TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VALAMBAL ARAVINDAN

PERFORMANCE ANALYSIS OF ISCSI BLOCK DEVICE IN VIRTUALIZED ENVIRONMENT

Master of Science Thesis

Examiner: prof. Evgeny Kucheryavy
Senior Researcher. Dmitri Moltchanov
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 3rd September 2014

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
VALAMBAL, ARAVINDAN: Performance analysis of an iSCSI block device in virtualized environment
Master of Science Thesis, 56 pages
October 2014
Major: Communication Engineering
Examiner(s): Prof. Evgeny Kucheryavy and Dmitri Moltchanov
Keywords: iSCSI block device, Storage area network, kernel virtual machine and quick emulator

Virtualization is new to telecom but it has been already implemented in IT sectors. Thus its benefits are already proven, which drags other sectors attention towards it. Now the telecom organizations are also focusing on virtualization to reap the full benefits of it. The main focus of this thesis is to conduct a performance analysis of a block storage device in a virtualization environment. Storage performance plays vital role in telecom sector. The performance and the reliability of the storage device is more important factor to fulfill the client request with minimum latency.

This thesis is comprised of three main areas. The first literature part is to study the different storage networking possibilities and the different storage protocol practice to establish communication between server and the storage in the storage area network. The study indicated that Internet Small Computer System Interface (iSCSI) has more advantages than other practices in the storage area network. The second part covers the design of storage area network (SAN) solution. The storage is offered by an iSCSI storage server. It offers a block level storage device access to the compute server. Different iSCSI targets are available in market, performance of those were compared. Linux-IO Target was concluded as better iSCSI target with better performance and reliability. The Storage server was implemented as a virtual machine for better resource utilization, thus there was a study about the hypervisor and the different networking options for the virtual machines were compared. The final part is to optimize the SAN solution. Multipathing, different caching options and different driver options provided by the kernel virtual machine (KVM)/ Quick emulators (QEMU) were considered for optimization.

PREFACE

This Master thesis was written at Product Design Unit, Ericsson at Herzogenrath, Germany during the period of May 2014 –October 2014.

This thesis is the integral part of my Master of Science program in Information Technology. It has been concluded and presented to the Faculty of Computing and Electrical Engineering at Tampere university of Technology in Finland for the graduation.

I am grateful to examiner Prof. Evgeny Kucheryavy and Dmitri Moltchanov of my university, for their valuable guidance which helps to keeping me on track during whole thesis period. And also I would like to thank all my professors and lecturer, who helped to acquire more insight in telecommunication by presenting their great lectures and guiding me in laboratory works in my Masters programme.

I extremely appreciate my supervisor Manuel Buil in Ericsson, for the encouragement and support provided by him throughout my thesis period and warmest thanks to my manager, who gave me this great opportunity. I would like to express my gratitude to all my colleagues for their technical support and advice.

I thank my friends and family for their support, encouragement and well wishes throughout this period. Finally, special thanks to my son Midhulan Aravindan for his love, support and patience.

Valambal Aravindan

Herzogenrath, October 2014

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ALUA	Asymmetric Logical Unit Assignment
API	Application Programming Interface
ATA	Advanced Technology Attachment
CIFS	Common Internet File System
CPU	Central Processing Unit
DAS	Direct-attached storage
ERL	Error Recovery Levels
eui	extended unique identifier
FC	Fibre Channel
FC SAN	Fibre Channel Storage Area Network
FCIP	Fibre channel internet protocol
FCoE	Fibre Channel over Ethernet
FCP	Fibre Channel Protocol
FC-PI	Fibre Channel Physical Interface
FDB	Forwarding Data Base
FIO	Flexible Input/Output
HBA	Host-Bus-Adapter
HDD	Hard Disk Drive
I/O	Input/Output
IaaS	Infrastructure as a service
IDE	Integrated Drive Electronics
IET	iSCSI Enterprise Target
IP SAN	Internet Protocol Storage Area Network
IP	Internet Protocol
iqn	iSCSI Qualified Name
iSCSI	Internet Small Computer System Interface
ISL	Inter-Switch Link
IT	Information technology
KVM	Kernel Virtual Machine
LAN	Local Area Network
LIO	Linux-IO Target
LU	Logical Unit
LUN	Logical Unit Number
MAC	Media Access Control
MC/S	Multiple connections per session
NAS	Network-attached storage
NFS	Network File System

NIC	Network Interface Card
PaaS	Platform as a service
PCI	Peripheral Component Interconnect
PDU	Protocol Data Unit
PR	Persistent Reservation
QEMU	Quick emulators
RAM	Random Access Memory
SaaS	Software as a service
SAN	Storage Area Network
SCSI	Small Computer System Interface
SCST	SCSI Target Subsystem
SDD	Solid-State Drive
SDN	Software-defined networking
SLA	Service Level Agreement
STGT	SCSI Target Framework
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VFS	Virtual File System
VM	Virtual machine
WAN	Wide Area Network

LIST OF FIGURES

1. INTRODUCTION

The cost is the crucial factor in today's business world. IT infrastructure is one of the important sectors where the organizations are spending more to achieve better performance and business continuity. The organizations are always keen to reduce the cost without sacrificing the performance, which leads them to win the business competitors. Cost and efficiency depends on each other. Efficient utilization of any resources reduces the cost. Virtualization technology provides a way to achieve efficient consumption of the resources, by virtualizing the physical hardware.

Most of the organizations are transforming there conventional datacenter to a virtualized datacenter to incur the profits from virtualization. The three main classification of on the IT infrastructure virtualization includes server, storage and network. This thesis focuses on the performance of the virtualized datacentres.Virtualization transforms the conventional data center into a more flexible datacenter through server virtualization and consolidation. It also simplifies the provisioning of IT resources. The resource consolidation reduces hardware cost. The possibility of dynamic virtual machine migration from servers to servers increases the flexibility significantly. Thus the organizations bring in transition from conventional to virtualization datacenters.

Data is one of the main assets of the company. Data storage has to be more secured and reliable to maintain the business continuity. SCSI is a widely employed storage protocol to establish connection and data transfer between compute and storage devices. But it cannot support for the long distance scenarios. To address this problem lot of technologies evolved to carry the SCSI commands between compute and the storage server. Most popular among them is iSCSI protocol. iSCSI protocol enables access between the compute and storage servers in the storage-area networks (SANs) over TCP/IP networks. Thus iSCSI protocol allows SCSI command sent over the underlying TCP/IP network. It is very popular, easy to implement and affordable, since it uses the already existing IP network rather than deploying any new infrastructure. It provides block-based high speed data transfer. But the organizations are running out of space due to increase in the application exploration. Virtualization reduces the growth rate of storage and other hardware demand by effective usage of the hardware resources.

In the conventional method the iSCSI storage server is a real machine, such that application run on the real hardware. The main limitation of this conventional method is scalability and inefficient utilization of the hardware resources. The solution is to virtualize the hardware. In this thesis work, iSCSI storage target server is a virtual machine, thus the storage devices are virtual device. The benefits include better utilization of resources, scalability, live migration etc. Since it is virtualized, its

performance is not similar to native performances. Several factors are influencing the performance of the virtual machine and the iSCSI target running on it. Thus the goal of this thesis is to study and analyze the factors influencing the performance of an iSCSI target device in the virtual scenario.

1.1 Structure of Thesis

This thesis comprises of six chapters. Chapter 1 presents the brief introduction and the goal of this thesis. Chapter 2 explains the virtualization and different types of virtualization. Chapter 3 offers the brief idea of storage area network (SAN) and network attached storage (NAS). And different protocols that connects compute and the storage in the storage area network are explained elaborately in chapter 3. Chapter 4 focus on the design and implementation of storage server in the storage area network, study of hypervisor KVM/QEMU, performance comparison of different iSCSI targets and virtual machine networking possibilities. Chapter 5 is about analyzing different possibilities to optimizing the storage performance. Finally Chapter 6 presents the conclusion of the thesis.

2. INTRODUCTION TO VIRTUALIZATION AND CLOUD

Virtualization and cloud computing technology go hand in hand to reap the maximum benefits of each other. In general Cloud refers to, on demand service. To leverage cloud, virtualization works in parallel with cloud in most environments.

2.1 Cloud computing

Evolution of internet leads to a technology called cloud computing. It means that all the resources needed for computing comes under a single cloud and it is provided to user as a service on demand. The resources can be either software or hardware. The computing process may run in one or many connected servers, with the help of virtualization technology. The cloud computing has the following service models:

Infrastructure as a service (IaaS): The whole infrastructure, required to run the organization operation is provided as a service. Some of the important infrastructure resources are physical and virtual machines, storage, virtual and physical network etc.

Platform as a service (PaaS): Platform is provided as service by the cloud providers which are ready to run user application on demand. The user does not care about the underlying operating system or the storage provision.

Software as a service (SaaS): Service provider provides their software or application as a service running in their data center to the remote user on demand through internet.

2.2 Need for cloud

There is no doubt that all the organization look forward to adopt a technology where reliable performance is achieved with reduced cost. The conventional IT infrastructure cannot help them anymore. Some of the vital problems the conventional system facing are underutilization of hardware, data centers are running out of space, IT infrastructure implementation and maintenance cost and energy cost are increasing rapidly. Virtualization addresses these problems. Different virtualization types are adopted to address these problems and to increase the overall efficiency and performance; those are discussed briefly in the following sections. The benefits of virtualization also grow alongside with the growth of virtualization. The significant benefits of virtualization includes business agility, operational flexibility, high availability, disaster recovery, consolidation of IT services, scale up and scale out on demand, self-service, pay only

for the service consumed, dynamic accessibility. The *Figure 1* shows the evolution of conventional system to virtualized cloud system.

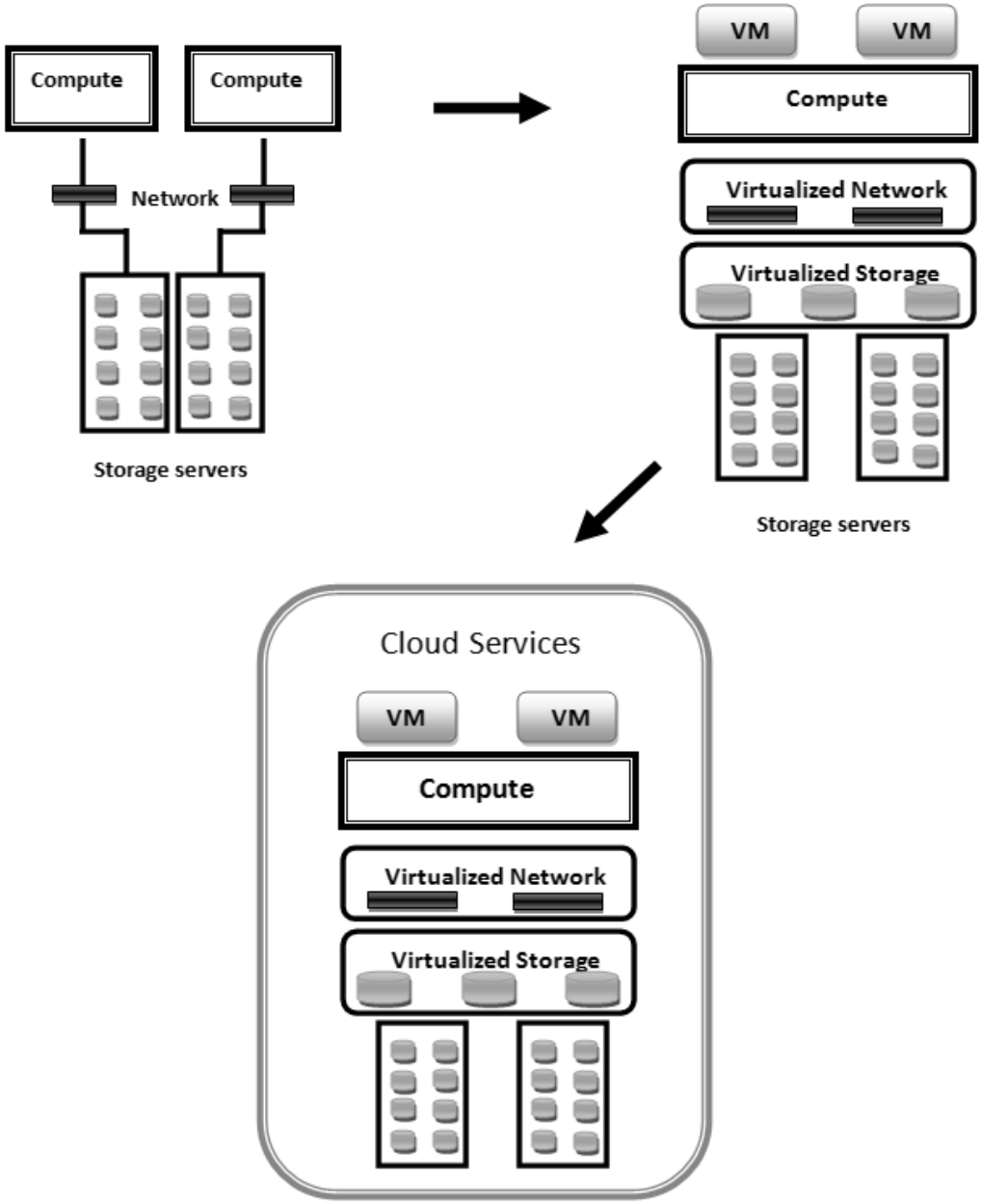


Figure 1 Evolution of conventional system to virtualized cloud system

2.3 Virtualization and its types

Virtualization is a technology which enables a way to create virtual resources out of physical resources and presenting it to the user rather than providing actual physical resources. Virtualization leads to pooling of physical resources that has already proven values like efficient utilization of resources, central point of resource management. Virtualization is implemented at different layers of IT infrastructure. They are classified as compute, network, storage and application layer virtualization of IT infrastructure.

2.3.1 Compute virtualization

It is the process of detaching the operating system from the physical layer. It is referred as compute virtualization, because in this layer together with other hardware components, the most important component i.e. CPUs are virtualized. CPUs are the components, where the computing process like running and executing the codes are being done. The abstraction of operating system from the physical layer is achieved by virtualization software called hypervisor which comes in between operating system and physical resources. This paves the way for running multiple operating systems on the same physical resources such that multiple virtual machines can run parallel on same physical machine. There are two flavors of hypervisor available in the commercial market: native and hosted.

Native hypervisor: It is running on the bare metal. It manages the physical resources, since it has direct access to it.

Hosted hypervisor: It is running on the host operating system, so it relies on the host operating for the physical resources management.

Hardware virtualization is applied in three different modes:

Full virtualization: Hypervisor completely simulate the underlying hardware. And the guest operating system does not need any modification. The operating system instructions are binary translated so the virtualized hardware can understand and serve for it. Hypervisor decouples the VMs and the underlying hardware. The guest operating systems are not aware of being virtualized. Hardware features like CPU, memory etc. required by VMs to run their operating system and applications are served by the hypervisor. The greatest advantage of this type is that the virtualized hardware architecture can be completely different from the host architecture. However the biggest disadvantage is the processing speed.

Para-virtualization: Hypervisor does not simulate the hardware. It just acts as software API in-between VM and the underlying hardware. Hypervisor resides on the hardware and guest operating system run on top of this hypervisor. The guest operating system instructions are bypassed to the real physical hardware. To achieve this para-virtualization device drivers are required in both guest and the hypervisor. Thus this enables the possibility of running the instruction in real physical hardware which is very faster than running in the emulated hardware. But here the guest operating system needs some modification in order to communicate with the drivers. And therefore guest operating systems are aware of being virtualized. This type of virtualization is can be utilized when the hardware is not supported for full virtualization or application in need of high processing speed.

Partial virtualization: In this method operating environment is virtualized, rather than the complete machine such that the address spaces are virtualized so that can be utilized by VM's. It provides isolation of VM from the host, by running VM in separate domain. It allows the VM user to install software system, upgrade system libraries in the guest without affecting those in the host, and vice versa. Thus, rather than emulating

physical hardware, operating system virtualization emulates a complete operating system user space. It enables sharing computer resources among multiple users but the on the other hand the compatibility is not good. If certain hardware features are not virtualized, then any software using those features will fail.

Benefits of compute virtualization:

- Effective utilization of hardware.
- It enables virtual machine to run in isolated environment like a physical machine.
- Virtual machines are hardware independent; migration is easy between different servers.
- Hardware investment cost is reduced massively.

2.3.2 Storage virtualization

It is the process of hiding the underlying complexity of physical storage resources and presenting them as a virtual storage to the compute systems. This is attained with the help of hypervisor. The compute system is not aware of the storage virtualization; it uses a virtual disk as if it was a physical storage disk attached to them. The virtual storage is mapped to corresponding physical storage; this operation is taken care by virtualization layer. Storage virtualization deals with storage provisioning to VMs, block and file level virtualization, virtual provisioning and automated storage tiering. Storage stack can be classified as compute, storage and network.

Storage provisioning: Storage provisioning is the process of providing storage space to servers, virtual machines or any other computing device and it is deployed in compute layer. One way is the hypervisor apply file system on the physical storage attached to it and it creates files in the physical storage space. These files are provided to the VMs by the hypervisor as a virtual disk. Virtual machines view this storage space as a real physical disk attached to it. Size of the file depends on the storage needs of the VMs. Instead of providing a single full disk to a compute system, many virtual disks are made out of it and it is attached to different virtual compute systems. Another way is the VM can store data directly on a LUN in the real physical storage system instead of storing its data in a virtual disk; this method is called as raw device mapping. ***LUN*** is a logical unit number used in storage to identify a logical unit of as storage device addressed by the protocols which encapsulate SCSI. This method of storing is useful when there is a requirement that the applications running on the VM should know about the physical characteristics of the storage device.

Block and file level virtualization: This virtualization deployed in the network layer and provides an abstract view of physical storage resources. The I/O coming from the compute system is sent to the physical storage through the virtualization layer at the network layer. The virtualization software hides the physical location of the storage hardware and presents the only the logical location of it to VMs. The virtualization

software needs special networking intelligence combine storage at different location. The logical storage is mapped to the physical storage device. Virtualization enables to pool multi-vendor storage resources. This can be implemented in both SAN and NAS environments. Virtualization is applied at the block level in SAN and it is applied at the file level in NAS. Block level storage virtualization allow us to combine one or more LUNs from different storage arrays to form a single virtual big volume based on the requirements and this volume is presented to the VMs. File-level storage virtualization eliminates dependencies between the file and its physical location, it enables users to use logical path than the physical path.

