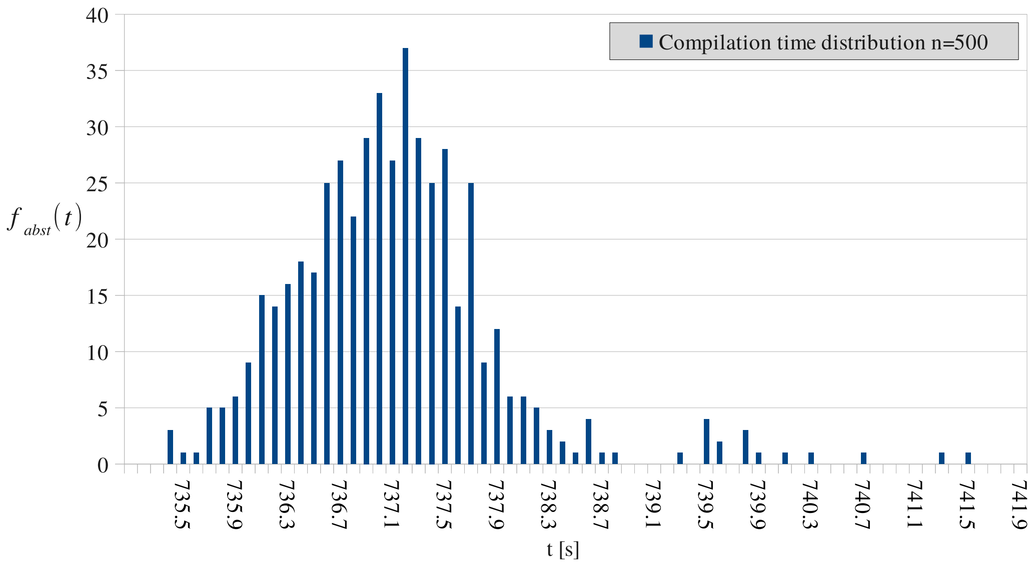


FIGURE 6.2: Compilation time distribution on native operating system

marked in yellow. The remaining percentage —which is marked in green— represents the CPU idle state. A highly loaded CPU can be seen for the vast majority of the samples while significant disk device utilization is only visible near the end of the task for about 20 seconds (10 samples). The nature of the example compilation task contains mostly CPU intense operations with few disk I/O transactions and no netwok usage (if source code files and results are stored locally).

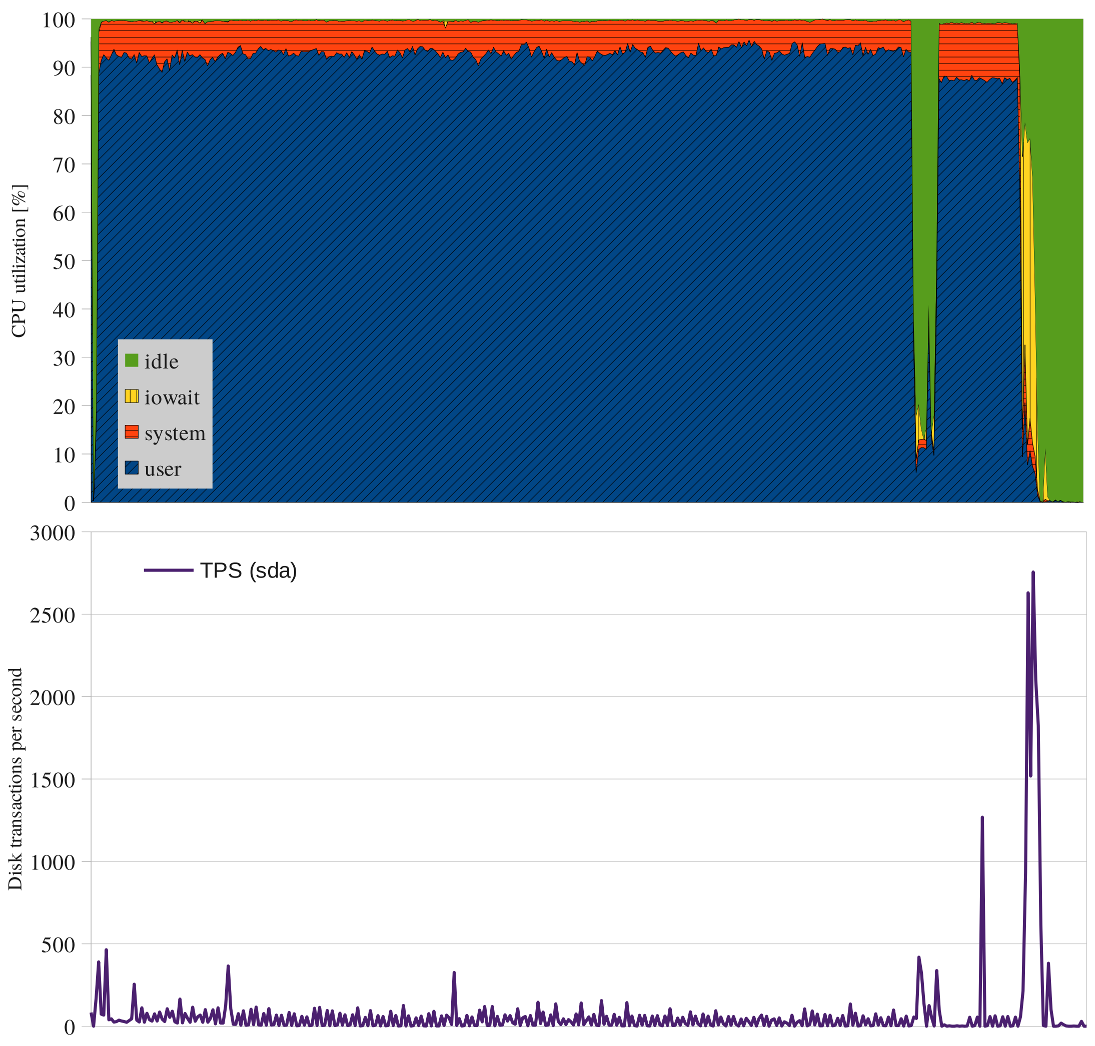


FIGURE 6.3: Compilation load on single machine with native operating system

6.4 Measurement results of build speed from single virtual installation

The same compilation job was executed n=500 times on a console only (without graphical desktop environment) virtual machine instance with the following parameters:

number of VCPUs	8
VCPU priority	highest (65535)
Allocated memory size	7.0 GB
Operating system	Ubuntu 12.04 (64bit) console only
Storage type	local virtual image on Vertex-4 SSD
Executed command	make -j10

The minimum and maximum execution time of all samples were $t_{min} = 772.61s$ and $t_{max} = 785.31s$. The calculated sample mean $\bar{a} = 777.87$ with standard deviation $s^2 = 2.128$. The absolute frequency of the execution time samples can be seen in figure 6.4. The histogram shape is asymmetrical and skewed which does not show Gaussian (normal) distribution in population.

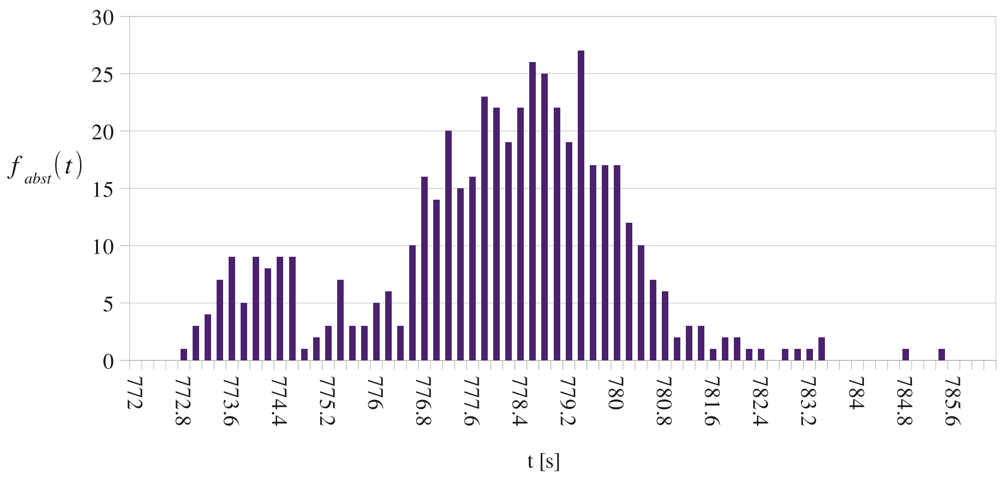


FIGURE 6.4: Compilation time distribution on single virtual machine

The mean execution times from the measurements show that the single virtual machine is on the average 5.45% slower than the single native machine. The variance of the execution times are 89% greater in the virtual machine compared to the native one.