Virtual provisioning and automated storage tiering: Virtual provisioning is based on thin provisioning; it is the ability of allocate a LUN to the VM with high capacity rather than the actual capacity of the LUN. It is deployed in storage layer. Storage tiering is said to be hierarchical storage management. Storage tiering is the way of storing data into a different categories of storage arrays. Categories are based on the SLAs with different customers. The SLA has different set of requirements such as performance, frequency of use, reliability etc...

Benefits of storage virtualization:

- Data can be migrated between the storage disks without any interruption.
- Storage space can be scale in or scale out depends on the demand.
- Effective utilization of storage.
- Easy management, since the storage is pooled.
- It provides different storage provisioning options to provide storage to VM.
- Different networking options for I/O between compute and storage device.
- Virtual provisioning and storage tiering optimizes the utilization of storage infrastructure.

2.3.3 Network virtualization

It is the process of creating multiple logical networks on top of the underlying physical network and operates them as separate independent physical networks. Network virtualization enables resources sharing among the virtual networks. Servers in the virtual networks can communicate without routing frames, even if they are in different physical networks. This enables grouping of server regardless geographic location.

Network virtualization is carried out at two levels as virtualizing the physical networks and virtual machine networks. The physical network has components such as routers, switches, bridges, repeaters and hubs. The physical network enables communication between all the physical devices attached to the network with networking capabilities. The virtual network resides inside the server, which enables the communication between different VMs connected to the virtual network. And also enable communication between the VM and the physical network. Thus the VM can

communicate with any other devices connected to the physical network. The VM network has component such as virtual NIC, HBA and virtual switch.

Benefits of network virtualization:

- It provides sharing of the network resources.
- It improves the network security by restricting the communication between the VMs in different virtual network.
- Depends on the organization requirements the servers can be grouped together logically.
- Logical grouping of the servers provides easy management.

2.3.4 Desktop virtualization

This is the process of abstracting the desktop from operating system and the application from the endpoint user. By this technology operating system is centralized and running in a VM in the datacenter. The end users can connect to the desktop VM via LAN or WAN. So executing the application and data storage are happens in the data center not in the end-user desktop.

Benefits of Desktop virtualization:

- It reduces cost of the organization, by replacing the personal computer system with thin clients.
- It improves data security, since organization's data is stored in their data center.
- Data storage backup is simplifies.
- It can be accessed by the employees regardless of their location.

2.3.5 Application virtualization

It is the process of abstracting the application from the underlying operating system. Application software creates the opportunity for the user to use the application without any concern of what underlying operating system they have. Virtualized Application run in an isolated environment.

Benefits of Application virtualization:

- Application deployment is easy.
- Application installation does not cause any changes to the operating system and the file system.
- It prevents any corruption of operating system during installation and easy operating system management.

3. STORAGE NETWORKING TECHNOLOGIES

Earlier the communication between the compute and the storage was achieved through the fixed channel. It means a constant channel or a wired connection is established between the compute and storage. The storage is directly attached to the compute system and the ways of communication is described in the operating system. Thus the host system retains the knowledge of all the storage devices attached to it. The storage may be external or internal; the communication channel remains static. The intermediate hops are very much limited. This fixed connection cannot be used by the other compute nodes to communicate with storage. Laying down a connection separately for each compute node and storage makes this technology very expensive. So it is utilized for communication over shorter distances. PCI, IDE/ATA, SCSI are some of the popular storage protocol used in fixed channel technology. Among these, SCSI provides better performance over PCI, IDE/ATA. As the channel technology is inefficient, there is a necessity to find a new methodology. The outcome is the IP networking technology; IP networks are also used to transfer storage I/O traffic between compute and the storage.

The IP network connects the compute nodes to share data among them. The same network can be utilized to transfer storage I/O traffic between compute and the storage. Thus it reduces the infrastructure cost very much. A network technology is more flexible than channel technology. The data transmission path between the transmitting node and the receiving node are not static. It changes dynamically, depends on the criteria like resource availability, current traffic situation etc. Each node in the network is identified by unique address. The data packets from the transmitting node are routed via in-between hops to each reach the final destination. Hence the networking technology supports longer distance topologies. Channel and network technologies employ networking components such as switches, routers, cables, buses, ports etc., and protocols for communication. iSCSI, FCOE, FCIP, and NFS are some of the important storage protocols used in IP networking technology. There are two ways of providing storage to the compute node using the network, they are

- Presenting storage as a block device (Block level access)
- Presenting storage as a file system (File level access)

The shows *Figure 2* the logical picture of storage area network (SAN) which provides block level access and the network attached storage (NAS) which provides file level storage access.

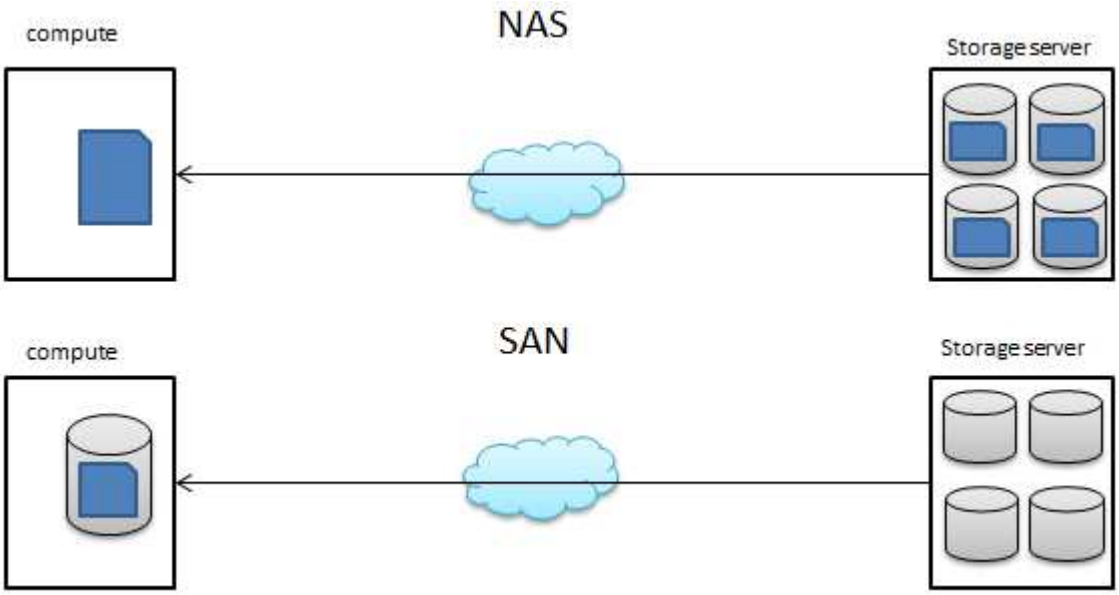


Figure 2 Logical representation of SAN and NAS

3.1 Storage area network (SAN)

Storage area network is a dedicated high performance network, whose primary role is to enable the communication between the compute systems and storage servers. SAN defines the entire hardware and software infrastructure that allows compute nodes to access storage that is not directly attached to it. Servers share the storage subsystem. SAN enables the capability of accessing the external storage as a block device. When there is a storage request from the compute, the storage servers allocate a block device in requested size in the storage array and it presents that to the compute. Then the compute system has to create a file system in the block for the further use. By this way block level access of the external storage is achieved by the SAN. Since the network is dedicated, communication between storage and servers are in high speed, thus increasing the overall performance. SAN could also provide highly secured data storage. There are different networking methodologies and protocols to establish a communication between storage and compute in SAN. They are explained below with more focus on iSCSI SAN.

Different types of SAN technologies are.

- Fibre channel SAN
- IP SAN
- FCoE

The SCSI is the underlying protocol which is base for all of the above mentioned technology to provide block storage service to the compute. At first SCSI protocol is introduced below and then the above mentioned SAN technologies are discussed. The

Figure 3 shows the different protocols used in SAN and protocols layering with respect to SCSI.

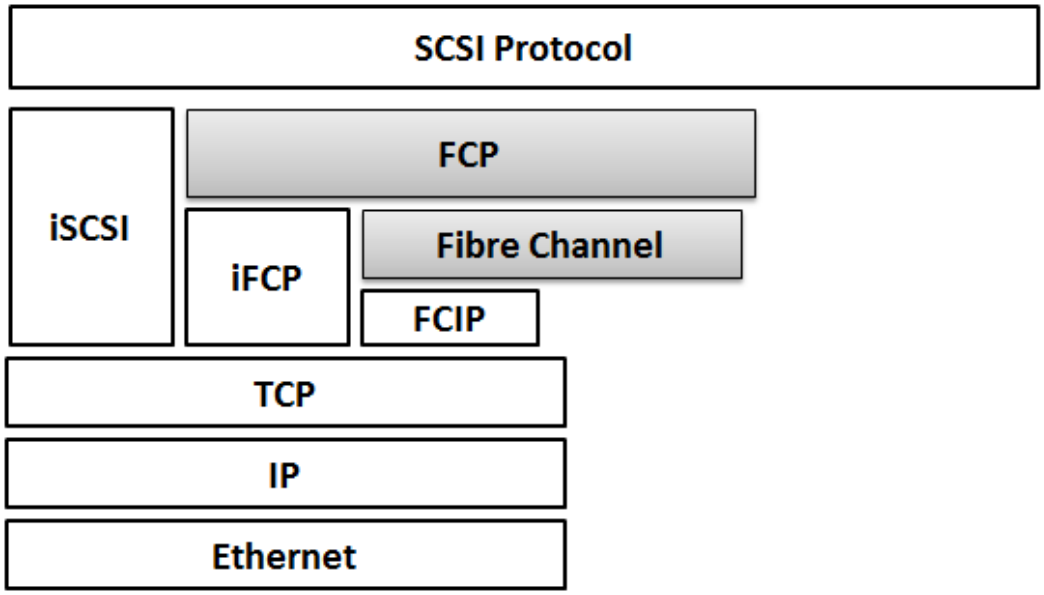


Figure 3 SAN protocols

3.1.1 Small computer system interface

SCSI (small computer system interface) is a protocol used for communication between server and storage. The SCSI standard defines an interface, protocols and commands to establish communication between an initiator and the target. Initiator is the server and target is the storage device. It provides a half-duplex communication path for SCSI commands and data. Interfaces such as cables, connectors, optical signals etc. that allow initiator and targets to communicate. SCSI commands are encapsulated and carried across the networks.

SCSI is a client server protocol. The client is called initiator which sends the request to the target server. The storage server is called target which serves for the initiator request. A single target can have multiple application clients. The target device is attached to the SCSI bus of the target. Up to 16 devices can be attached to a single bus. Host bus adopter is a mandatory device, it takes single slot. A SCSI host adapter is a device used to connect one or more SCSI devices to a computer bus. Thus host bus adapter is connected to the SCSI bus on one side, and to the host computer bus on the other side. After reserving one slot for host bus adopter, 15 devices can attach to one SCSI bus.

I/O operation of SCSI between target and initiator:

It has two categories of protocol services

- Execute command/confirmation services.
- Data transfer services.

This operation between the client and server is represented as a block diagram in *Figure 4*. The Initiator starts the operation and selects a target. The target accepts it and requests a command from the Initiator. The Initiator responds by sending the command descriptor block to be executed.

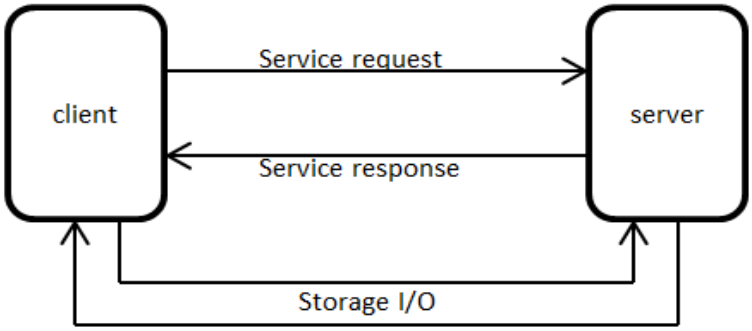


Figure 4 SCSI client-server operations

This leads to three main phases of I/O operation

1. Command execute: Send required command and parameters via CDB (command description Block).
2. Data: Transfer data in accordance with the command
3. Confirmation: Receive confirmation of command.

Targets have one task manager and one or more Logical Units (LU), which are numbered as LUN. A task manager is a server within the SCSI target device that processes task management functions. Logical unit contains a device server to process SCSI commands. It represented in *Figure 5*.

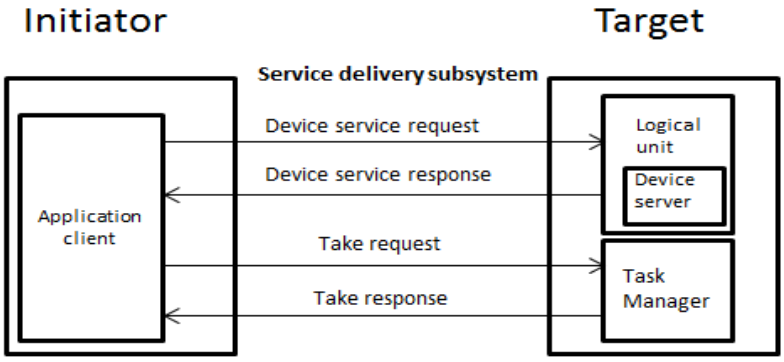


Figure 5 SCSI target

With the knowledge of SCSI , now the focus has been moved on to different SAN technologies as mentioned above in 3.1.1.

3.1.2 Fibre channel SAN

Fibre channel SAN is a very common high-speed, dedicated network between compute systems and shared storage devices. FC SAN uses SCSI over fibre channel protocol (FCP) to transfer data between compute systems and storage devices. FCP is a transport

protocol similar to TCP used in IP networks. Fibre Channel does not incorporate a command set as SCSI, but it does provide a mechanism to lay over other protocols onto Fibre Channel. SCSI generally rides on top of Fibre Channel. It provides block-level access to storage devices; block I/O is over FC. FC SAN enables storage consolidation and allows a storage system to be shared by multiple compute systems. This improves utilization of storage resources, compared to DAS architecture. The basic topology of FC SAN is represented in *Figure 6*.

Layers of FCP protocol:

FC4 – Protocol-mapping layer, in which application protocols such as SCSI is encapsulated into a PDU for delivery to FC2.

FC3 – Common services layer, a thin layer that could eventually implement functions like encryption or RAID redundancy algorithms.

FC2 – Network layer, defined by the FC-PI-2 standard, consists of the core of fibre channel, and defines the main protocols; this layer contains the basic rules for sending the data across the FC network.

FC1 – Data link layer, which implements line coding of signals; this layer defines the transmission protocol that includes serial encoding and decoding rules, special characters used, and error control.

FC0 – Physical layer, this is the lowest layer in the FCP stack. This layer defines the physical link between the systems, including the fibre cables, connectors etc.

Each device in FC SAN are called nodes, each of them has at least one port to communicate each with other. Ports are the outlets in the FC network. Some of the FC ports are mentioned below.

N_port: An end point in the switched fabric. This port is also known as the node port. Typically, it is a compute system port (HBA) or a storage array port that is connected to a fibre channel switches.

E_port: It is the connection between two fibre channel switches. It is also known as an Expansion port. When E_ports between two switches form a link, that link is referred to as an inter-switch link (ISL).

N_port: A port on a switch that connects an N_port. It is also known as a fabric port.

G_port: A generic port that can operate as an E_port or an F_port and determines its functionality automatically during initialization.

NL_port: It is a port on the node used with an FC-arbitrated loop topology. It is also known as node loop port.

All nodes in the FC network communicate with one another through an FC switch or multiple interconnected FC switches. Switches are interconnected through fibre channel. A link is established between two switches, that link is referred as inter-switch link (ISL).

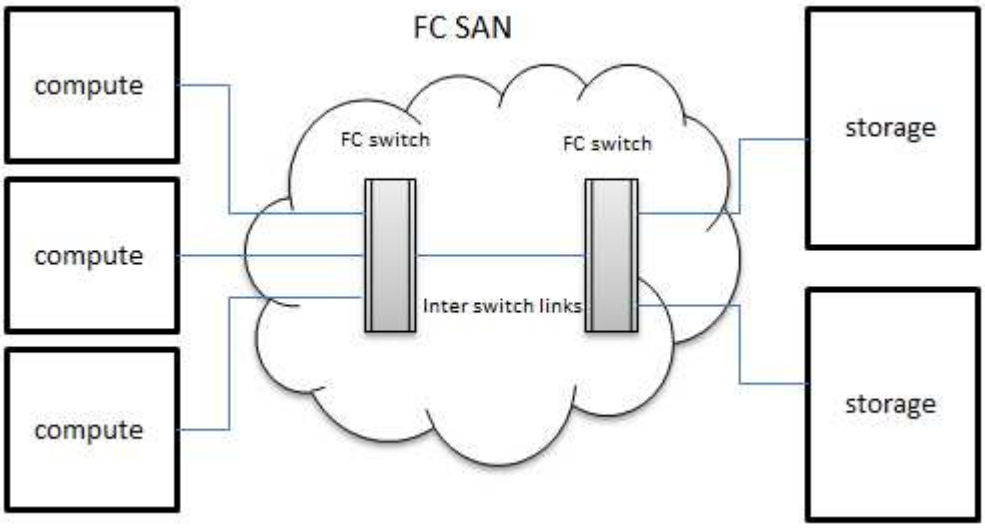


Figure 6 Basic topology of FC SAN

The different types of FC SAN topologies are discussed below.

Point to point: Two devices are connected directly to each other. Such that dedicated connection between N_port of the two devices.

Arbitrated loop: All devices are in a loop or ring. Transmitting end of one node is connected to the receiving end of the other node until a closed loop is formed.

Switched fabric: All nodes in FC network or loops of nodes communicate with one another through an FC switch or multiple interconnected FC switches. Switches are interconnected through fibre channel. A link is formed between two switches, that link is referred as an inter-switch link (ISL).

3.1.3 Internet Protocol SAN

It entails the technologies that transfer storage traffic over internet protocol (IP) based network. Conventional SAN transfers storage data over the fibre Channel. Organizations require high performing and scalable SAN at low cost. The emergence of IP technology enables a way for storage data over IP network. If the storage data is sent over the IP network, the existing network infrastructure can be utilized efficiently and there is no need of laying an additional network for SAN. This is more economical than investing in FC, since it avoids new SAN hardware and software installation. IP network offers easier management and better interoperability. Since the storage data is transmitted over IP network, the distance is not a problem. IP SAN make use of all available IP network solutions for the better performance of storage network. Two protocols are primarily used in IP storage networks, they are:

- Internet Small Computer System Interface (iSCSI)
- Fibre channel internet protocol (FCIP)

3.1.3.1 Internet Small Computer System Interface

Internet Small Computer System Interface is an IP based storage protocol for connecting servers and storage. iSCSI is an end to end protocol. The topology of iSCSI is represented in *Figure 8*. It encapsulates SCSI commands over IP networks; iSCSI managed to facilitate data transfers over long distances. iSCSI can be used to transmit data over local area networks (LANs), wide area networks (WANs), or the Internet and can enable location-independent data storage. It operates on top of TCP. It allows IP hosts to access IP or Fibre channel connected iSCSI targets. It also allows fibre channel hosts to access IP targets. The *Figure 7* shows the protocol stack.

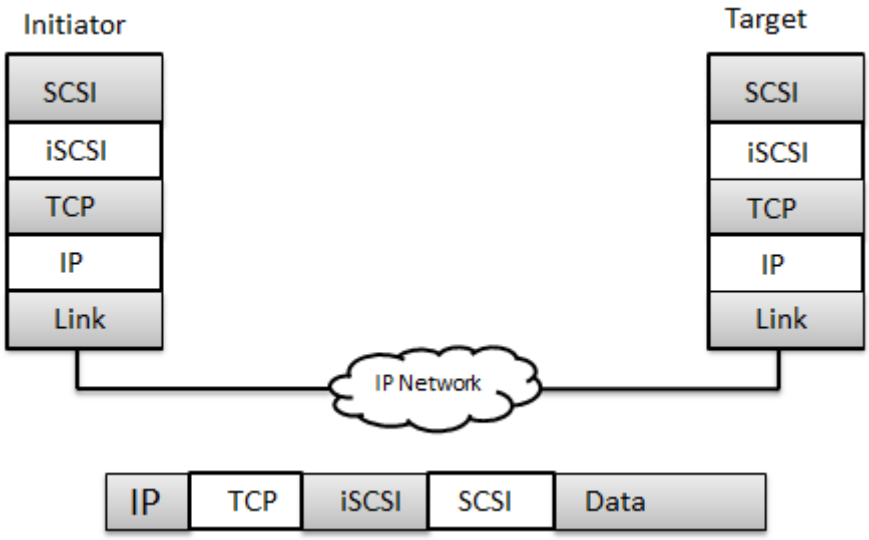


Figure 7 iSCSI layer in protocol Stack

SCSI layer: SCSI layer issues SCSI commands depending on the request from the upper layer.

iSCSI layer: iSCSI layer it encapsulates the SCSI command and data. Then it forwards the iSCSI pdu with iSCSI header to the underlying TCP layer.

TCP: This is the transport layer and third layer of the TCP/IP stack. The two main protocols used in transport layer are UDP and TCP. iSCSI utilize the underlying TCP but not UDP, because UDP provides connectionless unreliable service. SAN network requires a reliable transport connection. The term reliable connection is used where it is not desired to lose any information that is being transferred over the network through this connection. So, the protocol used for this type of connection must provide the mechanism to achieve this desired characteristic. TCP provides an end-to-end, connection-oriented, reliable communications service. TCP is responsible for the establishment of a transport connection, sequencing and acknowledging sent packets and the recovering of lost packets during transmission. It takes care of data flow control, congestion control and error control. TCP segments the data from the above layer into proper sized chunks and then passes these chunks onto the network. It acknowledges

received packets, waits for the acknowledgments of the packets that it sent and sets timeout to resend the packets if the acknowledgements are not received in time.

IP: This layer is also known as Internet layer. It is the second layer of the TCP/IP model. The core protocol of the Internet layer is IP. The internet layer is responsible for addressing, packaging and routing functions. The position of Internet layer is between link layer and transport layer. If needed, internet layer fragment the data from the upper layer thus the packets are in proper size to pass over the network. Each fragmented IP datagrams contain source and destination address (logical address or IP address) information that is used to forward the datagrams between hosts and across networks. IP at the destination rearrange fragmented packets as how it was before fragmentation. It provides connectionless network route between the transmitter and the receivers. Hence it is not necessary that all the packets should travel via the same route. At the destination side, the data packets may appear in a different order than they were sent. It is the job of the higher layers to rearrange them in order to deliver them to proper network applications operating at the application layer.

Link Layer: It transfers data that it receives from the network layer of one machine to the link layer of another machine. The link layer defines the procedures for interfacing with the network hardware and accessing the transmission medium. The link layer moves network frames between two hosts. The hosts may be end systems, such as computers or intermediate devices such as routers and switches. The link layer only moves frames directly between two physically connected devices. All other tasks are the responsibility of the upper levels.

iSCSI works with Initiator and Target model:

Target is an SCSI storage device which is capable of receiving the request from the SCSI initiator and executing it. At first in the target server iSCSI targets are created with a unique iqn name and LUNs are added to the target. Now the target is ready and it is listening to the TCP port for any request from the iSCSI initiator. The target can be enabled to accept any initiator or specific initiator depends on the access rights the target has in its initial settings. To provide access only to specific initiators, the initiator names can be mentioned in the target settings.

iSCSI target naming:

Targets and initiators require names for the purpose of identification, so that iSCSI storage resources can be managed regardless of location. The iSCSI name is the unique identifier for an iSCSI node and it is also the SCSI device name .The iSCSI name is the primary information used in authentication of targets to initiators and initiators to targets. This name is also used to identify and manage iSCSI storage resources. iSCSI names are associated with iSCSI nodes not with network adapter cards. There are two iSCSI naming formats:

- iqn - iSCSI qualified name
- eui -extended unique identifier

Brief description about iqn format is below, since this format is used in my iSCSI SAN set up.

iqn. date. organization or your domain name: storage-identifier

Example: iqn.2014-09.com.examplename:storage.disk1

Initiator is an SCSI server which is capable of issuing SCSI commands to the iSCSI target, based on its storage requirements. iSCSI initiator name is created at iSCSI driver at load time of the host system. iSCSI initiator issues a discovery command to search any available targets. Once the target is discovered, it can login to the targets. A session is created between the target and the initiator once it is logged in. Now the iSCSI target device is accessible for the initiator. The session termination is accomplished by simple logout.

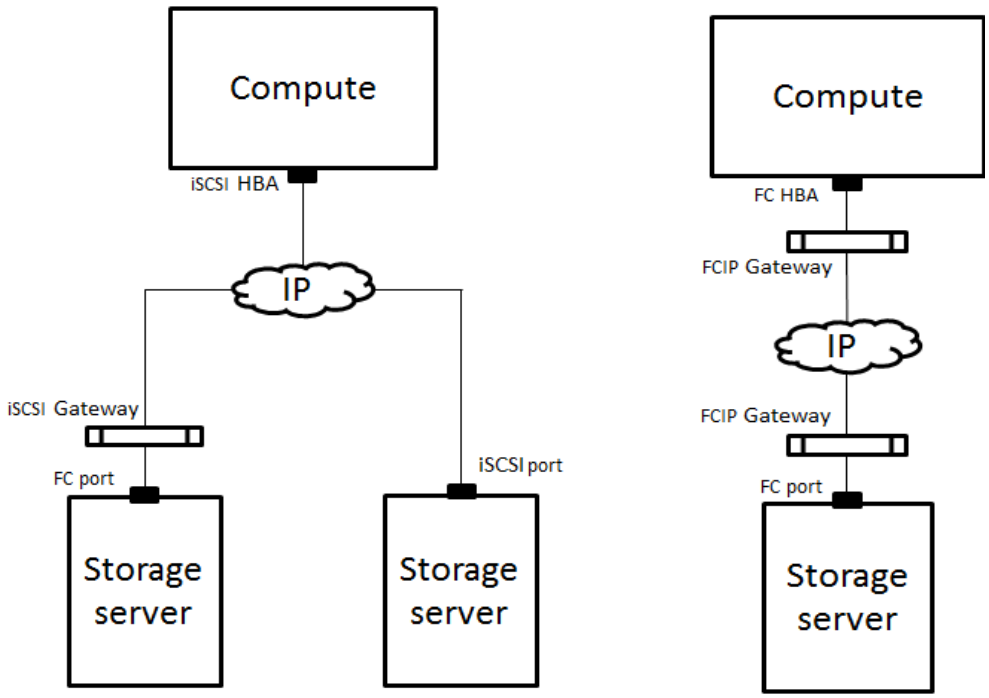


Figure 8 iSCSI and FCIP

3.1.3.2 Fibre Channel Internet Protocol

FCIP defines a way for encapsulating the fibre channel frames within the TCP/IP, which are transferred between two FC SAN. FCIP uses a pair of bridges communicating over TCP/IP, which is the transport protocol. The two bridges acts as a gateway for the FC SAN. The communication path between two FC SAN is over an IP network. FCIP is a tunneling protocol. Between the two bridges a tunnel is created in the IP network and the FC frames are sent over this tunnel. The advantage of using this FCIP bridging concept is, FC networks can be extended over distances using an existing IP-based infrastructure. The topology of FCIP is represented in *Figure 8*.

3.1.4 Fibre Channel over Ethernet

It is a protocol that encapsulates fibre channel frames, to transfer it over the ethernet networks. As explained before, if the organization is using FC for SAN and IP networks for LAN it results in two network infrastructure. Thus organizations have to run parallel network infrastructures for their local area networks and their storage area networks. It leads to huge maintenance and implementation cost. FCoE gives solution to this problem by enabling the consolidation of SAN traffic and ethernet traffic on to single converged ethernet infrastructure. It also reduces the administrative overhead and the complexity in managing the data center.

3.2 Network Attached Storage

Data or information is shared across the organization or within the organization for their day today business activities. Therefore there is always requirement for storing the data at one place and shared it among the users. It is more efficient than transferring the data personally to each of them. NAS addresses this problem. In NAS, the storage devices are connected to the network that provides file level access to the compute system. NAS is a dedicated high performance file server with storage system. It also allows us to consolidating disk management; instead of duplicating directories on each system, it provides a single copy of directory which is shared by all systems in the network. Besides it allows clients to share files between them.

NAS utilizes network and file sharing protocols, which include TCP/IP for data transfer and Common Internet File System (CIFS) and Network file system (NFS) protocols for remote file services. NFS is taken in this example scenario. NFS is a distributed file system protocol that enables clients to access the files from remote. The server enabled with NFS technology is called as NFS server, with file system in it and makes them available on the network. And client server enabled with the NFS client feature can access the files provided by the NFS server from remote. The NFS server export the file to the client and it is mounted in the client server. Thus this works on a basic client–server model. The *Figure 9* shows the basic idea behind NAS with NFS.

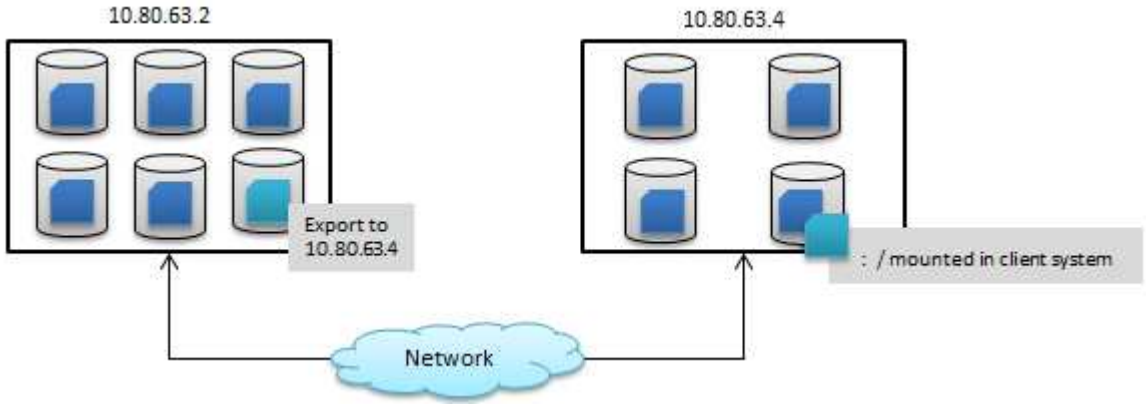


Figure 9 NAS with NFS

3.2.1 Why we need NAS

SAN connects the client servers and the storage servers as descried in 3.1.1. Then the disk in the storage array can be accessed by client server. The data written on the disk cannot be shared among the different users. Sharing the same disk among different users has two main problems; disk space allocation inconsistency and file data corruption. If the same disk is exposed to different servers, each server applies its own file system on the disk. Thus the file allocation table which is in the memory of each server is not exposed to other servers. The file allocation table and file system cache becomes servers specific; it is not common for all the users. For example consider a disk that is shared between server one and two. And both want to write on the same disk at same time. Since the file system is not common, if server one is writing on the disk, server two does not aware of it. It is vice versa, if server two is writing. Both write at the same time, this leads to data corruption. In read scenario, each time the servers will not reach the disk to retrieve data, rather the file system of each server retrieve the data from its cache when the request is same. But when server one wrote on the disk its own cache gets updated but not others. So the other server still uses the old information in its own cache. Like the same way there is problem in file allocation as well. In its own file allocation table, server may see some free blocks and it will assign it to the application as per the request. But it does not aware whether this block has been already allocated to any other application by another server. Thus SAN creates big complexity in sharing the same disk which is solved by NAS by having centralized file system. Hence with the centralized file system, file allocation table and file system cache become common for all the clients accessing the file system.

4. STORAGE AREA NETWORK DESIGN FOR CLOUD INFRASTRUCTURE

While designing a SAN infrastructure, several factors should be kept in mind. Because there exists lots of different methodologies or ways or components, that increase or decrease the overall SAN performance. Each has its own ways of working principle and has its own pros and corns. So the different possibilities are analysed below to obtain a most reliable and highly performing SAN infrastructure.
The infrastructure requirements are

- The iSCSI target server should be VM
- The iSCSI initiator can be either a VM or host
- The operating system is LINUX
- The hypervisor is KVM/QEMU
- Ericsson hardware

4.1 Kernel virtual machine (KVM) and quick emulator (QEMU)

In this SAN design, Linux operating system and the KVM/QEMU hypervisor is employed. Because it is popular open source software, it would not lead to any additional cost for the organization and it is easy to contribute in development.
Quick emulator (QEMU) is an open source machine emulator and virtualizer. When QEMU runs as a machine emulator, it emulates hardware so that operating systems and programs made for a specific hardware can work on a different hardware. But the performance is very slow since the guest operations are executed on the emulated hardware. When QEMU runs as a virtualizer, QEMU achieves performances close to native performance by executing the guest code directly on the host CPU. QEMU functions as a virtualizer when executing under the Xen hypervisor or using the KVM kernel module in Linux. QEMU can make use of KVM when running a guest architecture that is the same as the host architecture. For instance, when running qemu-system-x86 on an x86 compatible processor, QEMU can take advantage of the KVM acceleration. KVM is a special operating mode of QEMU that uses CPU extensions i.e. hardware-assisted virtualization for virtualization via a kernel module. It gives the benefit for both the host and the guest system.
Kernel-based Virtual Machine (KVM) is a full virtualization infrastructure built into the Linux kernel. KVM has support only for x86 processor enabled with virtualization extension and provides hardware assisted virtualization. KVM allows a user space program (QEMU) to utilize the hardware virtualization features of various processors.
Hardware-assisted virtualization is an approach to enable efficient virtualization. It

means, even though the guest is running in the separate address space, it allows the guest to execute its codes directly on the physical hardware instead doing it on emulated hardware resources. Thus it avoids binary translation and increases the performance.

4.2 KVM-QEMU I/O architecture

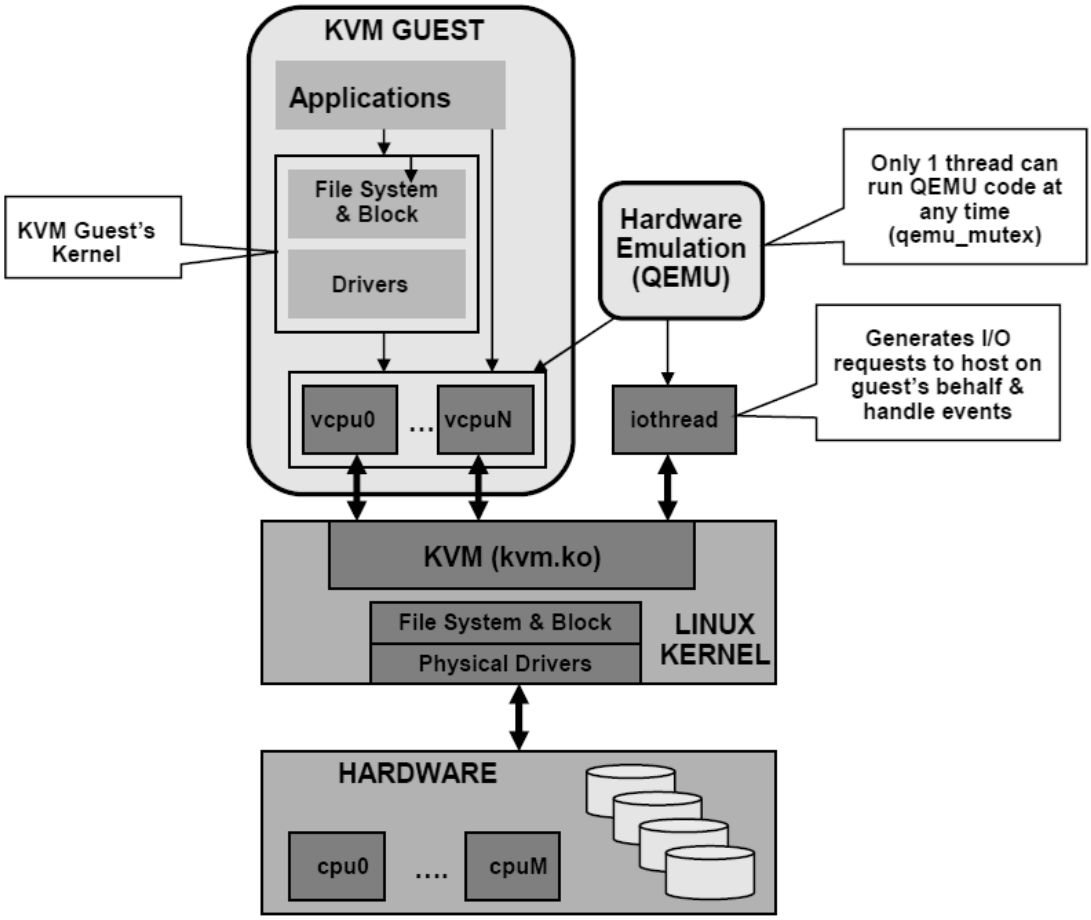


Figure 10 KVM/QEMU I/O architecture

When running in virtualized system, application and guest operating systems are normally not aware of virtualization and for them it is like as if they would run on the bare metal hardware. This is possible because of the benefits of both KVM and QEMU. KVM provides CPU virtualization and the QEMU is responsible for hardware emulation. Regarding KVM, as it can be observed in the previous *Figure 10*, each guest CPU has a dedicated virtual CPU thread that uses KVM kernel modules in the LINUX kernel to execute the guest code. When there is a request coming from an application, the process code executed on the hosts physical CPU but the I/O is taken care by the QEMU. Only one thread can execute QEMU code at a given time. It is represented in *Figure 10* [33]. Application, on the other hand can access only the emulated hardware and thus it requires binary translation. QEMU does binary translation and forward I/O to the host kernel. Host kernel treats this I/O as like any other user application. Therefore the I/O thread is generated by QEMU on behalf of the guest. Thread uses the event loop

to handle the events. ***A thread*** is the execution of the little sequence of program or a process. A process can have multi thread that shares the resources allocated for that process. QEMU does not support multi-threading. As stated in [43] ***multi-threading*** operating systems made it possible for one thread to run while another was waiting for something to happen. The processor switches between different threads based on the pre-defined timing policies. ***Event loop*** receives the events from the event queue of the operating system and pass it to the program for the processing. It is a link between the user space program and the operating system.