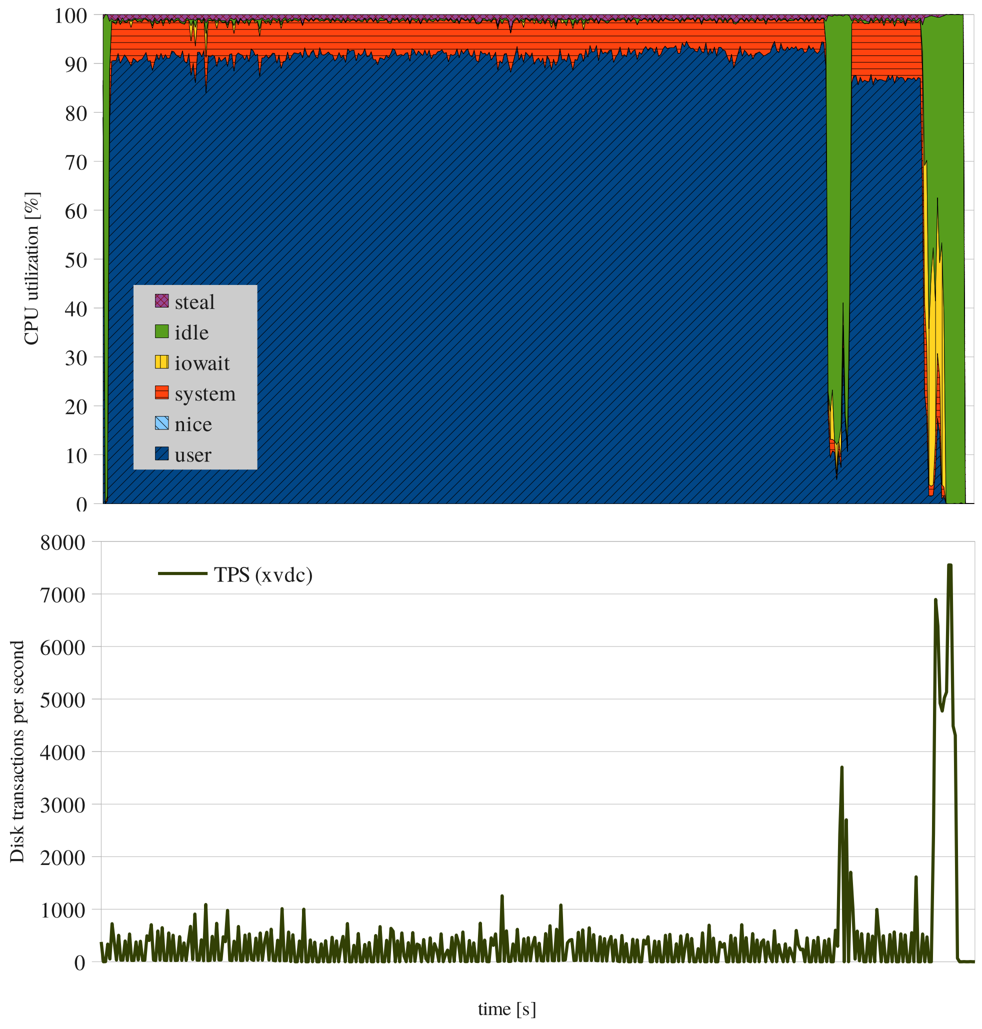


FIGURE 6.5: Compilation load on single virtual machine

XCP Dom0 resides within a small fixed portion of physical memory. Therefore, the maximum allocatable memory is less than the physical (8GB). Consequently, the virtual memory size was limited to 7.0 GB in this test. The load pattern of the virtual and native compiler machines look very similar. In both cases the CPU load was high and near the maximum. The magenta color is only visible on the virtual machines which shows the *"steal"* percentage of the virtual CPU. In other words, it is the time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor (25). The virtual disk utilization shows significantly higher numbers on the virtual machine compared to the physical, peaking 7000 TPS. These numbers mean logical transactions where multiple logical requests can be combined into a single I/O request to the actual physical device. In spite of the greater number, the actual transferred bytes are identical.

6.5 Measurement results from distributed build

The compilation task was executed n=500 times and the execution time samples were recorded in the same way as in the previous test. The population was later processed and analysed. The minimum and maximum execution time of all samples were $t_{min} = 353.44s$ and $t_{max} = 424.11s$. The calculated sample mean was $\bar{a} = 359.62$ with standard deviation $s^2 = 6.09$. The standard deviation is more than five times larger than in the single-machine compiler, which can be explained by the differences in the system complexity. The distributed build machine has much more variable dependencies, e.g. network latency. The absolute frequency of the execution time samples can be seen in figure 6.6. The shape of the histogram shows a normally distributed population.

The machine load profile of the distributed build in figure 6.7 looks more scattered. It is clearly visible that the load is not well balanced between the master and slave computing nodes. The master has a higher average load because the master node has more and different kinds of tasks to perform. Because of the different roles, it is logical to have the master and computing slave nodes configured differently. A reasonably good solution is to create two kinds of virtual machines or keep the master node machine on a native installation.

The workload is measured on one of the three slave computing nodes together with the master node. Only one node was measured because it is assumed that the other two slaves have similar patterns. The lower, (30%..40%) average CPU utilization was observed and can be seen in figure 6.8. The light blue color represents the user processes with the *nice* priority other than 0. The name *"niceness"* originates from the idea that a process with a higher niceness value is *"nicer"* to other processes in the system as it allows the other processes more CPU time (53). By default, *distcc* version 3.1 uses +10 as *nice* priority in order to avoid an undesirable high CPU load on the systems where *distcc* runs in a background. The *nice* priority can be verified or modified by editing the file */etc/default/distcc*. The default parameters were not altered during the tests unless otherwise noted.

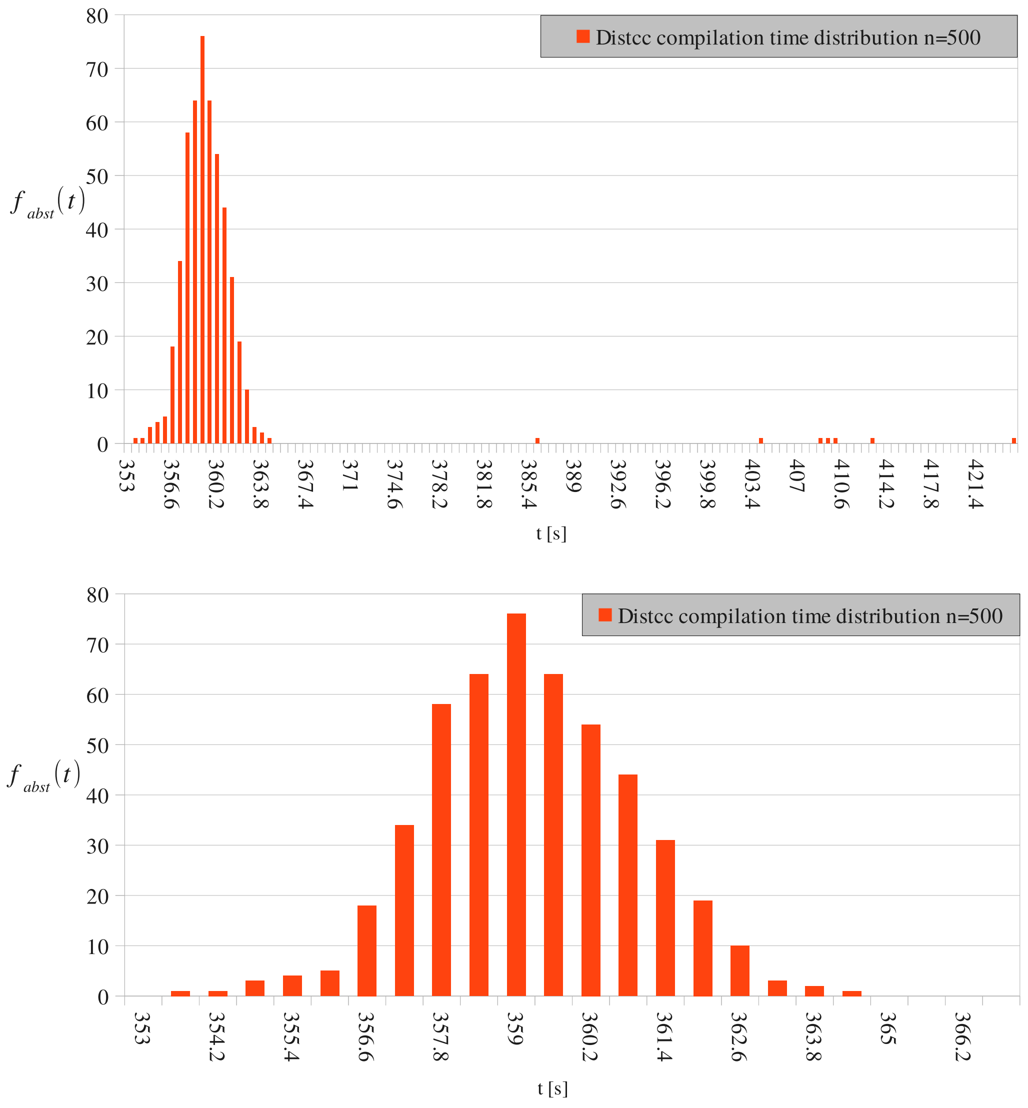


FIGURE 6.6: Compilation time distribution with distcc

From the Ethernet utilization graph (figure 6.7 bottom) it is visible that the Transmit (Tx) compilation job data is 35–50% more than the processed Received (Rx) data on the master node. It is typical that the compiled binary files are smaller in size than the source code. The well balanced distribution can be verified by comparing the amount of slave Rx and master Tx data. In the current setup the master transmitted data is approximately three times more than the slave received data. This is normal, because there are three equal slave nodes. In addition, the workload difference between the slave nodes can be compared using the above mentioned methods.