The *Figure 11* shows the preliminary lab setup. Two physical and one virtual server were employed in the implementation. Two physical servers were connected through the switches. The virtual machine was running in one of these physical servers. One of these physical servers was acting as an iSCSI-initiator. The initiator can be either a host or a physical machine. There are different possibilities for setting up an iSCSI target server

1. It could be run on a physical host by using local disk
2. It could be run in a virtual machine by using the virtual disk or
3. The iSCSI target can be provided by the storage vendor externally

The iSCSI target provided by the storage vendor has its own target and initiator driver mechanisms, through which the clients can communicate to the storage system. Thus it needs as additional driver and switches and other connecting devices to connect all clients to target in the SAN network. Thus by it forms a centralized storage system. For the some application it may not be suitable. For application which needs faster storage I/O communication such that with reduced latency it’s better to have an iSCSI target in the host or the VM. The reasons are provided below.

The host has three SSDs (solid-state disk) as a storage device. SSD is an electronic disk that uses integrated circuit assemblies as memory to store data persistently. SSD does not have any moving mechanical components. This makes it to differ from the traditional electromechanical magnetic disks such as hard disk drives (HDD) or floppy disks, which contain spinning disks and movable read/write heads disks. SSDs have more benefits when compared to HDDs. It has lower access time, reduced latency and less noisy. And also it has more resistance to physical disturbances. But the price of SSDs is much higher than the HDD. Organizations should consume the SSD store space efficiently, to avoid over cost. Apart from storage space needed for storage real data, there is also need of additional storage for data replication, it is a very important feature which saves organization most critical data and enables reliable data availability. And storage required for migrating the VM from one storage device to another in case of failure or fault. Even though the price of SSDs are reduced over the period it is still very expensive than HDDs.

Hence organization cannot employ SDD for all application for economic reasons. For the application where the speed is not a crucial factor HDD can be opted and for application where storage I/O speed is really a crucial factor SSD can be employed.

Since the Ericsson hardware has already three SSDs in the compute node, it is better to utilize.

In this SAN solution virtual machine is opted to run as target server not the host, in order to efficiently utilize the resources of the host. It is possible to create several virtual machine in the same host depends on the resource availability. The iSCSI–client can be a virtual machine or any other physical machine including physical machine where the target is running.At first the VM is started with help QEMU and KVM. The below script was created to start the VM. The characteristics of the virtual machine like CPU, network interfaces, memory, language etc. are defined in the below scripts. For explanation of the scripts see the appendix [1].

```
/usr/bin/qemu-system-x86_64 \
-enable-kvm \
-cpu phenom \
-smp 1,sockets=1,cores=1,threads=1 \
-drive file=/home/ubuntu.img,if=virtio,cache=directsync \
-drive file=/dev/sdb1,if=virtio,cache=directsync  \
-drive file=/dev/sdc2,if=virtio,cache=directsync  \
-netdev tap,id=tap2,fd=58 58<>/dev/tap58 \
-device virtio-net-pci,netdev=tap2,mac=c6:af:77:89:bc:39 \
-netdev tap,id=tap3,fd=62 62<>/dev/tap62 \
-device virtio-net-pci,netdev=tap3,mac=d2:23:b9:bb:93:00  \
-netdev tap,id=tap4,fd=61 61<>/dev/tap61 \
-device virtio-net-pci,netdev=tap4,mac=b6:a2:c9:c2:59:a9 \
-boot c \
-m 3000  \
-vnc :3&
```

Program 1 Virtual machine start-up script

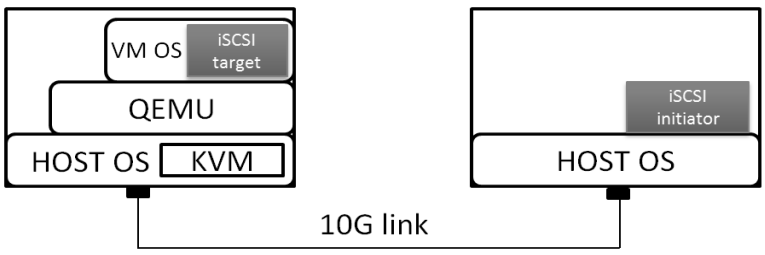


Figure 11 Lab setup

Then iSCSI target is configured inside the virtual machine and the iSCSI client is configured on the other host called iSCSI-initiator. The target always listens to its TCP port for any request from initiator. The iSCSI-client discovers the target and sends the request. Once the initiator logged into the target, it appears as a local disk and it is ready for usage.

4.3 iSCSI targets

As it was explained in 3.1, the storage might come from an external server, if server‘s own local storage space is not enough for the application. As explained before in 3.1.3.1 this can be accomplished by iSCSI. To achieve that in iSCSI, we require an iSCSI initiator and an iSCSI server. The iSCSI server is also known as an iSCSI target and in this section a description and a comparison among the most widely used iSCSI targets is performed. Thus the storage can be scaled in or out, by providing an external storage support. There are different open source iSCSI target frameworks which can be supported by Linux operating system.

- STGT
- LIO
- SCST
- IET

4.3.1 IET (iSCSI Enterprise Target)

It is an iSCSI target designed to run in the Linux kernel. Its aim was to develop an open source iSCSI target with features that works well in large scale environment under real workload. This target was quite popular, but is slowly deteriorating. Its successor is SCST, came up with advanced features and it is quite popular among the SCST targets. Hence it is now unsupported in Linux kernel.

4.3.2 SCST (SCSI Target Subsystem)

It is a generic SCSI target engine for Linux. SCST devices can provide advanced functionalities like replication, thin provisioning, deduplication, high availability, automatic backup, etc. SCST is a GPL licensed SCSI target framework. The SCST implementation is an improved version of IET target. SCST devices have support for many communication link such as iSCSI, Fibre Channel, FCoE, SAS, SCSI RDMA Protocol, InfiniBand (SRP), Wide (parallel) SCSI, etc. It supports multiple storage backend interfaces; SCSI pass-through, block I/O and file I/O. It has ISCSI target support in kernel and also user-space. SCST storage drivers can be implemented in user-space with support of the scst_user driver.

4.3.3 STGT (SCSI Target Framework)

It is a standard multiprotocol SCSI target in Linux. STGT aims to simplify creation and maintenance of SCSI Targets such as iSCSI, fibre channel, SRP etc. Its key goals were the clean integration into the SCSI-mid layer and implementing a great portion of the target in user space. STGT consists of kernel-space and user-space code. For iSCSI target only user space code is utilized. It means STGT run as an application in user

space. STGT was superseded by LIO with Linux kernel 2.6.38 with many advanced features.

4.3.4 LIO (Linux-IO Target)

It has been the Linux SCSI target since kernel version 2.6.38. It supports different fabric modules like Fibre channel, FCoE, iSCSI, SCSI RDMA Protocol InfiniBand (SRP), etc. The advanced feature set of Linux-IO Target has made it more popular among other SCSI targets. QEMU/KVM, libvirt, and open Stack provide native support for LIO. This makes organization to consider LIO as a storage option for cloud environment. It is designed to support highly available and cluster storage. LIO includes targetcli. Targetcli is a command line interface, provides a powerful and easy way to configure and manage LIO.

4.4 iSCSI target perforamnce results

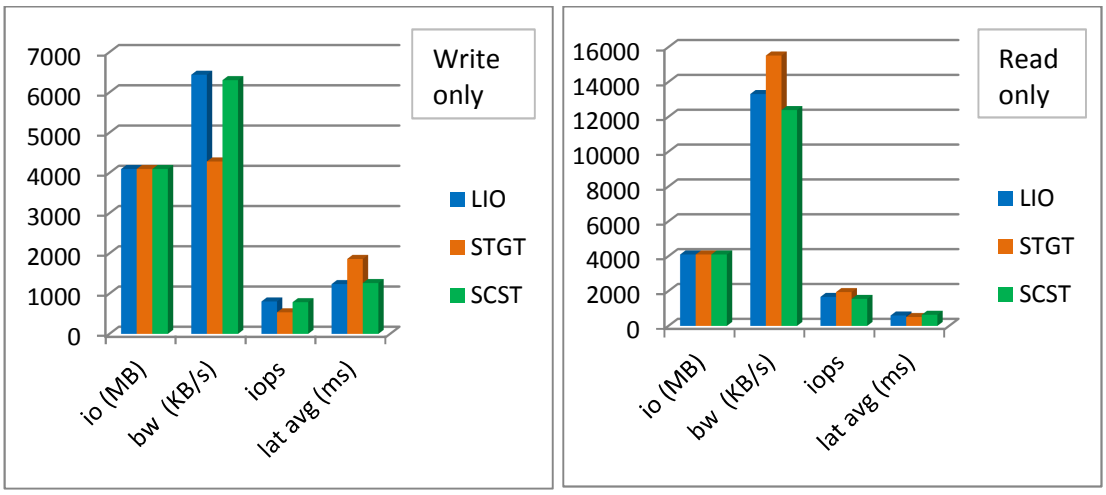


Figure 12 Write performance and Read performance

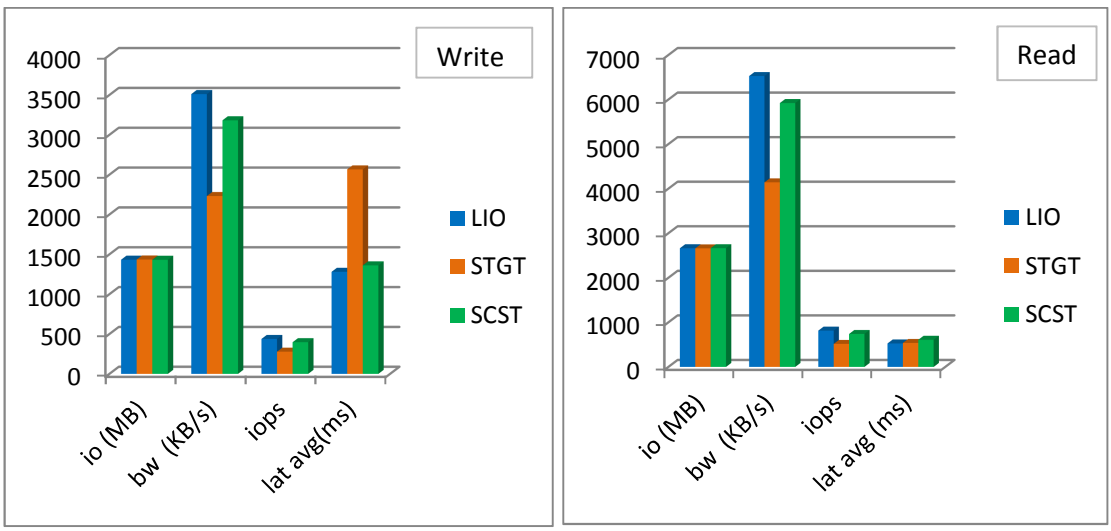


Figure 13 Performance of read and write in parallel

io= Number of io performed (mb)
bw= Average bandwidth rate(kb/s)
iops= Average IOs performed per second
lat avg= Average latency(msec)

All of these targets performances are conducted in similar environment. File I/O is configured as a back store for all targets. In aspects of features, both SCST and LIO have most advanced features than STGT and IET. It is clear that the competition is between LIO and SCST. IET's performance is not measured here, since it is now unsupported. Thus it is out of the race. Performance comparison is between SCST, LIO, and STGT. These three targets are running in a three different virtual machines and the targets are discovered and utilized by another machine as shown in *Figure 11*.Three different test cases were executed they were, only writing on the disk, only reading from the disk, performing read and write parallel on the same disk. This performance test was conducted with help of FIO tool.

Flexible I/O (FIO) is a tool to measure I/O performance against different storage types. While comparing the features of SCST and LIO, each has benefits in its own way. In aspect of performance it is clear that LIO won the competition. From *Figure 12* and *Figure 13*, it is clear that LIO has better throughput with less latency. Next in line is SCST. And STGT hold the last position. But in right side of the *Figure 12*, STGT's performance is better than other two. This is because; it is only reading from the disk such that it requests the data which already exists. Most of the scenarios, if the requested data is same; data is retrieved from the cache but not from the disk. So it is predicted that STGT's better performance is due to cache. This is in practice to increase the performance. This is not applicable in write, because most of the times write data differs. So the possibility of fetching from the cache is less. Thus while measuring performance it is better to consider the write performance. The different modes of cache and its effects are described elaborately in next chapter.

LIO has been chosen as iSCSI target in this SAN for the following reasons: because of its performance, since it is included in most Linux distributions, the support provided for LIO in QEMU/KVM, libvirt, and open stack. This makes LIO target configuration very easy, reliable and stable. In case of SCST, it does not come with Linux Kernel, so it consumes more time and effort to implement and configure. And also the stability is not assured, since it is externally patched to the LINUX kernel. Now the focus is more on LIO. The general Architecture of LIO is presented below.

4.5 LIO architecture

Linux-IO Target is a Linux SCSI target. LIO offers a unified storage system; LIO as a single infrastructure that functions to support simultaneously fibre channel, FCoE, iSCSI, InfiniBand etc. The target core engine implements the semantics of a SCSI target, but it does not directly communicate with initiators and it does not directly

access data on disk. Instead it communicates through the fabric modules. Fabric modules are the frontend of the SCSI target. It communicates with the specific protocols that transport SCSI commands. Backing stores implement methods of accessing data on disk. This includes RAM disk, file, block device, and SCSI pass-through. It can support file-based storage access and block-based storage access. It means the storage can be a file or the block device, but in both the case it is presented to the initiator as block device. The LIO architecture is shown in *Figure 14* [18] and logical representation of the LIO configuration is show in *Figure 15*. For the real command line configuration please check the appendix [2].

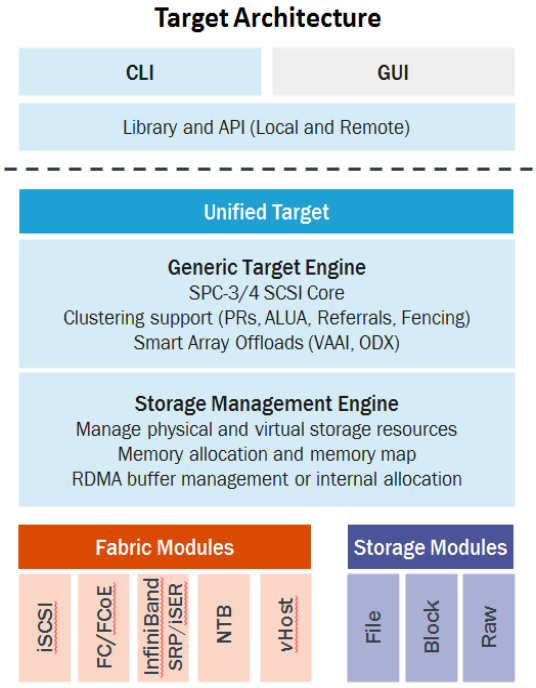


Figure 14 LIO architecture

The following features of SCSI protocol were implemented in Linux IO target which makes it even more popular.

Persistent Reservations (PRs): It is a feature which enables I/O fencing. Fencing is the process of isolating a faulty node from a cluster of nodes. Thus Persistent reservations provide the capability to control the access of each node to storage devices.

Asymmetric Logical Unit Assignment (ALUA): ALUA defines a standardized protocol in the SCSI standard for accessing a LUN through multiple controllers of a storage system). It is a multi-pathing method with intelligent path selection. ALUA allows the data to reach the LUN via the most optimized path. It means all the traffic that is directed to the non-active controller will be routed internally to the active controller. For a storage system with a single controller, there is no demand for ALUA.

Error Recovery Levels: The iSCSI standards were defined in internet engineering task force, with three hierarchy error recovery levels, which is now included in LIO. They are, as follows

1. ERL=0 Session recovery class: Session recovery implies the closing of all TCP connections, internally aborting all executing and queued tasks for the given initiator at the target, terminating all outstanding SCSI commands with an appropriate SCSI service response at the initiator, and restarting a session on a new set of connection.
2. ERL=1 Digest failure recovery: It includes the capabilities of ER Level 0. Digest failure recovery is comprised of two recovery classes: within-Connection recovery class and within-Command recovery class.

Within-Connection recovery class: At initiator side if the requests are not acknowledged for a long time by the target and the target side status/response not acknowledged for a long time by initiator enables within-connection recovery. Requests are acknowledged explicitly through ExpCmdSN or implicitly by receiving data and/or status. Next expected command sequence number (ExpCmdSN) is a sequence number that the target iSCSI returns to the initiator to acknowledge command reception. The initiator may retry non-acknowledged commands.

Within-Command recovery class: The loss in data PDU enables the within-command recovery. At initiator and target side it has different schemes to find out data PDU loss.

3. ERL=2 Connection recovery class: It includes the capabilities of ER L=0 and ERL=1. The following scenarios at the initiator let it to start the Connection recovery, tcp connection failure and receiving an asynchronous message that indicates one or all connections in a session has been dropped. In case of failure it logout from failed connection and establish a new connection. When a TCP connection failure occurs at the target side, it closes the connection and send asynchronous message to the initiator to start the connection recovery process.