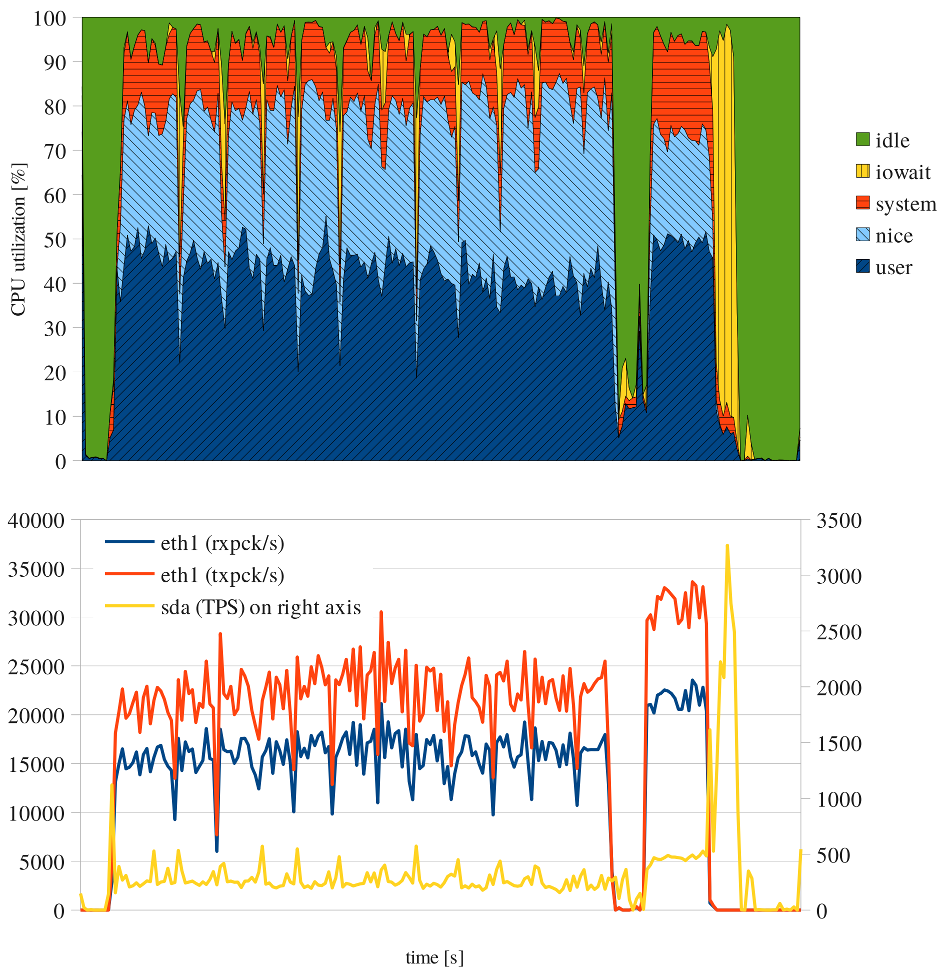


FIGURE 6.7: Compilation load on master node of distributed build system

6.6 Number of parallel compilation jobs

It is important to run as many parallel compilation jobs as possible to achieve the maximum performance. It is logical to assume that considering *n* number of jobs, there is an optimum number where the compilation time curve has a minima. A series of overlapping tests were driven (three runs) to determine the optimum number of the parallel jobs for the system. The results can be seen in figure 6.9.

When launching less than the optimum number of concurrent processes, the CPU spends more time in the idle state. Too few jobs do not optimally utilize the computing resources (CPUs) and cause the tasks being blocked waiting for the disk or network I/O. On the

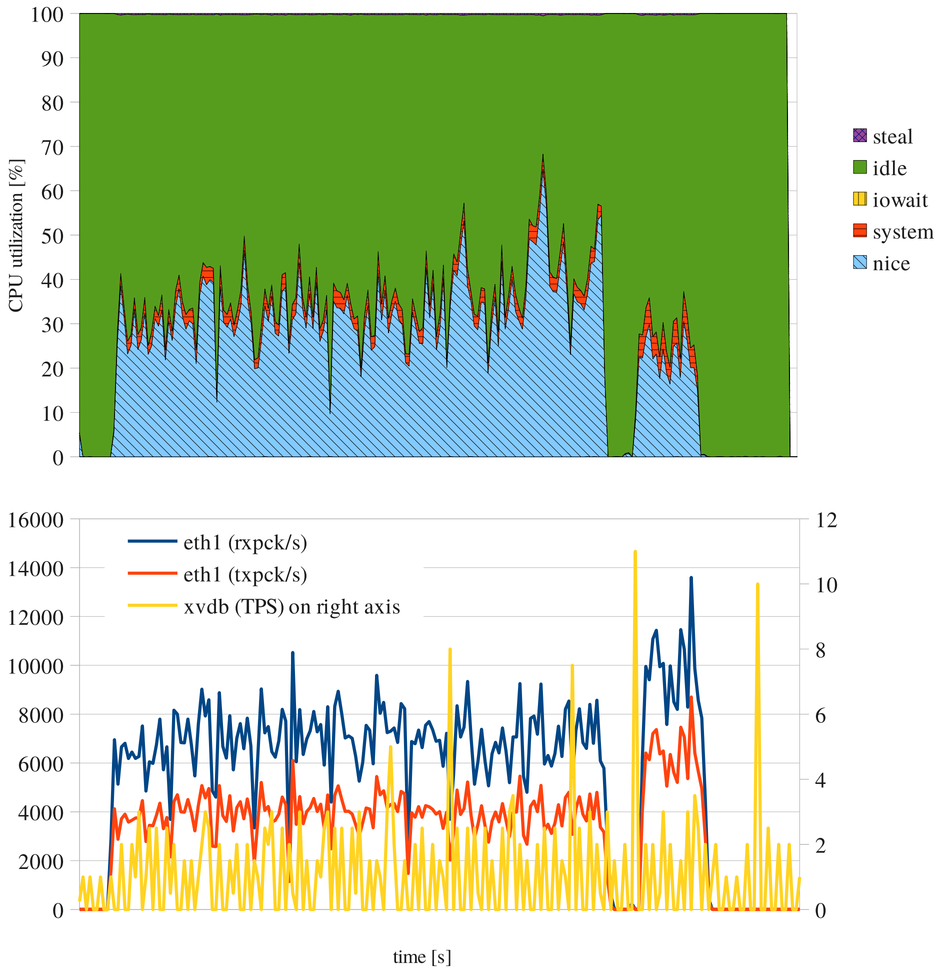


FIGURE 6.8: Compilation load on slave computing node of distributed build system

other hand, if the number of jobs are more than the optimum, the CPU is forced to switch between the concurrent processes. This makes the compilation slower. As it is visible in figure 6.9, the *'overloaded'* system has less negative effect on compilation performance. Even if the number of the jobs is twice the optimum (35 $\Rightarrow$ 70), the compilation time is hardly increased.