Active-active task migration and session continuation: ERL2 feature of iSCSI provides active-active task migration and session continuation. It increases the system availability. It migrate the connections from failed routes to available connections that prevent session and data loss.

Multiple connections per session (MC/S): MC/S allows creating multiple communication paths in a single session. MCS allows the initiator to establish multiple TCP/IP connections to the same target within the same iSCSI session. It improves performance and fault tolerance.

T10 Data Integrity Format (DIF): T10 Data Integrity Field standard provides a way to check the data integrity between a host controller and a disk. This check is carried out through the data integrity field defined in the T10 standard.

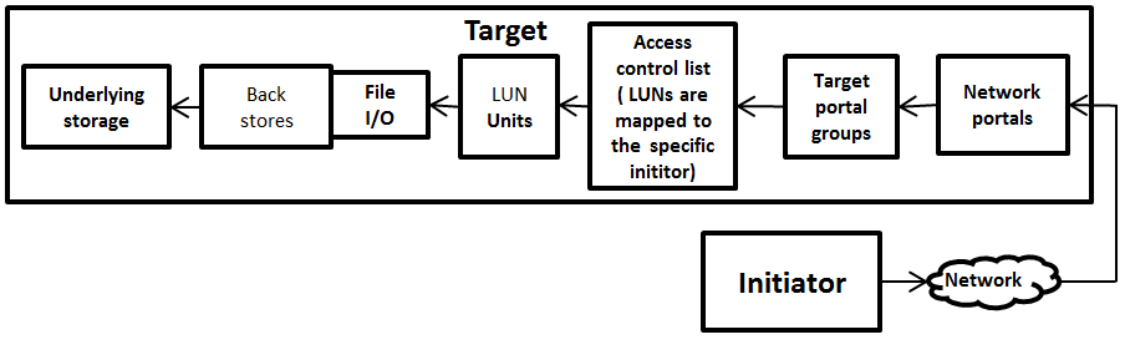


Figure 15 LIO configuration

4.6 Virtual machine networking

Virtual networks have the functionalities same as physical networks, it enable communication between the machines attached to its network. The only difference is the components employed in the virtual networks are virtualized form of physical network elements or the software which exactly behaves like a physical components. Software defined networking is currently emerging and dynamic topic in the networking industry. And one of the key enablers of SDN is virtual networking. A machine must be network-capable to get attached to the network, meaning that it must have a network interface controller installed, that enables the computer to interface with a network. In most physical network environments, computers are usually connected to a device called a switch, which creates a local area network and enables a communication between the computers by routing ip packets. The virtual network follows the same idea. Virtual network consists of one or more virtual machines. The virtual machines in the host can communicate with each other and to the host through a software switch. Thus the switches are used to establish a connection between the virtual network and physical network. Currently Linux supporting three types of software switches

- Bridge
- Macvtap
- Openvswitch

4.6.1 Bridge

Linux Bridge is software, whose functions are similar to the hardware switch. This software bridge is installed within a Linux host in order to emulate hardware, thus the virtual machines with virtual NICs can share a single physical NIC of the host. Some important functionality of bridges includes forwarding data base (FDB), spanning tree protocol (STP), and aging. brctl is a command line interface used to set up, maintain and inspect the bridge configuration in the Linux kernel.

Each bridge has a number of ports attached to it and has its own forwarding database. Whenever new network interface is added to the bridge port, its forwarding

database gets updated with the MAC address of the newly added interface. Thus the bridge keeps record of MAC address seen on each port and it can deliver the frames to the appropriate port which has the destination address. By forwarding the frames to appropriate destination port, it avoids transmitting redundant copies to all ports. However, the ethernet address location data is not a static data. Machines can move to other ports, network cards can be replaced; this leads to confusion in delivering packets to the correct destination. This problem is address by aging.

Aging is a concept of setting up life time for each MAC addresses in the forwarding data base. The MAC addresses are retaining in the FDB, if it receives frames from it. If there is no frame coming from a particular MAC address even after the aging time, the MAC address is deleted from the FDB. Thus the unused MAC addresses in FDB are eliminated by an aging technique.

Multiple bridges can work together to create larger networks. It is good to have spanning tree protocol functionality enabled in this condition, to find the shortest path and for eliminating loops in the network. The spanning tree protocol feature can be disabled in circumstances where it does not make sense, for example when there is only one bridge in the network or when there are no loops in the network. Bridges are capable of running in a promiscuous mode, which means it receives all traffic on a network even which are not intended for it. The below block diagram explains the operational flow of Linux bridge.

As shown in *Figure 16* [28], Linux kernel is with a software bridge installed. The physical interface eth0 of host in added to the software bridge, through which all virtual machine can reach outside network. This behaves like an interface of the bridge. Now the bridge interface has the IP address and the physical interface has only a MAC address.

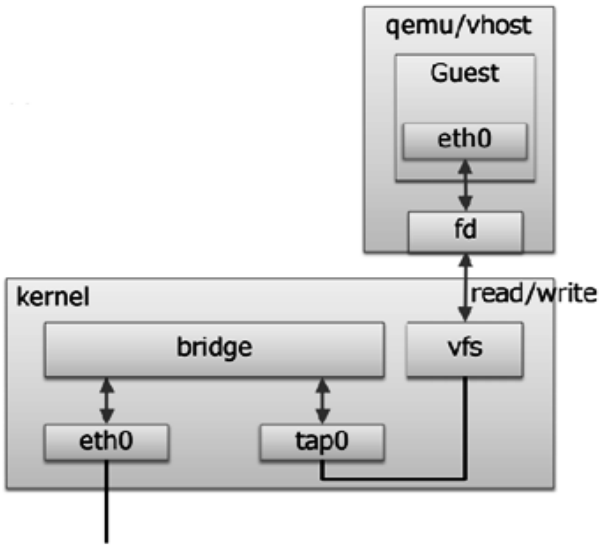


Figure 16 Bridge architecture

A tap device was created to connect the guest to the bridge. A ***tap*** is software device in the kernel, simulates a link layer device. It passes the packets between the kernel and

the guest. Virtual machines network interface can only process ethernet frames similar to physical interface. In virtualized environments, host physical NIC interface is the main interface that receives frames intended for the virtual machines running in it. By nature it receives and process the ethernet frames and then it forwards the payload further up to the operating system. But the virtual NICs expect ethernet frames. The tap device addresses this issue. Tap communicates with the Linux Bridge to forward ethernet frames as it is. Thus the virtual machines connected to tap interfaces will be able to receive raw Ethernet frames.

QEMU define a network interface and creates a network bridge by connecting the host tap network device to a specified network of guest. When the guest attached to the tap interface, it gets a special file descriptor. While writing operation, the guest writes into the file descriptor and tap receives that as input. Then it forwards the data to the bridge. It works vice versa for read. Tap writes into file descriptor and the guest read the data from the file descriptor. Thus the file descriptor works between the tap and the guest.

Virtual file system is an abstraction layer on top of file system. VFS layer provides a uniform interface for the kernel to deal with various I/O requests for different file system. It is clearly shown in the *Figure 17* [34].

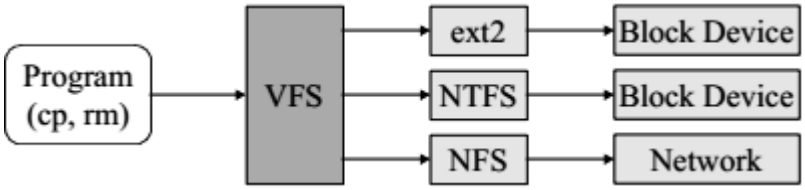


Figure 17 Virtual file system

Finally with all this features bridge brings out the virtual network interface visible to the outside network.

4.6.2 Macvtap

Macvtap allows the direct attachment of a virtual machine's virtual NIC to a physical NIC on its host. Macvtap is a Linux device driver entailed to simplify the complexity in Linux bridged networking. Macvtap is a combination of the Macvlan driver and a Tap device. Macvlan driver is another Linux kernel driver that is utilized in virtual networking to create one or more virtual network interfaces on a physical network interface. Each virtual interface has its own MAC address different from the physical interface's MAC address. Frames sent to or from the virtual interfaces are mapped to the physical interface, which is called the lower interface. Macvlan allows isolation of traffic. Macvlan allows the interface listen to the traffic which has MAC address matches it. Thus the interface does not listen to the traffic which is not intended for it. The *Figure 18* shows the functional flow of macvtap.

Thus by combining the functionalities of macvaln and tap, macvtap creates a tap interface on the physical interface. It can create one more tap interfaces on the same physical interface. Each of these tap interface has a MAC address which is different from host MAC address and other tap interfaces address. Macvtap has four different operational modes.

- Vepa
- Bridge
- Private
- Pass-through

Vepa: All the packets from the virtual machines are sent to the external switch to which the physical interface is connected. If the destination of the packet is on the same host where from the packet originates, the external switch sent it back to host where from it received that packet. By this configuration a virtual machine can communicate with its host, outside network and the other virtual machine on the same host. The drawback of the vepa mode is, even though the virtual machines are on the same host it cannot communicate with it internally, it needs a support from external switch to route back the packets. Not all the switches are capable of doing this. Only the switches support reflective relay mode can do this job. Reflective relay mode means, the switch send back the packet to the same port from where it received it.

Bridge: The virtual machines on the same host can communicate with each other internally. If the packets destination is on the same host as where they originate from are directly delivered to the target macvtap device. Such that the macvtap interfaces can communicate each other if they operate in bridge mode. But the drawback of bridge mode is that the virtual machines cannot communicate with its own host. This is because the packets coming from the guest is forwarded to the bridge. The bridge in turn forwards the incoming packets to the physical interface of the host, which are immediately sent to the outside network, cannot be bounced back up to the host's IP stack. And also the packets coming from the host are sent to the physical interface cannot be bounced back up to the macvtap bridge for forwarding to the guests. This can be solved by creating another virtual interface on the host with the help of macvlan driver. As like in bridge the physical interface eth0 is added to the macvlan virtual interface. Macvlan interface has the ip address and the physical interface has the MAC address. Thus this macvlan interface is utilized instead of eth0 for communication between guest and host. The main advantage of macvtap is easy configuration, but this problem in bridge mode leads to complex configuration as like in bridge.

Private: All the packets are sent to the external switch as like in vepa mode and but the difference is the switch cannot send back packets to the same host. The packets will be delivered to a virtual machine on the same host physical machine only if they are sent

through an external router or gateway. The router route back the packets to the switch and the switch send it to the host where the target virtual machine resides.

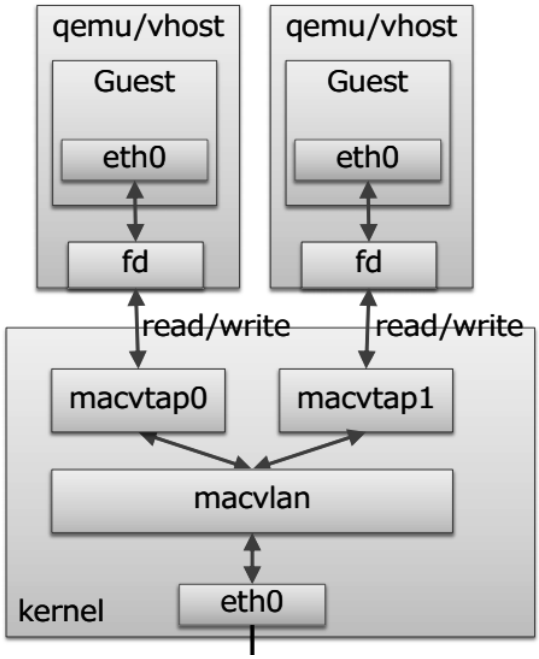


Figure 18 Macvtap architecture

Pass-through: This mode gives control of physical devices to the guests. Each physical interface allowed to use by a single guest interface at a time, thus the scalability is very much reduced.

4.6.3 Openvswitch

Openvswitch is a booming switching technology. The openvswitches are open flow capable, this makes it more popular. And also it leads to consider it in the software defined networking which is a hot emerging technology where most of the networking organization is focusing on. It is supported by openstack for the cloud deployments. It is a distributed virtual switch. The openvswitches are well suited for multi-server virtualization environments.

It has centralized controller. The idea behind centralized controller is to take the intelligence out of the routers in the network and locate it at a central point. Thus it is referred as abstraction of data plane and control plane. It has control plane in the user space and data plane in the kernel. Packet forwarding is performed in the data plane. Routing is performed in the control plane it means exchange of routing information. The centralized controllers will communicate with each one of router in the network with open flow interface. Open flow is a communication protocol that gives access to the forwarding plane of a network switch or router over the network. The controller configures the flow table in the remote router. This router makes decision based on this flow table. The open switch is not implemented and tested in my design solution

because Ericsson concluded that the current openvswitch is limited with features and does not scale. It means that just switching networking traffic results in reaching the limits of the openvswitch. Therefore, adding storage traffic on top would destroy it. The idea of openvswitch is represented in *Figure 19*.

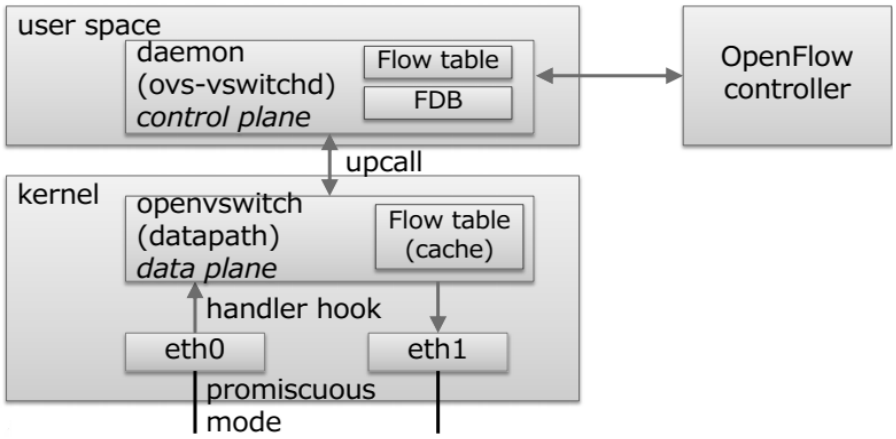


Figure 19 Openvswitch

4.6.4 Performance results of Macvtap and Linux Bridge

Macvtap tap has been chosen for this SAN solution because of the following reasons, better performance than bridge, easy configuration and this single software has different operation modes.

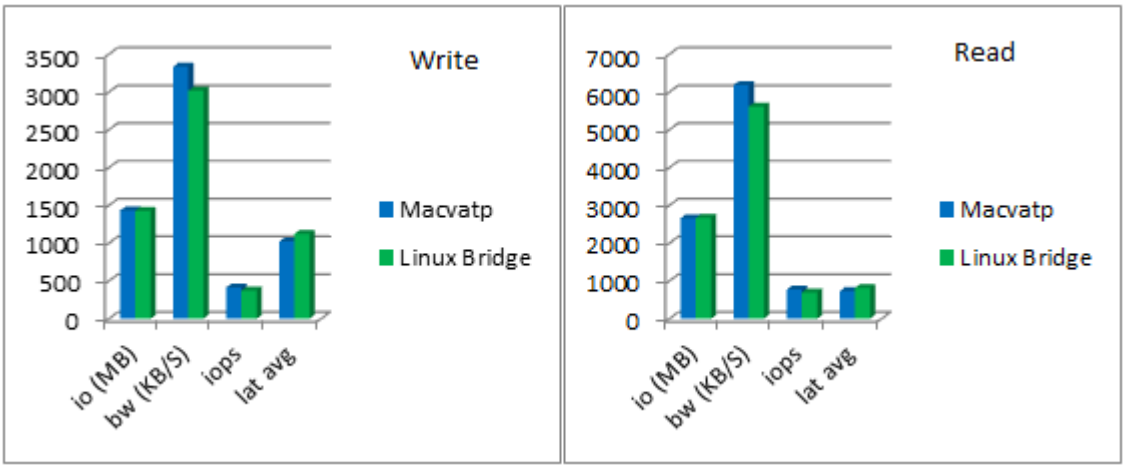


Figure 20 Perforamnce of macvtap and linux bridge

io= Number of io performed (mb)
bw= Average bandwidth rate(kb/s)
iops= Average IOs performed per second
lat avg= Average latency(msec)

5. PERFORMANCE OPTIMIZATION

This chapter presents various techniques to optimize the performance of a storage area network solution. The different methodologies like multipathing, multiple connections per session and NIC teaming improves performance and reliability. The para-virtualized drivers of KVM improve the I/O performance. Para-virtualized drivers are employed in a scenario where there is performance offset due to full virtualization. The cache setting also helps to tune the I/O performance. All of these different possibilities are analyzed elaborately in the following subchapters.

5.1 Multipathing

Multipathing is a technique that allows establishing more than one physical path that enables data communication between the host and an external storage device. By transmitting the payload across the multiple paths, the congestion on the path is reduced and the speed is increased. Thus it increases the overall performance. If one of the path in the SAN network is not functioning due to some failures, the data transfer can be switch to any another physical path which is functioning perfectly. The process of switching from failed path to active functioning path is called path failover. Thus this failover mechanism improves the performance and fault tolerance.

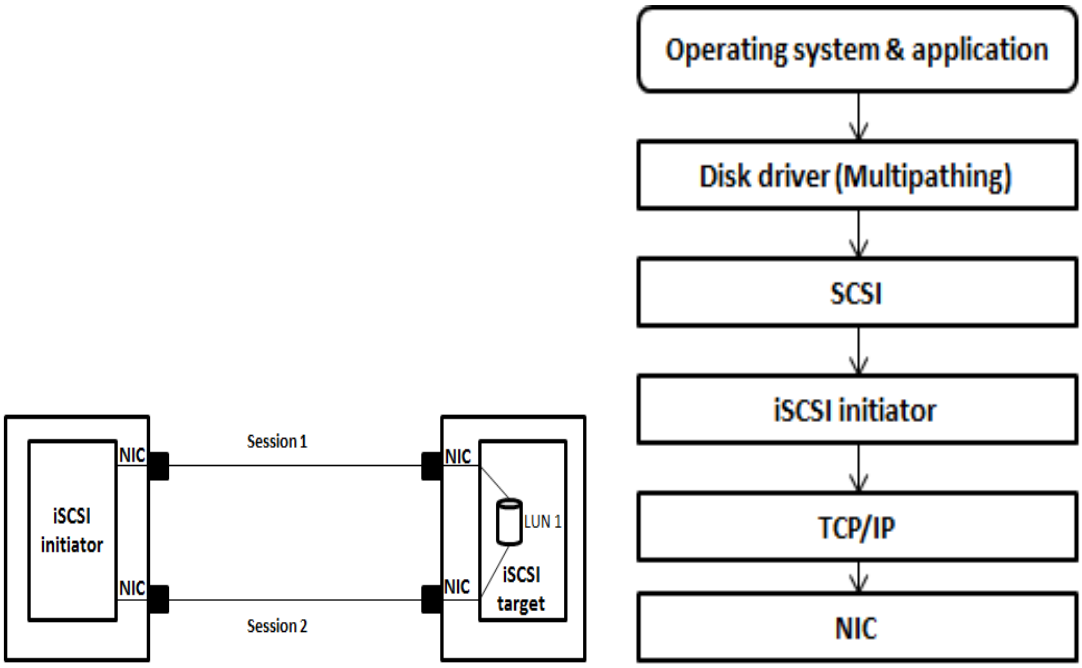


Figure 21 iSCSI multipath sessions and operation flow

The default multipathing driver supported by Linux is MPIO. MPIO allows the initiator to establish multiple iSCSI sessions to the same target, by effectively aggregating the duplicate devices into a single device. Each iSCSI session has single TCP/IP connection as shown in *Figure 21*. Thus a reliable connection between a host and its storage is maintained.