6.7 Discussion and summary of tests

The chosen task of the Linux kernel compilation is analyzed in detail with regard to the CPU utilization, disk I/O, and network utilization. Both the virtual machine performance and *distcc* performance matched the expectations. The single virtual machine model was

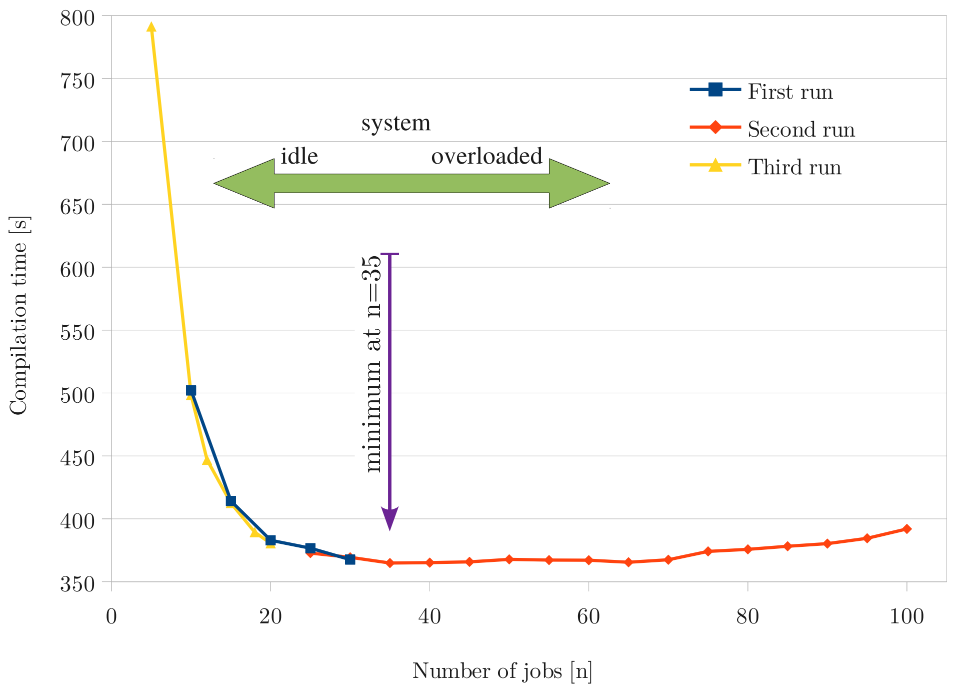


FIGURE 6.9: Distcc number of jobs

slightly slower than the (primary) model without virtualization. The *distcc* distributed compiler was the most performant with the native master and three virtual slave nodes even if the computing virtual nodes had untapped CPU resources. Tuning the *distcc* configuration may potentially further increase the performance by the utilization of the additional exploitable computing resources, i.e. changing *nice* priorities or adjusting the number of the accepted jobs per node. The measurements showed that having too few jobs severely degraded the performance but having too many had less effect on the performance.

The testing method was suitable for determining the resource usage and highlighting the possible bottlenecks of the system. The hypothesis in 6.1 is verified by the measurements. The performance of a single virtual machine was 5.45% less than a native machine. The distributed build system with multiple virtual machines performed the best where the compilation time was 48.75% of the primary model.

7 Performance of distributed storage

In virtual environments, efficient data management is a key component in achieving good performance. Unix/Linux applications typically share data between the tasks using files, e.g. log files. When the tasks are distributed, these files are either transferred from one computational node to another, or accessed through a shared storage system. Otherwise it can be laborious to collect –for example– the output files from hundreds of nodes executing a distributed task.

For the virtual machine cluster prototype the GlusterFS distributed storage solution was designed and implemented. In this chapter the performance measurements of this storage are explained.

7.1 Test environment

A number of performance parameters can be measured on the distributed storage systems including the throughput or the IOPS in various concurrent operation (read and write) scenarios. Each of these can be tested with various storage configurations using distributed, replicated or striped volumes.

Due to the few number of bricks (four), only a distributed volume was tested with no replica and no striping set. A directory structure creator and separate file reader scripts were made for performance tests. In each case the execution time of the script was measured and compared between the local and the distributed filesystems. The server nodes of the distributed filesystem were connected using a dedicated GbE LAN. Throughput tests were not planned because it would only be meaningful on striped Gluster volumes. The summary of the executed tests can be seen in table 7.1. The test scripts and local disk parameters can be found in appendix 6.

TABLE 7.1: Summary of executed tests

Storage type	Generate structure	Find file	Read all files
Local Seagate HDD partition (ext4)	X	X	X
Local SSD partition (ext4)	X	X	X
Network File System (NFS) share (ext4)	X	X	X
GlusterFS distributed partition (xfs)	X	X	X

7.2 Test results

Figure 7.1 demonstrates that the NFS share speed results are showing similarity to the SSD only with higher latency. This is caused by the overhead on the network compared to local SATA bus. The GlusterFS performance is worse (the script execution time is higher) than the NFS share. This can be explained by the lack of a true parallel operation. By analysing the file structure on each brick, it was observed that in spite of the distributed metadata, the directory structure was created on each brick. The SSD find file execution time was significantly better than Seagate HDD. It took only 10.199 seconds to execute.

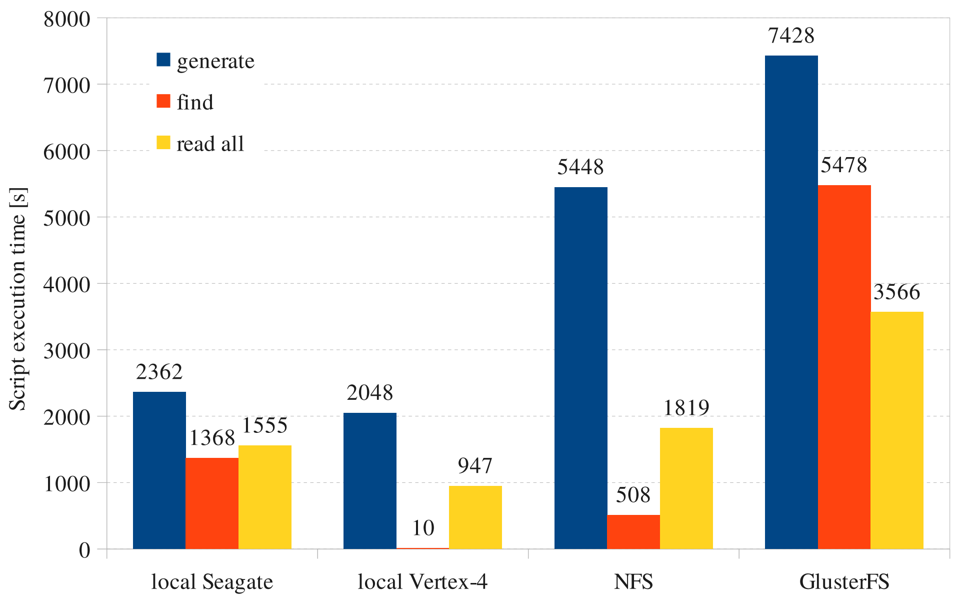


FIGURE 7.1: Test script execution times comparison between local and distributed volumes

8 Conclusion

The designed platform meets all the earlier defined requirements in 4.3 and operates properly at the time of the writing. The explained virtualization solution can be utilized in various environments. However, it requires familiarization with the technology. Studying additional guidelines and developing expertise is advised to properly set it up with confidence. Due to the increased complexity, virtualization might not be suitable for everyone.

8.1 Commodity hardware

Using desktop-class computers for a virtual server is an affordable, yet a powerful solution for both the operating system and the storage virtualization. Some of the HW suppliers –such as Dell– do not always provide full detailed documentation on their components. The SATA bus bandwidth of the motherboard was not enough to benefit the maximum performance of the SSDs.

8.2 Virtual server with XCP

The deployed open source Xen virtual platform is proven to be an excellent choice for those seeking an open source virtualization solution. The platform is well-documented and understandable even to first-time users. The installation is easy and self-explanatory and does not require more than 30 minutes. However, the customization and virtual machine configuration require significantly more time depending on the complexity of the desired system. Various Linux (Ubuntu, Mint, Fedora) distributions and Microsoft Windows 8 evaluation version guest operating systems were installed and tested. The virtual machine migrations across servers were carried out without problems.

8.3 Distributed disk array with Gluster

During the experiments, both distributed and distributed-replicated volumes were created and tested. A simulated node failure was tested with success on a replicated volume by disconnecting the network cable. The performance measurements were only made on one distributed volume and showed significantly lower performance compared to the local volumes. It is assumed that using more nodes in a larger network the benefits of Gluster distributed filesystem can be greater. Although the fault tolerance option is available and tested in Gluster, it was not enabled inproduction environment due to the few number of bricks.

8.4 Improvement possibilities and future plans

The designed and implemented virtual machine cluster with a distributed storage is functioning as expected. It can be further improved by applying one or more ideas of the following possibilities:

- Prepare various VM templates with different roles for quick deployment
- Make individual task based optimizations (such as optimizing process *nice* levels on build machines)
- Extend the cluster with more computing nodes and Gluster bricks
- Install more operational memory to server machines
- Convert existing physical machines to virtual and free-up resources
- Build a cloud platform stack on the top of existing cluster
- Organize a demonstration session and present the advantages of virtualization
- Prepare providing application as service (build private cloud)

REFERENCES

[1] 2010 virtualization and cloud computing survey presented by zenoss, inc. Available at: http://mediasrc.zenoss.com/documents/wp_2010_virtualization_and_cloud_survey.pdf, Retrieved 14 July 2012.