The benefits of MPIO include dynamic load balancing, dynamic path selection, its positioning above the SCSI layer and support given by Linux. Multipathing driver is placed above the SCSI layer; hence a single disk driver is enough to support network transport protocols such as Fiber channel, iSCSI, etc. as shown in *Figure 21*. Multipath I/O has two types of configuration active/active and active/passive. In both scenarios it can provide failover. In active/active mode, the I/O is spread over all paths. In active/passive mode, the I/O is spread over half of the available paths.

In our scenario we have two paths as shown in *Figure 22*, in active/active mode the I/O is transferred over both the paths. If failure occurs in any one of these paths, it just neglects the failed path and start sending all I/O in the active functioning path. In case of active/passive mode, one path is in active mode and the other path is in passive mode. The I/O is sent only over the active path. If failures occur in that path then it switched to the passive path, which becomes active now. Therefore at any cause it utilizes only one path thus the performance of active/passive configuration is less than active/active. Some important features of multipathing includes load balancing policy, path grouping policy, priority based path selection etc. The configuration information is included in the appendix [3] .

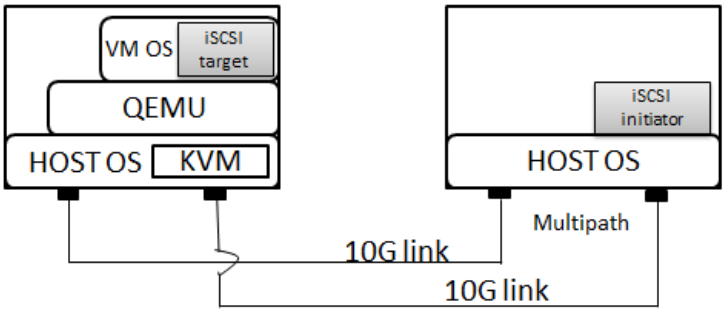


Figure 22 Multipathing lab setup

The failover time is calculated as

```
nop timeout + nop interval + replacement_timeout
```

Nop-out Request and Nop-in Response: This request/response used by an initiator and target as a ping mechanism to affirm that a connection is still active and all of its components are operational.

Replacement_timeout: It controls how long the iSCSI layer should wait for a timed-out path/session to reestablish itself before failing any commands on it.

Lower the value of these parameters, the failover response is quick.

These parameters can be changed based on the requirements. In this configuration the settings are
node.session.timeo.replacement_timeout =1 sec (time to wait for a timed-out path)
node.conn[0].timeo.noop_out_interval=1 sec (intervals between NOP-Out requests)
node.conn[0].timeo.noop_out_timeout= 1 sec (Nop-out requests time out)

The failover time presented here is for the active/active configuration. The theoretical failover time calculated with formula is 3 seconds. With the help of wireshark the practical failover time is calculated as 3.27, which is very close to the theoretical value. The wireshark results are presented in the appendix [5] .

5.2 Network interface card teaming

Network interface card (NIC) teaming allows combining two or more NICs together to make one virtual NIC. It is based on link aggregation. It is a method of combining similar physical links into one logical link as shown in *Figure 23*. It is also called as NIC bonding. NIC teaming is employed to achieve load balancing and failover.

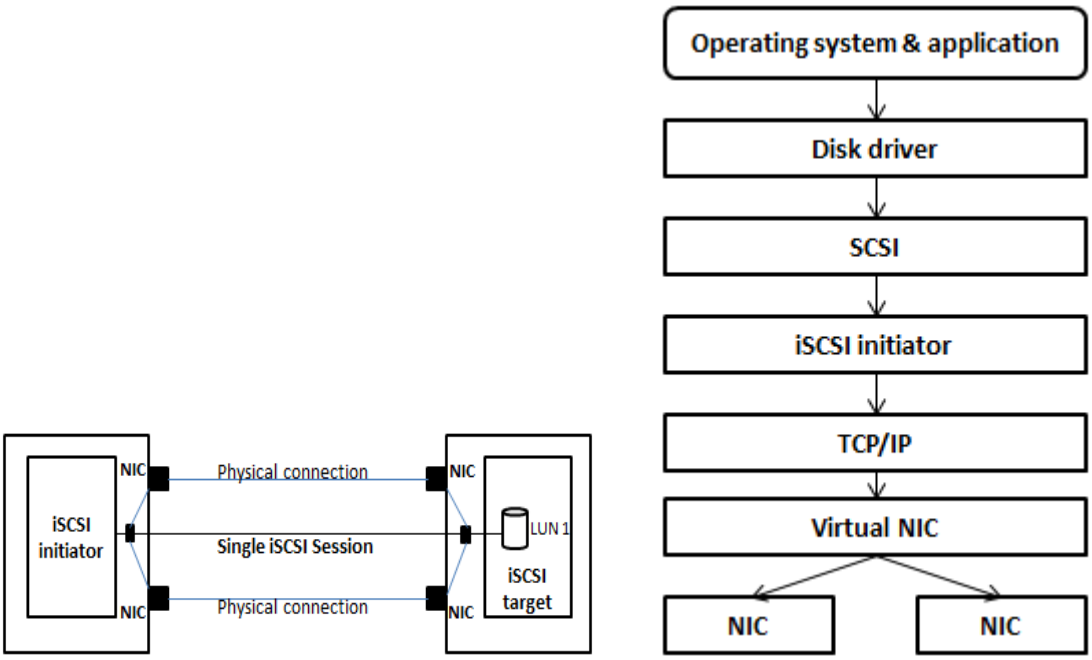


Figure 23 NIC teaming iSCSI session and operation flow

The physical NICs of a server are teamed and presented as a single NIC to the application. If the application requests for data or writing data into the storage target, a single TCP connection for an iSCSI session is created for each request. It is shown in the *Figure 23*. The teamed physical NICs are connected to the network. The data packets of a single TCP connection are distributed among the NICs. Thus, hike in the performance can be expected. In case of failure of any one of the NIC, all the traffic is

routed to another available properly functioning NIC. Hence by providing redundant path, it improves the fault tolerance of a server. But NIC teaming has drawback that makes it less popular in the SAN deployment. Even though the two NICs are teamed, it is not meaning that the performance or the throughput will be doubled, because the NICs are not scaled out properly.

For example if two NICs are used, it is not assured that the packets are sent equally through two NICs. It may send or may not. As mentioned in NIC teaming, there is only one TCP connection between the storage and the server. The TCP segments of a single TCP session are sent through two physical NICs. So the arrival at the destination will be out of order. Each NIC had different delays. Thus in the destination it needs an additional algorithm to arrange it in order. It increases the overall latency. And for the incoming packets there is a need for switch in-between the server and the storage. The switch is configured in such a way that it will send the packets equally across the NICs. But in contrast, in multipathing each path has separate iSCSI session and TCP connections. It means that each NIC has separate TCP connection.

Hence the packets arrived in order and the paths are properly scaled out with specific load balancing function. Therefore multipathing performs better than NIC teaming. NIC teaming works between server and switch. Multipathing works between storage server and client server. NIC teaming is best suited for a network where more intermediate hops are deployed like NAS.

5.3 Multiple connections per session (MC/S)

MC/S allows us to create multiple communication paths in a single session, to improve performance and fault tolerance. Therefore the initiator establishes multiple TCP/IP connections to the same target within the same iSCSI session. It is represented in the *Figure 24.*

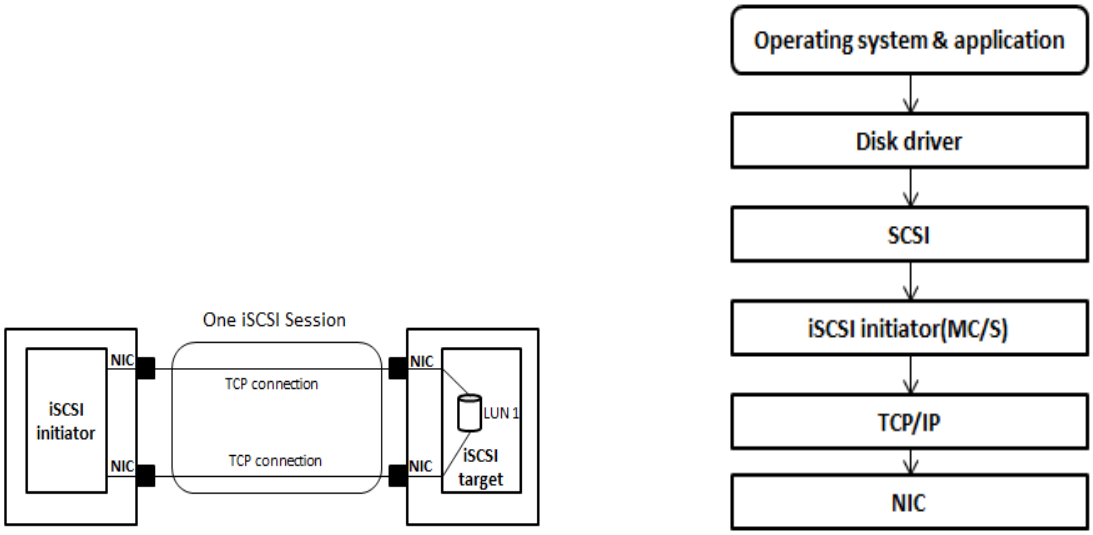


Figure 24 Multiple connections per session and operation flow

MC/S is done on the iSCSI level. Thus, this feature depends on the storage protocols. But in contrast, in MPIO the multipathing feature in is included in the disk driver, hence it supports all the underlying storage protocols. If there is requirement to apply different load balancing policies to different targets it will be better off using MPIO. This is because load balancing policies are session adherent. It means while applying policy to MCS it is for the whole session, no matter how many connections are aggregated in this session.

In MC/S failover recovery is easier. If one connection is failed all the commands are reassigned to another connection in the same session. It is easy because, all the failure recovery actions are taking place within the same session. Hence all the reservations information of the targets and the initiators connected to the device are remaining unaffected. But failover recovery in MPIO is complicated, because here the commands reassignment is between the sessions. All the commands are transferred from faulty session to another active session. Thus, all the session to the specific LUN has been terminated and then the sessions are established newly. Then it starts retransmitting the commands. Therefore the failover recovery time is more and the initiator reservation information in the targets is not persistent. But this can be addressed if the target is supported with persistent reservations feature. It enables access for multiple nodes to a target device and simultaneously blocks access for other faulty nodes. Thus only the faulty session is terminated. The other initiator sessions are unaffected and the information on the target is persistent. In our scenario LIO supports persistent reservation. And it can be concluded that MPIO perform well than MC/S in the SAN with its more advanced feature.

5.4 Effects of cache

The important factor while measuring the performance of a server is the time taken by the server to read or write data from or to its internal storage or the external storage to which it is connected. The server has different levels of cache to increase the read and write performance. The speed of a processor to execute a program depends on the hardware, software and the data locality. Data closer to the processor, the processor can execute that program faster. Thus to increase the performance and to bring the data more near to the processor, the computing system has main memory where the data are cached during read or write operation. Hence the future requests for the same data can be served faster, because it is much closer than the original storage device.

Caching is the processes of storing data temporarily in to the memory to serve the application faster in future, that requesting the same data. Thus the data is fetched from the memory rather than from storage disk. Hence the cache increases the overall performance. The data of a specific application can be cached in the memory until the memory is demanded by any other application. The caches influencing the I/O operation are page cache and the disk cache.

Disk cache is the data cached in the memory of the storage hard disk. Page cache is the data cached in the memory of the server's operating system. But the problem with cache is, the data storage is not guaranteed. During write operation, most of the data are written into the memory rather than in the original hard disk. Once there is a demand for memory from other application then the data is flushed into the hard disk. Otherwise it resides in the memory. In case if there is power failure or any other disturbances that turned off the system, then the data in the memory will be lost.

In our scenario the virtual machine image file has been opened as a normal file in the host, which results in them being cached by the host operating system like any other file. It is represented in the *Figure 25*. When the guest is executing I/O operation, the data is getting cached in the page cache of both the virtual machine operating system and the host operating system. Thus there are two copies of data, leads to more memory consumption. It is always good to bypass any one page cache, so the memory can be utilized by any other application. The virtual machine image file is very large. Caching them can consume more memory of the host operating system. If more virtual machines are running on the host, then it consumes almost all the memory space of the host and then there will be no memory space for any host applications.

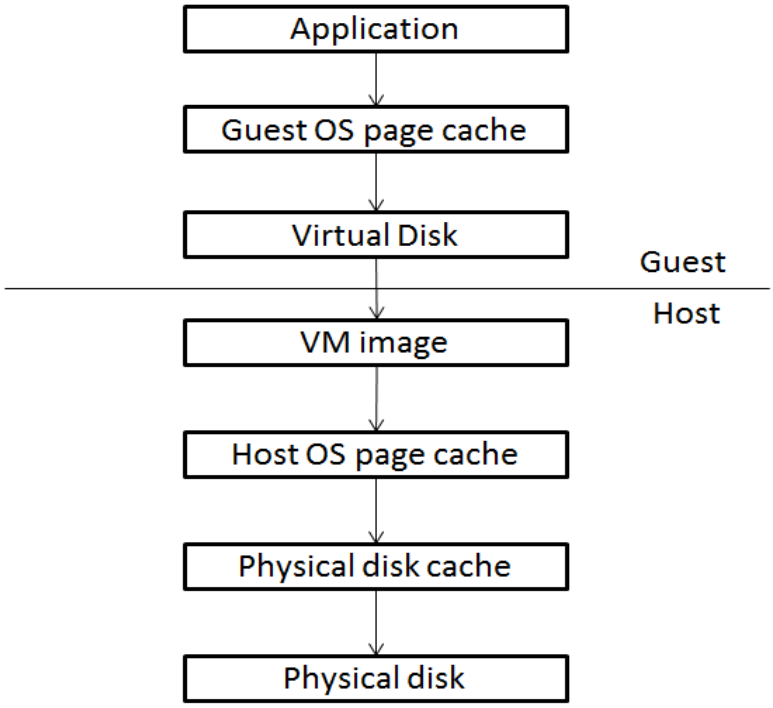


Figure 25 Cache at different level

If the application is writing in the virtual machine, virtual machine page cache can be bypassed with O_sync flag that flushes all data to the disk. But in VM the storage disk is a virtual disk. Thus still the data is not stored in real hard disk; it is cached in the host page cache. Hence the data is insecure. To conclude it is always good to bypass the

host page cache. O_sync instruct the write operations on the file will complete according to the requirements of synchronized I/O file integrity completion. Thus it allows the write command to return and accept the data in the queue once it is completed with write operation of current data and associated file metadata it has, on the real hard disk. There are several cache modes supported by hypervisor KVM and its performances are shown in *Figure 26*.

Cache mode unspecified

If the cache mode is unspecified, the default caching mode is writethrough for the qemu-kvm versions older than version 1.2. After that version, the default caching mode is writeback.

Cache = writethrough

Host page cache – used *Disk write cache – bypassed*

The qemu-kvm uses O_DSYNC semantics, to communicate with the storage device where writes are reported as completed only when the data has been written completely on the storage device. The host page cache is used and the disk write cache is bypassed. So there is no need to send flush commands to manage data integrity.

Cache = writeback

Host page cache – used *Disk write cache – used*

The qemu-kvm uses neither O_DSYNC nor O_DIRECT semantics, to communicate with the storage device where writes are reported to the guest as completed but actually written on the host page cache. And the flush command would be expected to send down data to the real storage to manage data integrity.

Cache = none

Host page cache – bypassed *Disk write cache – used*

The qemu-kvm uses O_DIRECT semantics, to communicate with the storage device, so where writes are reported as completed but actually written on the disk write cache. The host page cache is bypassed and the disk write cache is used. Thus the data are placed in write queue only, so flush commands would be expected to send down to manage data integrity.

Cache = unsafe

Host page cache – used *Disk write cache – used*

This mode is similar to the cache=writeback mode. The unsafe mode ignores all flush commands from the guests. This mode is used in the scenario where the user has accepted to compromise with risk of data loss in the case of failure situations over performance.

Cache=directsync

Host page cache – bypassed *Disk write cache – bypassed*

The qemu-kvm uses both O_DSYNC and O_DIRECT semantics, to communicate with the storage device, where writes are reported as completed only when the data has been written completely on the storage device. The host page cache is bypassed and the disk write cache is bypassed.

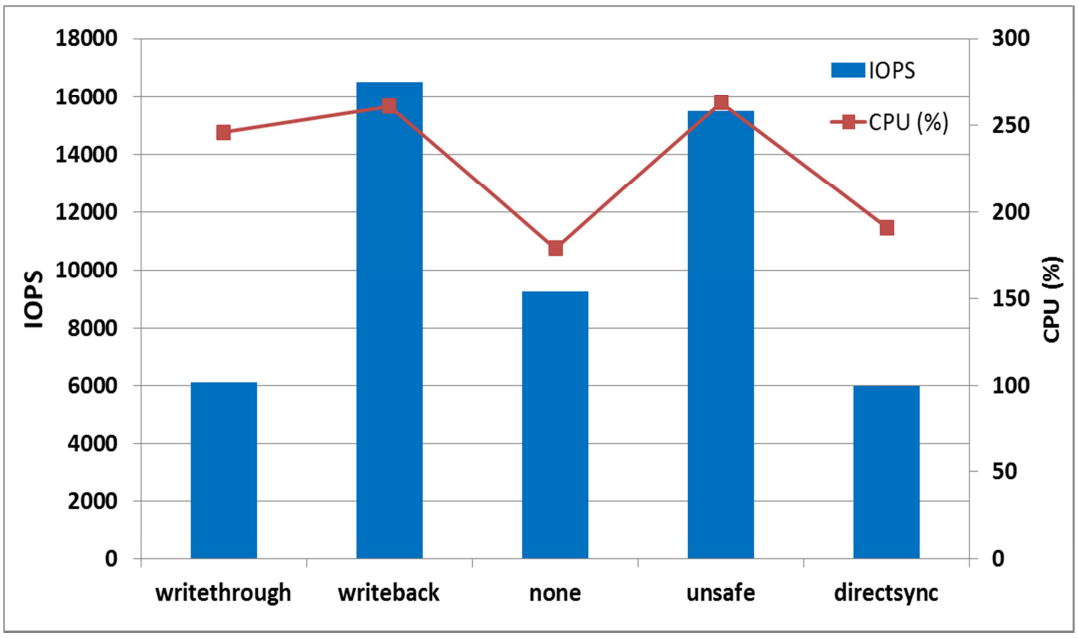


Figure 26 Performance comparison of different cache modes

It is recommended to use Cache=*directsync* while measuring storage performance. Because with directsync no cache is utilized, thus it helps to get down the real performance of a storage system.

5.5 KVM para-virtualized drivers for block device

KVM /QEMU provide full virtualization which is already explained in chapter 4. It has some limitations with I/O performances. The I/O performance is very much limited when compare to native performance, since it is using the emulated hardware. But KVM supports many para-virtualization drivers to improve the virtual I/O performance. Para-virtualized drivers allow running the I/O operation directly on the real device. Virtio is the framework for I/O para-virtualization in KVM. The two important para-virtualized drivers supported by KVM, which influences the block I/O performance are

- Virtio-blk
- Virtio-SCSI

5.5.1 Virtio-blk

Virtio–blk is a para-virtualized I/O driver for the block device. This para-virtualized driver gives better performance than executing I/O operation on an emulated hardware.

Even though it provides better performance it has some limitations, they are described below.

Limited scalability: Virtio-blk puts a strong limitation on the number of block storage devices that can be added to a guest. Currently virtio-blk supports approximately 30 disks per guest only. When the hardware is virtualized and presented to VM, it inherits the same characteristics of hardware as like it behaves in the host. As like in host the PCI device has up to 32 slots. Peripheral Component Interconnect (***PCI***), as its name implies is a standard that describes how to connect the peripheral components of a system together in a structured and controlled way. A ***bus*** is a communication system that transfers data between components inside a computer, or between computers. Each device can be a multifunction board with a maximum of eight separate functions, without getting affected by the other function in the same slot. Virtio presents each disk and network card as a separate function. Whenever devices are added or removed from a running machine, all functions in the same slot have to be added or removed simultaneously. Thus due to this limitation, each PCI slot will usually hold a single virtio device and limiting the KVM guest to 28 virtio devices. The remaining four slots are reserved for various pieces of virtual hardware.

Limitation in adding features: Virtio-blk defines its own command sets for I/O operations like read, write etc. In case of adding new features it needs modification in both guest and the host operating system.

SCSI pass-through: SCSI command from guest reaches to storage only when it is attached as LUN, but cannot support file or disk storage. The guest sees the attached LUN as /dev/vd. Sometime programs may refuse to send SCSI commands to device which varies from host device naming. In host the device name starts as /dev/sd. It is represented in the *Figure 27* [40].

Most of the limitations of virtio-blk are addressed by the virtio-scsi driver.

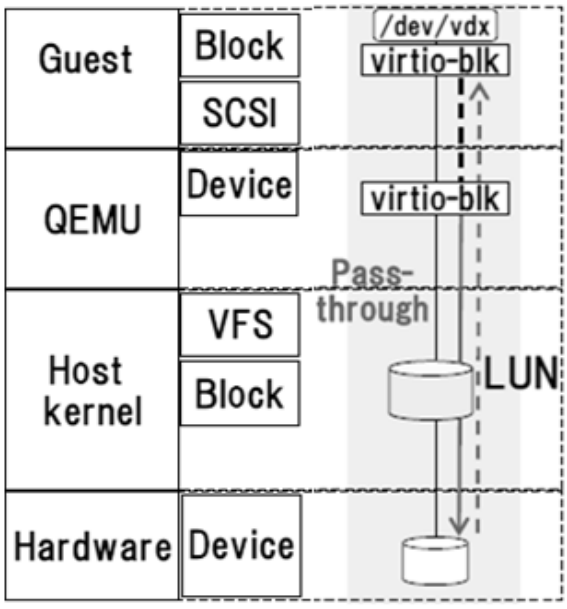


Figure 27 Storage access by virtio-blk

5.5.2 Virtio-SCSI

The virtio-SCSI is a new feature of the KVM. It is a storage interface for the virtual machine. The virtio-scsi is high performance para-virtualized storage device. Virtio-scsi provides anything that the underlying SCSI target supports. It is a virtual small computer system interface (SCSI) host bus adopter and it is the successor of the virtio-blk, with improved capabilities. SCSI is standard electronic interfaces that allow personal computers to communicate with peripheral hardware such as disk drives, tape drives etc. Virtio-SCSI provides the ability to connect directly to SCSI LUNs and significantly improves scalability compared to virtio-blk. It allows accessing multiple storage devices through a single controller, and enabling reuse of the guest operating system's SCSI stack.

The benefits are

- Improved scalability
- Standard command set
- Standard device naming
- SCSI device pass-through—virtio-scsi can present physical storage devices directly to guests.

Improved scalability: Virtio-scsi has capability to connect to multiple storage devices and presenting it to the virtual machines. Virtio-scsi accomplishes this by multiplexing numerous storage devices on a single controller. Each device on a virtio-scsi controller is represented as a logical unit, or LUN. The LUNs are grouped into targets. The limit of each virtio-scsi device is 256 targets maximum per controller and 16,384 logical units per target.

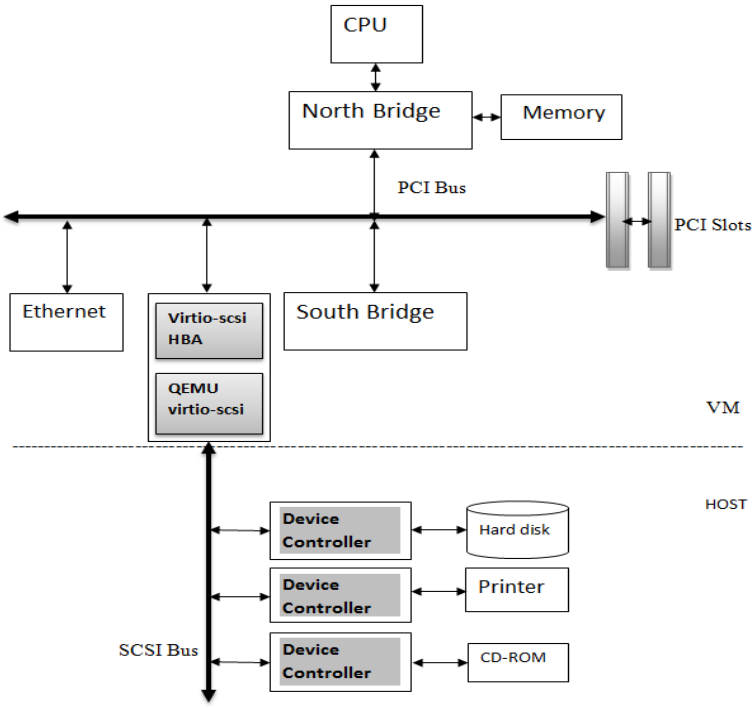


Figure 28 Virtio-SCSI

Standard command set: Virtio-scsi uses standard SCSI command sets. Its specification does not define commands like read, write for disks. Instead, it defines a transport protocol for the commands. Since it acts as a transport protocol, host operating system does not require any modification. If there is a modification in the virtio-scsi module, only the guest operating system required modification. Thus it will be added or the existing virtio-scsi should get updated only in the guest operating system.

Standard device naming: virtio-scsi disks use the same paths as a bare-metal system. This simplifies physical-to-virtual and virtual-to-virtual migration. But virtio-blk devices are represented by the guests with files whose names start with /dev/vd which is different than the host device name. The virtio-scsi devices are represented by /dev/sd as like in host.

SCSI device pass-through**:** For virtual disks that are backed by a whole LUN in the host, it can be desirable to let the guest send SCSI commands directly to the LUN. This is known as pass-through, which is represented in *Figure 29*. Disks are presented to the guest on a SCSI bus. And the guest sees the attached LUN as /dev/sd as like in host. It natively accepts the SCSI command set. Disks are presented to the guest on a SCSI bus as shown in *Figure 28* and the virtual machine definition script are included in the appendix [4] .

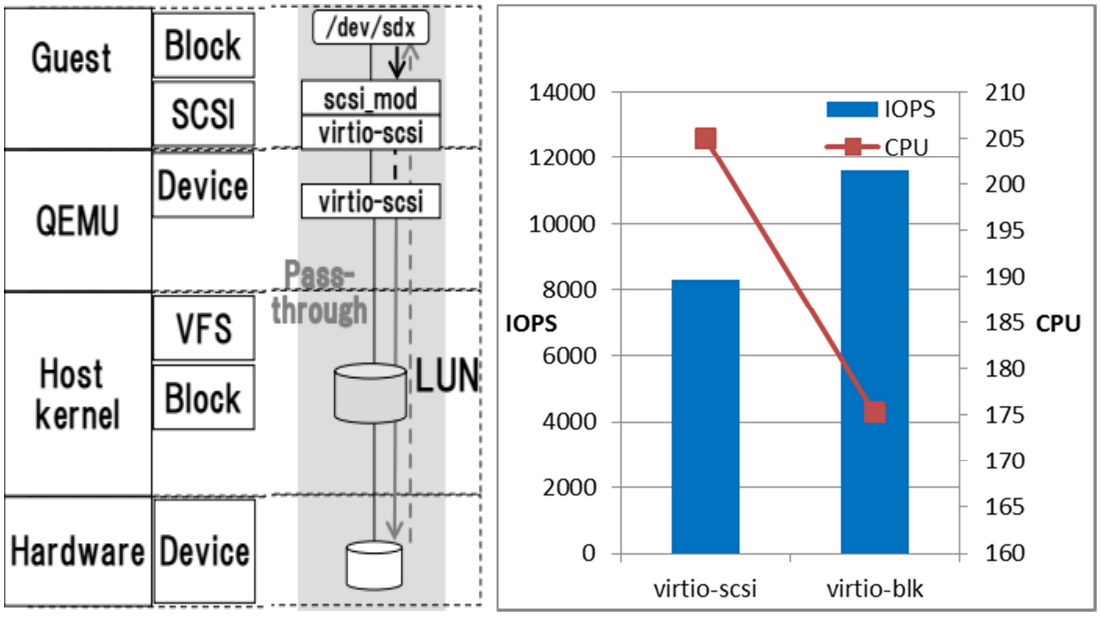


Figure 29 Virtio-scsi passthrough and performance comparison with virtio-blk

5.6 Libiscsi

Libiscsi is a client-side library to implement the iSCSI protocol that can be used to access the resources of an iSCSI target. It is a user space initiator, provides support only for iSCSI targets. By configuring the libiscsi into the QEMU, the iSCSI-initiator can run on the QEMU. So it accesses the iSCSI targets directly by bypassing the host. Thus it

enables hypervisor to access the iSCSI targets directly and this can be employed in a scenario where the host must not see the storage devices of the virtual machine. Virtio-scsi has support for the libiscsi. It is show in *Figure 30* [40].

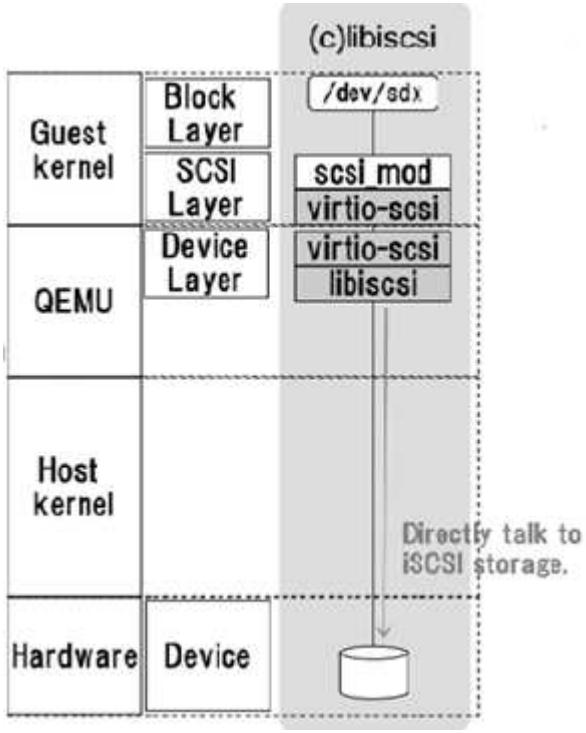


Figure 30 Libiscsi iscsi initiator

6. CONCLUSION

The purpose of this thesis is to study the different possibilities of designing a storage area network (SAN) and to provide an optimized SAN solution. The different protocols for storage area network such as iSCSI, SCSI, FC, FCIP and FCoE were studied. From the study, iSCSI is recommended as the best suitable protocol for the storage area network. Hence iSCSI is employed to enable communication between storage server and client in this SAN design. And it is also concluded that storage server is a virtual machine for the following reasons. The main reasons are efficient utilization of hardware, scalability of storage, replication and possibility of live migration. Virtualization is achieved by the hypervisor KVM/QEMU.

KVM/QEMU has been selected as hypervisor because the KVM is included in the kernel of the operating system LINUX. Hence it is more reliable and it also minimizes the effort and time required for the configuration. There is also requirement to acquire insight knowledge about working means of KVM/QEMU. Thus the working means of KVM/QEMU were studied. Since the storage server is a virtual machine, virtual network has been employed to enable communication between virtual machine and others. There are many options like Bridge, Macvtap, and Openvswitch for deploying virtual network. The benefits and drawbacks of each of them were analyzed and Macvtap is the final choice for this virtual machine network configuration because of its performance and easy configuration. The iSCSI target is running on the virtual machine. There are different iSCSI targets like LIO, SCST, STGT and IET are available in market. The performance of each of them was measured. Among all LIO is the best performing and reliable target as per this environment. With these basic configurations the storage server is ready and the clients can start utilizing the storage space offered by the storage server. But the performance can be still improved.

In order to improve the performance, various parameters have been taken into account. It includes network level optimization, para-virtualized drivers of KVM and cache. Network level optimization includes Multipathing, MC/S, NIC teaming. All of these technologies are mainly used to improve the performance and redundancy. From the study, Multipathing is concluded as the best option when compared to other options like NIC teaming and MC/S. Multipathing can be configured in two ways active/active and active/passive. Active/active multipathing is preferred than active/passive for its better performance since the I/O is spread over all paths. The para-virtualized driver is employed to increase the I/O performance in a full virtualization scenario. The para-virtualized drivers of KVM include virtio-scsi and virtio-blk. Virtio-scsi is employed in this design even though the performance of the virtio-blk is high. This is because virtio-scsi has rich features than virtio-blk and it is a new driver of KVM, expected to grow

more in the future. The cache setting of the virtual machine also has high impact on the storage I/O performance. The cache setting for this SAN solution is direct sync. Direct sync mode does not use any cache. Thus data integrity is secured even in case of any unexpected system failure. It is recommended to practice directsync cache mode while measuring the performance of SAN, which helps to get the real performance of the storage device and SAN configuration without any influence of cache. Cache mode of writeback or writethrough can be used to increase storage performance by taking the risk of data integrity. With these configurations the virtual storage server was optimized such that there is betterment in the crucial parameters like efficiency, performance and redundancy.

There are some limitations in this study. The performance analysis of storage server was conducted in Ericsson specific equipment, it cannot be generalized. Due to the time limit, the SAN solution was not integrated to the openstack for the cloud services. The future study based on this thesis can be integrating it with openstack to provide cloud services. An in-depth study can also be conducted on the para-virtualized drivers of KVM in code level to improve the storage performance.

REFERENCES

[1] Dan Kusnetzky, Virtualization: A Manager's Guide, Big picture of the Who, What and Where of virtualization, O'REILLY 2011. Availability:http://it-ebooks.info/read/583/

[2] Bernard Golden, Virtualization for Dummies, Wiley publishing, Inc. Availability:http://it-ebooks.info/read/2777/

[3] The Most Complete and Integrated Virtualization: From Desktop to Datacenter, An Oracle White Paper, October 2010. Availablility:http://www.oracle.com/us/technologies/virtualization/virtualization - strategy-wp-183617.pdf

[4] Jean S. Bozman Gary P. Chen, Optimizing Hardware for x86 Server Virtualization,WhitePaper, August 2009. Availability:http://www.intel.com/Assets/PDF/whitepaper/IDC choosingvirthardware.pdf

[5] Understanding Full Virtualization, Paravirtualization, and hardware Assist, White Paper, Availability:http://www.vmware.com/files/pdf/ VMware_paravirtualization.pdf

[6] Modeling and Performance Evaluation of iSCSI Storage Area Networks over TCP/IP- based MAN and WAN networks, C. M.Gauger, M. Kohn, S. Gunreben, D. Sass, and S. Gil Perez.