[2] Alphabetical list of institutions that are using hadoop for educational or production uses. Available at: http://wiki.apache.org/hadoop/PoweredBy, Retrieved 02 May 2012.

[3] Amazon ec2 instance types. Available at: http://aws.amazon.com/ec2/instance-types/, Retrieved 17 June 2012.

[4] Amazon ec2 pricing. Available at: http://aws.amazon.com/ec2/pricing/, Retrieved 17 June 2012.

[5] Ata over ethernet. Available at: http://en.wikipedia.org/wiki/ATA_over_Ethernet, Retrieved 01 May 2012.

[6] Ata-over-ethernet enables low-cost linux-oriented san. Available at: http://www.linuxfordevices.com/c/a/News/ATAoverEthernet-enables-lowcost-Linuxoriented-SAN/, Retrieved 17 June 2012.

[7] Ata over ethernet for converged data center networks? no way. Available at: http://searchnetworking.techtarget.com/ATA-over-Ethernet-for-converged-data-center-networks-No-way, Retrieved 14 October 2012.

[8] Ata over ethernet tools. Available at: http://aoetools.sourceforge.net, Retrieved 01 May 2012.

[9] Best distributed filesystem for commodity linux storage farm. Available at: http://stackoverflow.com/questions/269179/best-distributed-filesystem-for-commodity-linux-storage-farm, Retrieved 11 May 2012.

[10] Bootoptions , common kernel options. Available at: https://help.ubuntu.com/community/BootOptions#Common_Kernel_Options, Retrieved 23 June 2012.

[11] Ceph documentation. Available at: http://ceph.com/docs/master/, Retrieved 11 May 2012.

[12] Cherrypy project home - a minimalist python web framework. Available at: http://www.cherrypy.org/, Retrieved 27 June 2012.

[13] Citrix xenserver 6.0 administrator's guide. Available at: http://docs.vmd.citrix.com/XenServer/6.0.0/1.0/en_gb/reference.html, Retrieved 09 June 2012.

[14] Client merged for 2.6.34 - ceph. Available at: http://ceph.com/updates/client-merged-for-2-6-34/, Retrieved 03 May 2012.

[15] Comparing virtualization technologies. Available at: http://www.informit.com/articles/article.aspx?p=1400336&seqNum=5, Retrieved 29 May 2012.

[16] Dell optiplex 990 technical guidebook version 1.5. Available at: http://www.dell.com/downloads/global/products/optix/en/optiplex-990-customer-brochure.pdf, Retrieved 22 July 2012.

[17] Gluster file system 3.3.0 administration guide. Available at: http://www.gluster.org/wp-content/uploads/2012/05/Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf, Retrieved 07 Sept 2012.

[18] The gnu manifesto - gnu project - free software foundation (fsf). Available at: http://www.gnu.org/gnu/manifesto.html, Retrieved 12 May 2012.

[19] A guide to open source software for australian government agencies (second edition). Available at: http://www.finance.gov.au/e-government/infrastructure/open-source-software.html, Retrieved 12 May 2012.

[20] Hdfs architecture guide. Available at: http://hadoop.apache.org/common/docs/current/hdfs_design.html, Retrieved 02 May 2012.

[21] Highpoint rocketraid 600 family series. Available at: http://www.highpoint-tech.com/USA_new/cs-series_rr600.htm, Retrieved 29 June 2012.

[22] History of virtualization. Available at: http://www.vmware.com/virtualization/history.html, Retrieved 30 April 2012.

[23] Introduction to the xvp suite of programs. Available at: http://www.xvpsource.org/, Retrieved 09 June 2012.

[24] An introduction to virtualization. Available at: http://www.kernelthread.com/publications/virtualization/, Retrieved 31 May 2012.

[25] iostat(1) - linux man page. Available at: http://linux.die.net/man/1/iostat, Retrieved 29 August 2012.

[26] Kdenlive - virtualbox images. Available at: http://www.kdenlive.org/user-manual/downloading-and-installing-kdenlive/virtualbox-images, Retrieved 28 July 2012.

[27] Ocz vertex 4 sata iii 2.5" ssd specifications. Available at: http://www.ocztechnology.com/ocz-vertex-4-sata-iii-2-5-ssd.html#specifications, Retrieved 21 July 2012.

[28] Open-iscsi project: Open-iscsi - rfc3720 architecture and implementation. Available at: http://www.open-iscsi.org/, Retrieved 03 June 2012.

[29] Openxenmanager's official wiki. Available at: http://sourceforge.net/apps/trac/openxenmanager/, Retrieved 09 June 2012.

[30] Power and memory usage of gnome, kde, lxde and xfce. Available at: http://www.phoronix.com/scan.php?page=article&item=linux_desktop_vitals&num=1, Retrieved 26 May 2012.

[31] Presentation virtualization. Available at: http://www.virtualizationpractice.com/topics/presentation-virtualization/, Retrieved 14 July 2012.

[32] A quick guide to iscsi on linux. Available at: http://www.cuddletech.com/articles/iscsi/index.html, Retrieved 03 June 2012.

[33] Server virtualization with the xen hypervisor. Available at: http://www.xen.org/files/Marketing/WhatisXen.pdf, Retrieved 12 May 2012.

[34] Storage virtualization. Available at: https://en.wikipedia.org/wiki/Storage_virtualization, Retrieved 14 May 2012.

[35] Thin provisioning storage, challenges & opportunities - wikibon. Available at: http://wikibon.org/wiki/v/Thin_provisioning, Retrieved 08 July 2012.

[36] Timeline of virtualization development. Available at: http://en.wikipedia.org/wiki/Timeline_of_virtualization_development, Retrieved 30 April 2012.

[37] tune2fs(8) - linux man page. Available at: http://linux.die.net/man/8/tune2fs, Retrieved 16 June 2012.

[38] Ulteo open virtual desktop v3.0 easy installation. Available at: http://doc.ulteo.com/3.0/Easy_Installation.pdf, Retrieved 14 July 2012.

[39] Understanding lustre. Available at: http://wiki.lustre.org/manual/LustreManual20_HTML/UnderstandingLustre.html, Retrieved 03 May 2012.

[40] Virtual machine - wikipedia entry. Available at: http://en.wikipedia.org/wiki/Virtual_machine#Process_virtual_machines, Retrieved 13 May 2012.

[41] Vm autostart option. Available at: http://forums.citrix.com/message.jspa?messageID=1558923#1558923, Retrieved 22 June 2012.

[42] Xcp - manuals and documentation. Available at: http://wiki.xen.org/wiki/Category:Manual, Retrieved 09 June 2012.

[43] Xen - xcp download. Available at: http://xen.org/download/xcp/index.html, Retrieved 26 May 2012.

[44] Xen cloud platform project. Available at: http://xen.org/products/cloudxen.html, Retrieved 09 June 2012.

[45] Xen overview. Available at: http://wiki.xen.org/wiki/Xen_Overview, Retrieved 20 May 2012.

[46] Xencenter. Available at: http://wiki.xen.org/wiki/XCP_toolstack_on_a_Debian-based_distribution, Retrieved 14 June 2012.

[47] Xencenter. Available at: http://community.citrix.com/display/xs/XenCenter, Retrieved 14 June 2012.

[48] Xencenter. Available at: http://www.citrix.com/xenserver/download, Retrieved 14 June 2012.

[49] Xenserver 6.0 virtual machine installation guide. Available at: http://support.citrix.com/article/CTX130422, Retrieved 07 June 2012.

[50] Xenserver tools. Available at: http://docs.vmd.citrix.com/XenServer/6.0.0/1.0/en_gb/guest.html#windows_pvdrivers, Retrieved 09 June 2012.

[51] Xenwebmanager sourceforge project. Available at: http://sourceforge.net/projects/xenwebmanager/, Retrieved 27 June 2012.

[52] Bin Fan, Wittawat Tantisiriroj, et al. Diskreduce: Raid for data-intensive scalable computing. page 4, November 2009.

[53] Brian Kernighan and Rob Pike. The unix programming environment. page 35, 1984.

[54] Carl Purvis and Morgan Marquis-Boire. Access over ethernet: Insecurities in aoe (whitepaper). Available at: http://www.security-assessment.com/files/documents/whitepapers/Access%20over%20Ethernet%20-%20Insecurities%20in%20AoE.pdf, Written: 21 Aug 2006, Retrieved 03 November 2012.

[55] Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. page 2, November 2006.