[7] iSCSI Technical White Paper, Nishan Systems. Availability:http://www.diskdrive.com/iSCSI/reading-room/white-papers/Nishan_iSCSI_Technical_White_Paper.pdf

[8] Storage Protocol Comparison, White Paper, vmware. Availability:http://www.vmware.com/files/pdf/techpaper/Storage_Protocol_ Comparison.pdf

[9] iSCSI Protocol Concepts and Implementation, White Paper, Cisco Systems. Availability:http://storusint.com/storage_protocols/iscsi/iSCSI%20White%20 paper.pdf

[10] Storage area networking protocols and architecture, Availability:http://www.cisco.com/networkers/nw04/presos/tech/docs/OPT-2T01.pdf

[11] John L Hufferd, Consultant, IP Storage area Protocols: iSCSI, Hufferd Enterprises, SNIA. Availability:http://www.snia.org/sites/default/education/tutorials/2011/spring/networking/HufferdJohn-IP_Storage_Protocols-iSCSI.pdf

[12] Internet Small Computer Systems Interface (iSCSI) Naming and Discovery, Availability:http://tools.ietf.org/html/rfc3721

[13] Fibre Channel (FC). Availability: http://tools.ietf.org/html/rfc3643

[14] Unified Fabric White Paper—Fibre Channel over Ethernet (FCoE) Availability:http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/UF_FCoE_final.html

[15] Fibre Channel over Ethernet: Enabling Server I/O Consolidation, White Paper Availability:http://www.brocade.com/downloads/documents/white_papers/Brocade_FCoE_WP-00.pdf

[16] Fibre Channel over TCP/IP (FCIP) Availability:http://www.ietf.org/rfc/rfc3821.txt

[17] Securing Block Storage Protocols over IP Availability:http://tools.ietf.org/html/rfc3723

[18] LIO Architecture Availability:http://linux-iscsi.org/wiki/LIO

[19] Comparing File (NAS) and Block (SAN) storage Availability:https://www.spectralogic.com/index.cfm?fuseaction=home.displayFile&DocID=4630

[20] Randy H. Katz, Network-Attached Storage Systems, IEEE paper Availability:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=232686

[21] Network-attached storage for growing small business Availability:https://education.emc.com/academicalliance/documents/EAA_Content/Exercises/IOMEGA%20NAS%20for%20SMB.pdf

[22] Cluster Technology and File Systems
Availability:http://www.stalker.com/notes/SFS.html

[23] KVM – KERNEL BASED VIRTUAL MACHINE
Availability:http://www.redhat.com/en/files/resources/en-rh-kvm-kernal-based-virtual-machine.pdf

[24] Jan Kiszka, Architecture of the Kernel-based Virtual Machine (KVM),Siemens
Availability:http://www.linux-kongress.org/2010/slides/KVM-Architecture-LK2010.pdf

[25] Paolo Bonzini, Effective multi-threading in QEMU, Red Hat
Availability:http://www.linux-kvm.org/wiki/images/1/17/Kvm-forum-2013-Effective-multithreading-in-QEMU.pdf

[26] Anthony Liguori, Multi-threading in QEMU
Availability:http://www.linux-kvm.org/wiki/images/7/70/2010-forum-threading-qemu.pdf

[28] Toshiaki Makita,Virtual switching technologies and Linux bridge, NTT Open Source Software Center.
Availability:http://events.linuxfoundation.org/sites/events/files/slides/LinuxConJapan2014_makita_0.pdf

[29] Linux I/O (LIO)
Availability:http://linux-iscsi.org/wiki/Main_Page

[30] iSCSI Error Handling and Recovery
Availability:http://tools.ietf.org/html/rfc3720#section-6.1.4.4

[31] Combining the Reliability of iSCSI with Full Error Recovery and the Performance of 10 Gigabit Ethernet.
Availability:http://www.force10networks.com/whitepapers/pdf/wp_iscsi_10ge.pdf

[32] Martin K. Petersen, T10 Data Integrity Feature
Availability:https://www.usenix.org/legacy/event/lsf07/tech/petersen.pdf

[33] Dr. Khoa Huynh, Exploiting The Latest KVM Features For Optimized Virtualized Enterprise Storage Performance, IBM Linux Technology Center.
Availability:http://events.linuxfoundation.org/sites/events/files/slides/

CloudOpen2013_Khoa_Huynh_v3.pdf

[34] The Linux Kernel's VFS Layer
Availability:https://www.usenix.org/legacy/event/usenix01/full_papers/kroeger/kroeger_html/node8.html

[35] The Linux Virtual File system
Availability:http://www.inf.fu-berlin.de/lehre/SS01/OS/Lectures/Lecture16.pdf

[36] Best Practices for Running VMware vSphere on iSCSI, TECHNICAL MARKETING DOCUMENTATION, July 2013
Availability:http://www.vmware.com/files/pdf/iSCSI_design_deploy.pdf

[37] Link Aggregation, IEEE.
Availability:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4668665

[38] NIC TEAMING, IEEE 802.3ad, vmware.
Availability:http://www.vmware.com/pdf/esx2_NIC_Teaming.pdf

[39] Asias He , Virtio-blk Performance Improvement, Red Hat, KVM FORUM Nov 2012. Availability:http://www.linux-kvm.org/wiki/images/f/f9/2012-forum-virtio-blk-performance-improvement.pdf

[40] Masaki Kimura ,Better Utilization of Storage Features from KVM Guest via virtio-scsi.
Availability:http://events.linuxfoundation.org/sites/events/files/slides/MasakiKimura_LinuxConNorthAmerica2013_1.pdf

[41] Virtio SCSI, An alternative virtualized storage stack for KVM.
Availability:http://www.linux-kvm.org/wiki/images/f/f5/2011-forum-virtio-scsi.pdf

[42] Laura Novich, Paolo Bonzini, virtio-scsi, Technical white paper, Red Hat Israel.

[43] Multithreading
Availability:http://www.tutorialspoint.com/operating_system/os_multi_threading.htm

APPENDIX

[1] A script to define virtual machine

```
/usr/bin/qemu-system-x86_64 \
-enable-kvm \
-cpu phenom \
-smp 1,sockets=1,cores=1,threads=1 \
-drive file=/home/ubuntu.img,if=virtio,cache=directsync \
-drive file=/dev/sdb1,if=virtio,cache=directsync  \
-drive file=/dev/sdc2,if=virtio,cache=directsync  \
-netdev tap,id=tap2,fd=58 58<>/dev/tap58 \
-device virtio-net-pci,netdev=tap2,mac=c6:af:77:89:bc:39 \
-netdev tap,id=tap3,fd=62 62<>/dev/tap62 \
-device virtio-net-pci,netdev=tap3,mac=d2:23:b9:bb:93:00  \
-netdev tap,id=tap4,fd=61 61<>/dev/tap61 \
-device virtio-net-pci,netdev=tap4,mac=b6:a2:c9:c2:59:a9 \
-boot c \
-m 3000  \
-vnc :3&
```

enable-kvm

Enable the kvm features when needed, otherwise use QEMU.

cpu cpu_model

Specify type of the processor (CPU) model.

smp number_of_cpus

Specifies how many CPUs will be emulated. This option also takes other CPU-related parameters, such as number of sockets, number of cores per socket, or number of threads per core.

file=image_fname

Specify the path of the disk image which will be used by the drive.

if=drive_interface

Specifies the type of interface to which the drive is connected. Currently only floppy, ide, or virtio are supported by SUSE. Virtio defines a para-virtualized disk driver.

If your device, such as -drive, needs a special driver and driver properties to be set, specify them with the -device option, and identify with drive= suboption.

cache=method

Specify the caching method for the drive. Possible values are unsafe, writethrough, writeback, directsync, or none.

netdev

This option defines a network interface and a specific type of networking for your VM.

tap

Specify a bridged or routed networking.

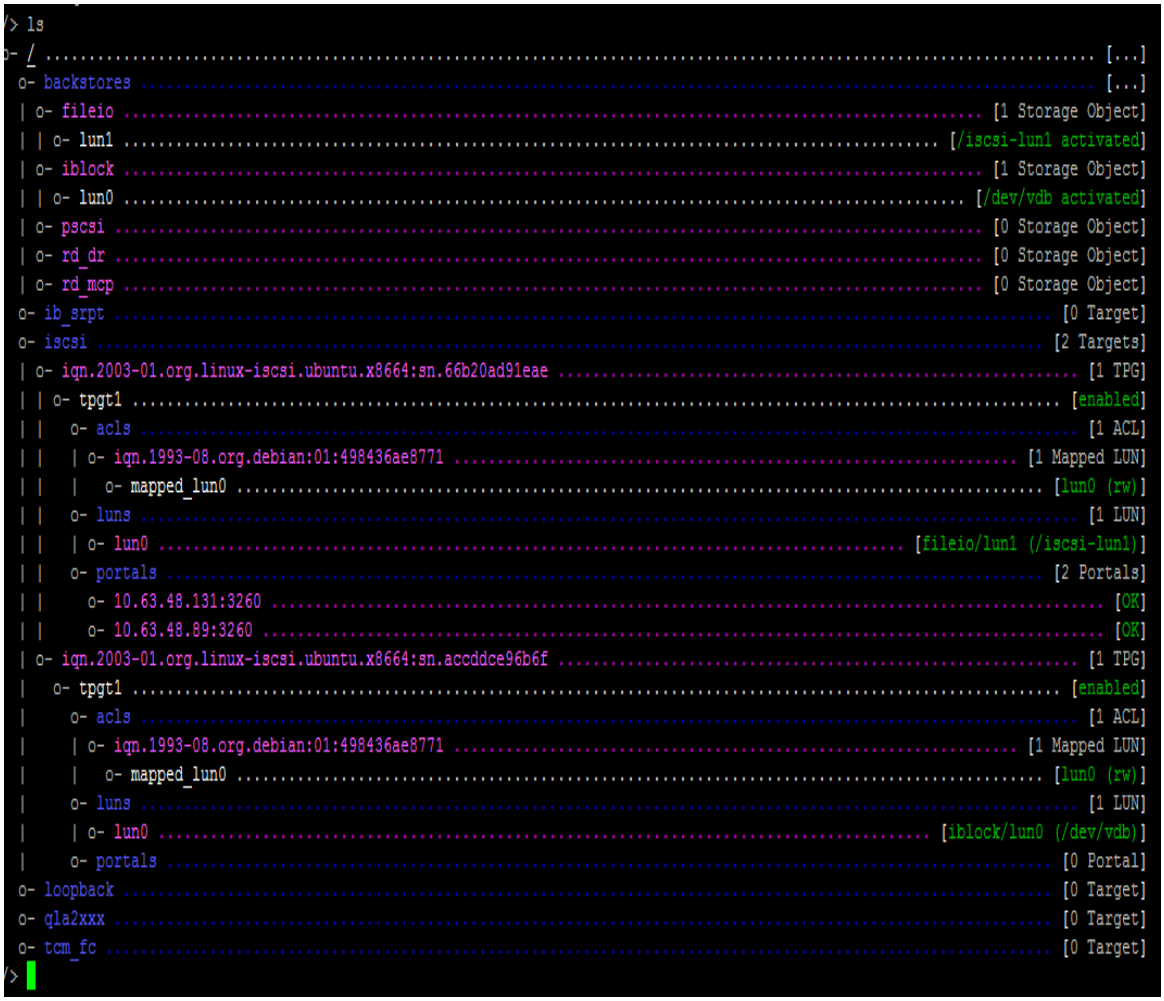
fd

file descriptor

boot

Specifies the order in which the defined drives will be booted. Drives are represented by letters, where 'a' and 'b' stands for the floppy drives 1 and 2, 'c' stands for the first hard disk, 'd' stands for the first CD-ROM drive, and 'n' to 'p' stand for Ether-boot network adapters.

[2] LIO Configuration



[3] Multipath configuration

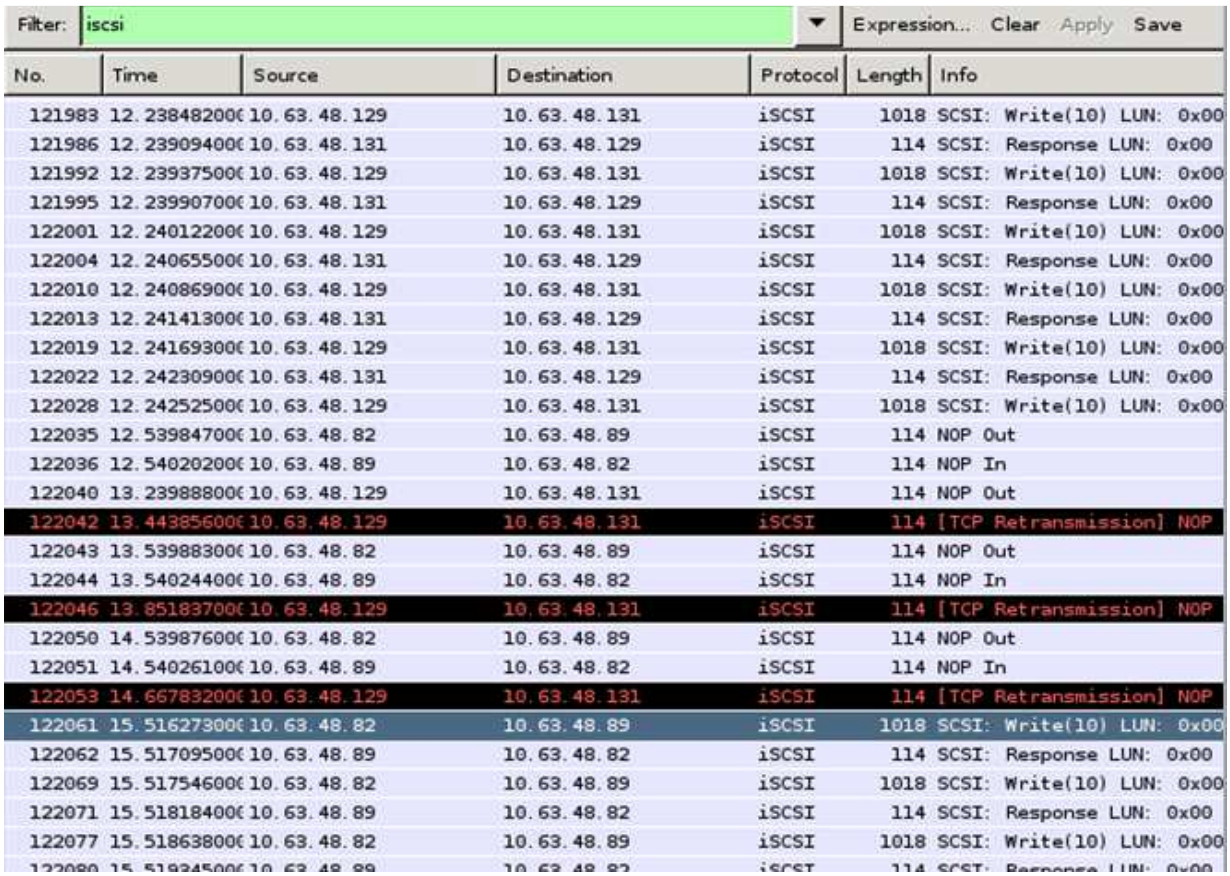
```
defaults {
user_friendly_names yes
# Use mpathn names for multipath devices
path_grouping_policy multibus
# Place all paths in one priority group
path_checker readsector0
# Method to determine the state of a path
polling_interval 3
# How often (in seconds) to poll state of paths
path_selector "round-robin 0"
# Algorithm to determine what path to use for next I/O operation
rr_min_io 1000  #(default)
# The number of I/O requests to route to a path before switching to the next path
failback immediate
# Failback to highest priority path group with active paths
no_path_retry 0
#Number of times the system should attempt to use a failed path before disabling queueing

}

blacklist {
devnode "^sd[a-d]$"
}
multipaths {
multipath {
wwid 360014057bbf7733f4c54da98a0a5757d
}
}
```

[4] VM script with virtio-drivers

```
/usr/bin/qemu-system-x86_64 \
-enable-kvm \
-cpu phenom \
-smp 1,sockets=1,cores=1,threads=1 \
-drive id=hd0,file=/home/vm1.img,if=none,cache=directsync \
-device virtio-scsi-pci \
-device scsi-hd,drive=hd0 \
-drive id=hd1,file=/dev/sdb1,if=none,cache=directsync  \
```

```
-device virtio-scsi-pci \
-device scsi-hd,drive=hd1 \
-drive id=hd2,file=/dev/sdc2,if=none,cache=directsync \
-device virtio-scsi-pci \
-device scsi-hd,drive=hd2 \
-netdev tap,id=tap6,fd=58 58<>/dev/tap58 \
-device virtio-net-pci,netdev=tap6,mac=c6:af:77:89:bc:39 \
-netdev tap,id=tap7,fd=61 61<>/dev/tap61 \
-device virtio-net-pci,netdev=tap7,mac=b6:a2:c9:c2:59:a9  \
-netdev tap,id=tap8,fd=62 62<>/dev/tap62 \
-device virtio-net-pci,netdev=tap8,mac=d2:23:b9:bb:93:00 \
-boot c \
-m 3000  \
-vnc :5&
```

[5] Failover time calculation with wireshark



Filter: iscsi — Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
121983	12.23848200(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
121986	12.23909400(	10.63.48.131	10.63.48.129	iSCSI	114	SCSI: Response LUN: 0x00
121992	12.23937500(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
121995	12.23990700(	10.63.48.131	10.63.48.129	iSCSI	114	SCSI: Response LUN: 0x00
122001	12.24012200(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122004	12.24065500(	10.63.48.131	10.63.48.129	iSCSI	114	SCSI: Response LUN: 0x00
122010	12.24086900(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122013	12.24141300(	10.63.48.131	10.63.48.129	iSCSI	114	SCSI: Response LUN: 0x00
122019	12.24169300(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122022	12.24230900(	10.63.48.131	10.63.48.129	iSCSI	114	SCSI: Response LUN: 0x00
122028	12.24252500(	10.63.48.129	10.63.48.131	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122035	12.53984700(	10.63.48.82	10.63.48.89	iSCSI	114	NOP Out
122036	12.54020200(	10.63.48.89	10.63.48.82	iSCSI	114	NOP In
122040	13.23988800(	10.63.48.129	10.63.48.131	iSCSI	114	NOP Out
122042	13.44385600(	10.63.48.129	10.63.48.131	iSCSI	114	[TCP Retransmission] NOP
122043	13.53988300(	10.63.48.82	10.63.48.89	iSCSI	114	NOP Out
122044	13.54024400(	10.63.48.89	10.63.48.82	iSCSI	114	NOP In
122046	13.85183700(	10.63.48.129	10.63.48.131	iSCSI	114	[TCP Retransmission] NOP
122050	14.53987600(	10.63.48.82	10.63.48.89	iSCSI	114	NOP Out
122051	14.54026100(	10.63.48.89	10.63.48.82	iSCSI	114	NOP In
122053	14.66783200(	10.63.48.129	10.63.48.131	iSCSI	114	[TCP Retransmission] NOP
122061	15.51627300(	10.63.48.82	10.63.48.89	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122062	15.51709500(	10.63.48.89	10.63.48.82	iSCSI	114	SCSI: Response LUN: 0x00
122069	15.51754600(	10.63.48.82	10.63.48.89	iSCSI	1018	SCSI: Write(10) LUN: 0x00
122071	15.51818400(	10.63.48.89	10.63.48.82	iSCSI	114	SCSI: Response LUN: 0x00
122077	15.51863800(	10.63.48.82	10.63.48.89	iSCSI	1018	SCSI: Write(10) LUN: 0x00