1 CPU Flags For Determining The Virtualization Support

An example of displaying the CPU flags for determining the virtualization support on Linux/Unix operating system.

```
user@desktop ~ $ cat /proc/cpuinfo | grep flags
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
 pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb
 rdtscp lm 3dnowext 3dnow constant_tsc rep_good nopl nonstop_tsc extd_apicid
 pni monitor cx16 popcnt lahf_lm cmp_legacy svm extapic cr8_legacy abm sse4a
 misalignsse 3dnowprefetch osvw ibs skinit wdt npt lbrv svm_lock nrip_save
```

There is an AMD-V compatible processor model can be seen in the example above with the *svm* flag. The second example shows CPU flags of an Intel processor model with the *vmx* flag.

```
user@xcpserver67 $ cat /proc/cpuinfo | grep flags
 flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
  pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp
  lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
  pni   pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid
  sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx lahf_lm ida arat epb
  xsaveopt pln pts dts tpr_shadow vnmi flexpriority ept vpid
```

2 VM Parameter List Before And After Installing Xenserver Tools

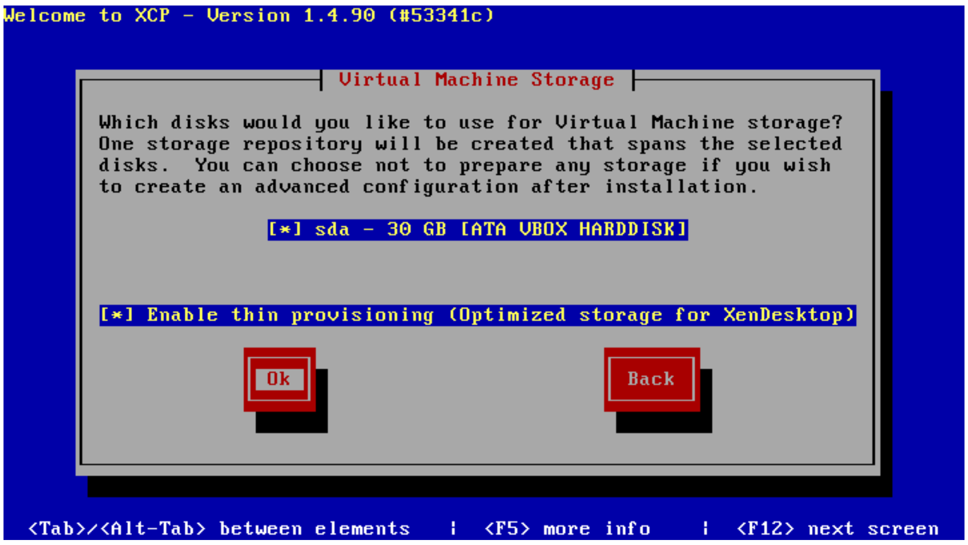
Virtual machine parameter list before installing Xenserver tools:

```
[root@xcpserver66 ~]# xe vm-param-list uuid=1e16e81e-3872-4fb2-3f1e-64c1dcd6919f
uuid ( RO)                         : 1e16e81e-3872-4fb2-3f1e-64c1dcd6919f
                    name-label ( RW): Linux Mint 13 MATE-virtual01

...lines removed................................................

            allowed-operations (SRO): changing_dynamic_range; changing_VCPUs_live;
                                      hard_reboot; hard_shutdown; clean_reboot;
                                      clean_shutdown; pause; snapshot

...lines removed................................................

           allowed-VBD-devices (SRO): 2; 3
           allowed-VIF-devices (SRO): 0; 2

...lines removed................................................

                    os-version (MRO): <not in database>
            PV-drivers-version (MRO): <not in database>
         PV-drivers-up-to-date ( RO): <not in database>
                        memory (MRO): <not in database>
                         disks (MRO): <not in database>
                      networks (MRO): <not in database>
                         other (MRO): <not in database>
                          live ( RO): <not in database>
    guest-metrics-last-updated ( RO): <not in database>

...lines removed................................................
```

Virtual machine parameter list after installing Xenserver tools:

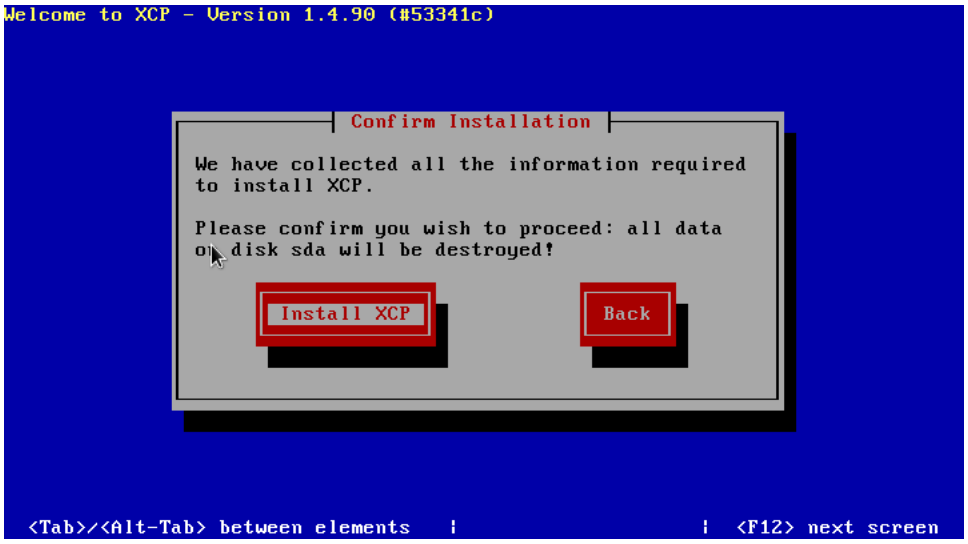
```
[root@xcpserver66 opt]# xe vm-param-list uuid=1e16e81e-3872-4fb2-3f1e-64c1dcd6919f
uuid ( RO)                          : 1e16e81e-3872-4fb2-3f1e-64c1dcd6919f
                    name-label ( RW): Linux Mint 13 MATE-virtual01

...lines removed................................................

            allowed-operations (SRO): changing_dynamic_range; pool_migrate;
                                      changing_VCPUs_live; suspend; hard_reboot;
                                      hard_shutdown; clean_reboot;
                                      clean_shutdown; pause; checkpoint; snapshot

...lines removed................................................

           allowed-VBD-devices (SRO): 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; 14; 15
           allowed-VIF-devices (SRO): 0; 2; 5; 6

...lines removed................................................

                    os-version (MRO): name: Linux Mint 13 Maya;
                                      uname: 3.2.0-23-generic; distro: linuxmint;
                                      major: 13
            PV-drivers-version (MRO): major: 1; minor: 4; micro: 90; build: 53341
         PV-drivers-up-to-date ( RO): true
                        memory (MRO):
                         disks (MRO):
                      networks (MRO): 0/ip: 172.16.128.201
                         other (MRO): platform-feature-multiprocessor-suspend: 1;
                                      feature-balloon: 1
                          live ( RO): true
    guest-metrics-last-updated ( RO): 20120607T13:42:36Z

...lines removed................................................
```

3 Screenshots From Xcp Installation



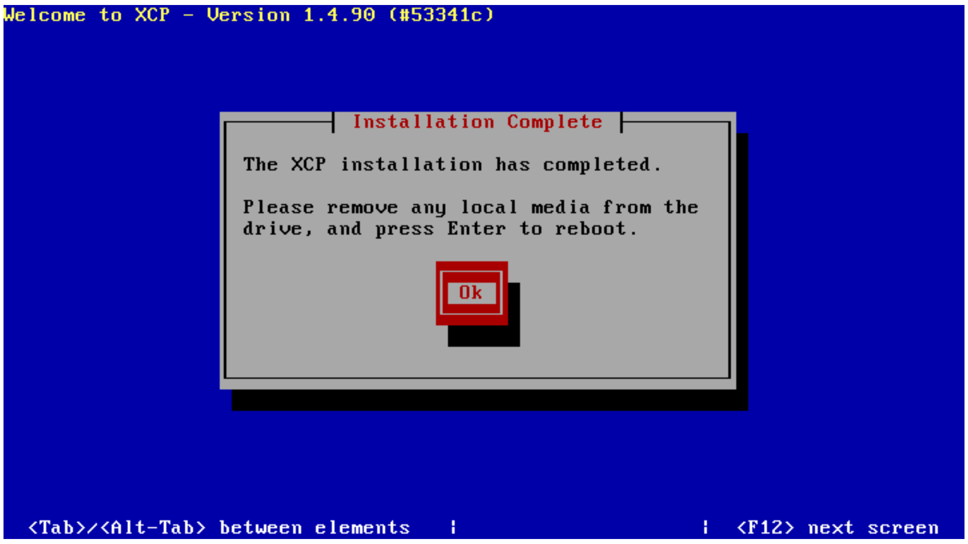
```
Welcome to XCP - Version 1.4.90 (#53341c)

                    Virtual Machine Storage

Which disks would you like to use for Virtual Machine storage?
One storage repository will be created that spans the selected
disks.  You can choose not to prepare any storage if you wish
to create an advanced configuration after installation.

              [*] sda - 30 GB [ATA VBOX HARDDISK]

[*] Enable thin provisioning (Optimized storage for XenDesktop)

            Ok                              Back

<Tab>/<Alt-Tab> between elements  |  <F5> more info  |  <F12> next screen
```

FIGURE A3.1: XCP installation screenshots, select destination



```
Welcome to XCP - Version 1.4.90 (#53341c)

                    Confirm Installation

We have collected all the information required
to install XCP.

Please confirm you wish to proceed: all data
on disk sda will be destroyed!

      Install XCP                      Back

<Tab>/<Alt-Tab> between elements  |                   |  <F12> next screen
```

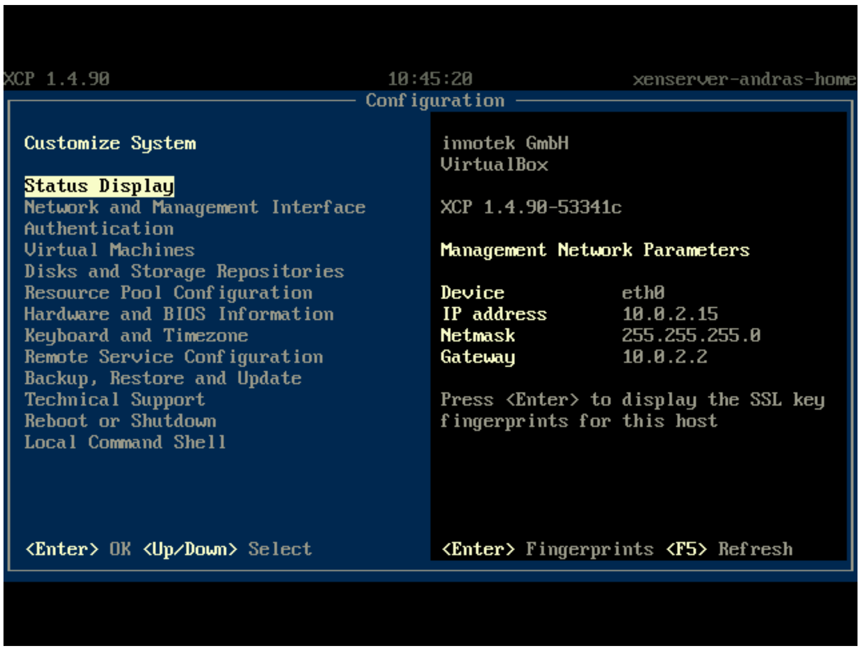
FIGURE A3.2: XCP installation screenshots, confirm and start installation



```
Welcome to XCP - Version 1.4.90 (#53341c)

                    Installation Complete

          The XCP installation has completed.

          Please remove any local media from the
          drive, and press Enter to reboot.

                           Ok

 <Tab>/<Alt-Tab> between elements   |            |  <F12> next screen
```

FIGURE A3.3: XCP installation screenshots, installation completed

FIGURE A3.4: XCP installation screenshots, XCP starting up



```
XCP 1.4.90                    10:45:20              xenserver-andras-home
─────────────────────────── Configuration ───────────────────────────────

 Customize System                    innotek GmbH
                                     VirtualBox
 Status Display
 Network and Management Interface    XCP 1.4.90-53341c
 Authentication
 Virtual Machines                    Management Network Parameters
 Disks and Storage Repositories
 Resource Pool Configuration         Device           eth0
 Hardware and BIOS Information       IP address       10.0.2.15
 Keyboard and Timezone               Netmask          255.255.255.0
 Remote Service Configuration        Gateway          10.0.2.2
 Backup, Restore and Update
 Technical Support                   Press <Enter> to display the SSL key
 Reboot or Shutdown                  fingerprints for this host
 Local Command Shell


 <Enter> OK <Up/Down> Select         <Enter> Fingerprints <F5> Refresh
```

FIGURE A3.5: XCP installation screenshots, XCP console status display

4 Screenshots From Ulteo Desktop Virtualization Demo Appliance

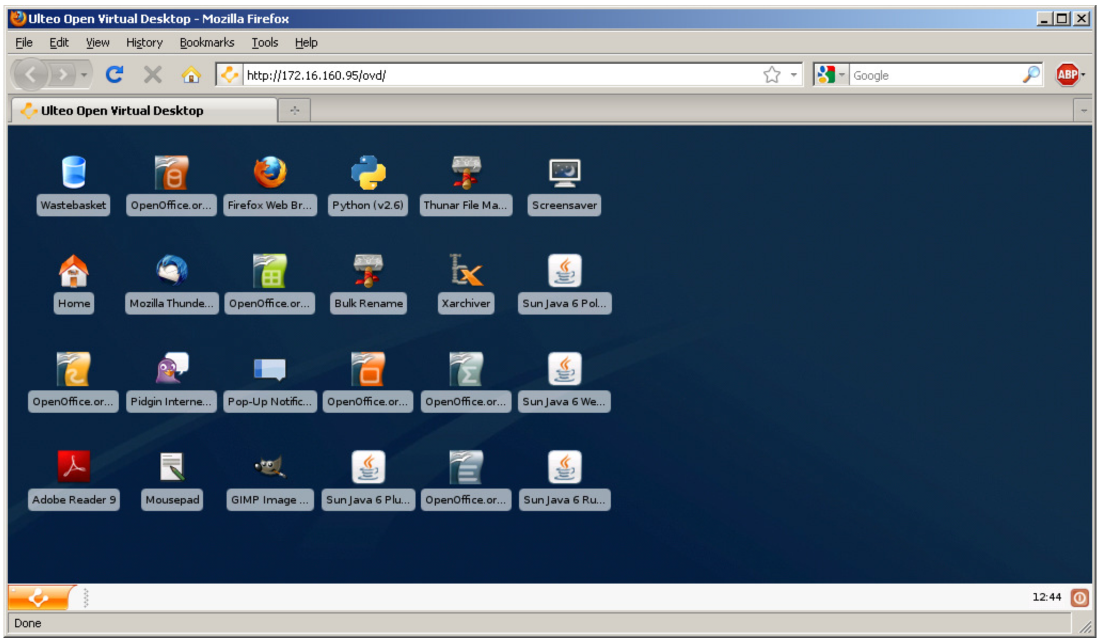


FIGURE A4.1: Ulteo virtual desktop from web browser after login

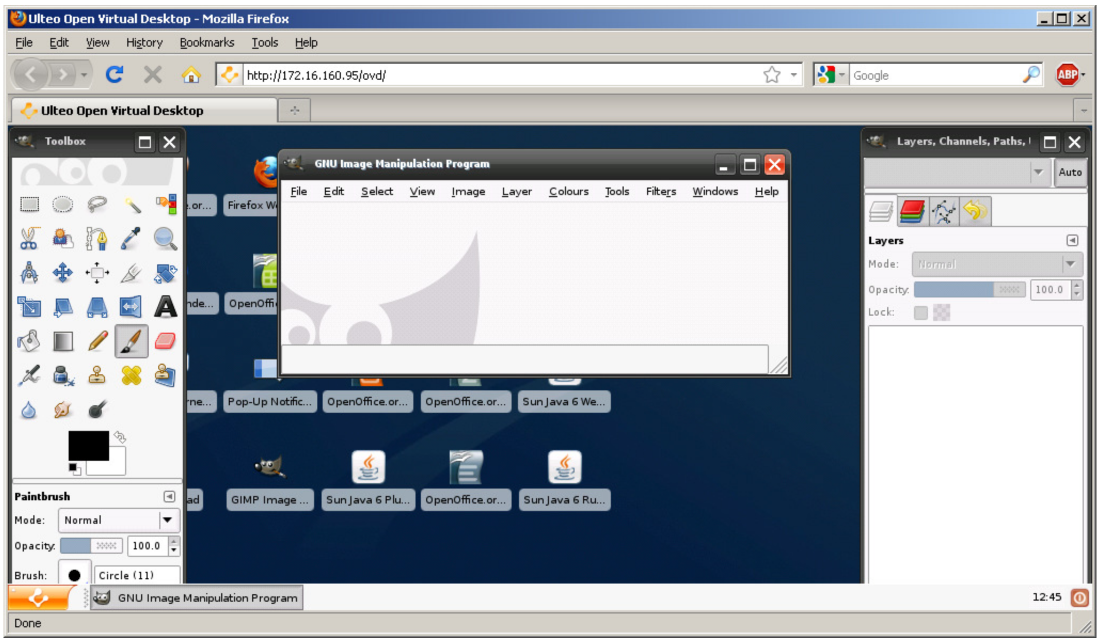


FIGURE A4.2: Gimp application running on virtual desktop within web browser window

5 Scripts And Commands For The Compilation Tests

Compilation loop for performance test:

```
#!/bin/bash
##  Warning. This may take 3-5 days to run !!!

for i in 100..600
do

   make clean ;
   /usr/bin/time --output=distcc_build$i.txt -f "%e real,%U user,%S sys"
   make -j35 CC=distcc

done
```

Performance and workload masurement commands

```
CPU and Disk

  iostat  -cd 2 200 > distcc_iostat.out.txt


Ethernet

  sar -n DEV 2 200 > distcc_sar-out.txt
```

Output file processing one-liners

```
/dev/sda device statistics

  awk '/sda/ print $2'  distcc_iostat.out.txt

CPU statistics

  awk '/^ / print $1'  distcc_iostat.out.txt

Ethernet statistics
```

```
  awk '/eth1/ print $3'  distcc_sar-out.txt
```

Example output of iostat

```
user@desktop ~ $ iostat  -cd 2 200
Linux 3.0.0-13-generic (desktop)  09/03/2012  _x86_64_ (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           8.30    0.00    2.27    0.03    0.00   89.40

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               4.72        63.67       169.36     941647    2504716
```

Example output of sar

```
user@desktop ~ $ sar -n DEV 2 200
Linux 3.0.0-13-generic (desktop)  09/03/2012  _x86_64_ (4 CPU)

04:40:27 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s
04:40:29 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00
04:40:29 PM      eth0      0.00      0.00      0.00      0.00      0.00      0.00

04:40:29 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s
04:40:31 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00
04:40:31 PM      eth0      0.00      0.00      0.00      0.00      0.00      0.00
```

6 Scripts And Commands For The Distributed Storage Tests

Distributed storage performance measurement script which generates directory structure:

```
#!/bin/bash
#  This script creates a two-level directory structure
#  and creates a file in each of subdirectory.

DIRECTORY=/tmp/1/

for i in 100..999
  do
     mkdir $DIRECTORY$i

     for ii in 100..999
       do
         mkdir $DIRECTORY/$i/$ii
         touch $DIRECTORY/$i/$ii/somefile$i$ii
       done
  done
```

Distributed storage performance measurement script which reads all file in directory structure:

```
#!/bin/bash
# This script reads all files in directory structure
# using "cat" command.

DIRECTORY=/dist_gluster_volume/1/

for i in 100..999
  do
     for ii in 100..999
       do
         cat $DIRECTORY/$i/$ii/somefile$i$ii > /dev/null
       done
  done
```

Find file command example:

```
user@desktop:/tmp/1$ time find . -name somefile666999
```

Local disk types and parameters:

```
user@testpc73:/tmp/1$ sudo hdparm -I /dev/sda
[sudo] password for user:

/dev/sda:

ATA device, with non-removable media
Model Number:       ST3500413AS
Serial Number:      ****Serial****
Firmware Revision:  JC49
Transport:          Serial, SATA Rev 3.0
Standards:
Used: unknown (minor revision code 0x0029)
Supported: 8 7 6 5
Likely used: 8
Configuration:
Logical max current
cylinders 16383 16383
heads 16 16
sectors/track 63 63
--
CHS current addressable sectors:   16514064
LBA    user addressable sectors:  268435455
LBA48  user addressable sectors:  976773168
Logical/Physical Sector size:           512 bytes
device size with M = 1024*1024:      476940 MBytes
device size with M = 1000*1000:      500107 MBytes (500 GB)
cache/buffer size  = 16384 KBytes
Nominal Media Rotation Rate: 7200
Capabilities:
LBA, IORDY(can be disabled)
Queue depth: 32
Standby timer values: specified by Standard, no device specific minimum
R/W multiple sector transfer: Max = 16 Current = 16
Recommended acoustic management value: 208, current value: 0
DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
     Cycle time: min=120ns recommended=120ns
PIO: pio0 pio1 pio2 pio3 pio4
     Cycle time: no flow control=120ns  IORDY flow control=120ns
Commands/features:
Enabled Supported:
   * SMART feature set
     Security Mode feature set
   * Power Management feature set
   * Write cache
   * Look-ahead
   * Host Protected Area feature set
   * WRITE_BUFFER command
   * READ_BUFFER command
   * DOWNLOAD_MICROCODE
     SET_MAX security extension
     Automatic Acoustic Management feature set
   * 48-bit Address feature set
   * Device Configuration Overlay feature set
   * Mandatory FLUSH_CACHE
   * FLUSH_CACHE_EXT
   * SMART error logging
   * SMART self-test
   * General Purpose Logging feature set
   * WRITE_DMA|MULTIPLE_FUA_EXT
   * 64-bit World wide name
     Write-Read-Verify feature set
   * WRITE_UNCORRECTABLE_EXT command
   * READ,WRITE_DMA_EXT_GPL commands
   * Segmented DOWNLOAD_MICROCODE
   * Gen1 signaling speed (1.5Gb/s)
   * Gen2 signaling speed (3.0Gb/s)
   * unknown 76[3]
   * Native Command Queueing (NCQ)
   * Phy event counters
   * unknown 76[15]
     Device-initiated interface power management
   * Software settings preservation
   * SMART Command Transport (SCT) feature set
   * SCT Long Sector Access (AC1)
   * SCT LBA Segment Access (AC2)
   * SCT Error Recovery Control (AC3)
   * SCT Features Control (AC4)
   * SCT Data Tables (AC5)
     unknown 206[12] (vendor specific)
Security:
Master password revision code = 65534
supported
not enabled
not locked
frozen
not expired: security count
supported: enhanced erase
80min for SECURITY ERASE UNIT. 80min for ENHANCED SECURITY ERASE UNIT.
Logical Unit WWN Device Identifier: 5000c5003f84c6ec
NAA : 5
```

```
IEEE OUI : 000c50
Unique ID : 03f84c6ec
Checksum: correct




user@xcpserver67 /tmp/1 $ sudo hdparm -I /dev/sda
[sudo] password for user:

/dev/sda:

ATA device, with non-removable media
Model Number:       OCZ-VERTEX4
Serial Number:      OCZ-PVB****Serial****
Firmware Revision:  1.3
Transport:          Serial, ATA8-AST, SATA 1.0a, SATA II Extensions, SATA Rev 2.5,
                    SATA Rev 2.6, SATA Rev 3.0
Standards:
Supported: 9 8 7 6
Likely used: 9
Configuration:
Logical max current
cylinders 16383 0
heads 16 0
sectors/track 63 0
--
LBA    user addressable sectors:  250069680
LBA48  user addressable sectors:  250069680
Logical  Sector size:                   512 bytes
Physical Sector size:                   512 bytes
Logical Sector-0 offset:                  0 bytes
device size with M = 1024*1024:      122104 MBytes
device size with M = 1000*1000:      128035 MBytes (128 GB)
cache/buffer size  = unknown
Nominal Media Rotation Rate: Solid State Device
Capabilities:
LBA, IORDY(can be disabled)
Queue depth: 32
Standby timer values: specified by Standard, no device specific minimum
R/W multiple sector transfer: Max = 16 Current = 16
DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
     Cycle time: min=120ns recommended=120ns
PIO: pio0 pio1 pio2 pio3 pio4
     Cycle time: no flow control=120ns  IORDY flow control=120ns
Commands/features:
Enabled Supported:
   * SMART feature set
     Security Mode feature set
   * Power Management feature set
   * Write cache
```

```
     * WRITE_BUFFER command
     * READ_BUFFER command
     * NOP cmd
     * DOWNLOAD_MICROCODE
     * 48-bit Address feature set
     * Mandatory FLUSH_CACHE
     * General Purpose Logging feature set
     * WRITE_DMA|MULTIPLE_FUA_EXT
     * 64-bit World wide name
     * Write-Read-Verify feature set
     * WRITE_UNCORRECTABLE_EXT command
     * Gen1 signaling speed (1.5Gb/s)
     * Gen2 signaling speed (3.0Gb/s)
     * Gen3 signaling speed (6.0Gb/s)
     * Native Command Queueing (NCQ)
       Non-Zero buffer offsets in DMA Setup FIS
     * DMA Setup Auto-Activate optimization
     * In-order data delivery
     * DOWNLOAD MICROCODE DMA command
     * WRITE BUFFER DMA command
     * READ BUFFER DMA command
     * Data Set Management TRIM supported (limit 16 blocks)
Security:
Master password revision code = 65534
supported
not enabled
not locked
not frozen
not expired: security count
not supported: enhanced erase
20min for SECURITY ERASE UNIT. 400min for ENHANCED SECURITY ERASE UNIT.
Logical Unit WWN Device Identifier: 5e83a97a8a2c059a
NAA : 5
IEEE OUI : e83a97
Unique ID : a8a2c059a
Checksum: correct
```

List of Tables

List of Figures