VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum – 590 018

A PROJECT REPORT ON

"Implementing a backend tool to share disc images among Virtual Machines"

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

INFORMATION SCIENCE & ENGINEERING

by

SHOBHIT KUSHWAHA (1CR16IS101)
SHREE CHARAN B G (1CR16IS102)
SUDHA KUMARI (1CR16IS110)
MARUTHI C N (1CR16IS051)

Under the guidance of

Mrs. Vidya U
Assistant Professor
Dept. of ISE, CMRIT, Bengaluru

CMR INSTITUTE OF TECHNOLOGY
DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
#132, AECS Layout, IT Park Road, Bengaluru-560037

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum – 590 018

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Certificate

This is to certify that the project entitled, **“Implementing a backend tool to share disc images among Virtual Machines”**, is a bonafide work carried out by **SHOBHIT KUSHWAHA (1CR16IS101), SHREE CHARAN B G (1CR16IS102), SUDHA KUMARI (1CR16IS110) and MARUTHI C N (1CR16IS051)** in partial fulfillment of the award of the degree of Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2019-20. It is certified that all corrections/suggestions indicated during reviews have been incorporated in the report. The project report satisfies the academic requirements in respect of the project work prescribed for the said Degree.

Name & Signature of Guide	**Name & Signature of HOD**
Mrs. Vidya U	**Dr. M. Farida Begam**
Asst. Professor	**HoD**
Dept. of ISE, CMRIT	**Dept. of ISE, CMRIT**

External Viva

Name of the examiners	Signature with date
1.	
2.	

CMR INSTITUTE OF TECHNOLOGY
BANGALORE-560037

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Declaration

We, **SHOBHIT KUSHWAHA (1CR16IS101), SHREE CHARAN B G (1CR16IS102), SUDHA KUMARI (1CR16IS110) and MARUTHI C N (1CR16IS052)** bonafide students of CMR Institute of Technology, Bangalore, hereby declare that the dissertation entitled, **"Implementing a backend tool to share disc images among Virtual Machines"** has been carried out by us under the guidance of Mrs. Vidya U, Associate Professor, CMRIT, Bangalore, in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Science Engineering, of the Visvesvaraya Technological University, Belgaum during the academic year 2019-2020. The work done in this dissertation report is original and it has not been submitted for any other degree in any university.

SHOBHIT KUSHWAHA (1CR16IS101)
SHREE CHARAN B G (1CR16IS102)
SUDHA KUMARI (1CR16IS110)
MARUTHI C N (1CR16IS051)

Acknowledgement

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.
We would like to thank Dr. Sanjay Jain, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

We would like to express our gratitude towards Dr. M. Farida Begam, Professor and HOD, Department of Information Science & Engineering CMRIT, Bangalore, who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honor to express our sincere gratitude to our internal guide Mrs. Vidya U, Asst. Professor, Department of Information Science & Engineering, CMRIT, Bangalore, for their valuable guidance throughout the tenure of this project work.

We would like to thank all the faculty members who have always been very cooperative and generous. Conclusively, we also thank all the non- teaching staff and all others who have done immense help directly or indirectly during our project.

Shobhit Kushwaha
Shree Charan B G
Sudha Kumari
Maruthi C N

ABSTRACT

The aim of the project is to develop a virtual host-user-block device backend inside QEMU (Quick Emulator) so that guests can efficiently access shared disk images.

QEMU can connect virtio-block disks to external processes that act as vhost-user-block device backend. This makes it possible for QEMU guests to access disks managed by SPDK (Storage Performance Development Kit) or other software-defined storage appliances.

QEMU itself does not offer a vhost-user-block device backend and hence depends on other applications such as SPDK for providing vhost-user backends. However, the QEMU block layer has a number of features that make QEMU desirable as a software-defined storage appliance in its own right. For example, multiple Virtual Machines could safely access a shared qcow2 disk image file with one of the QEMUs acting as the vhost-user-block device backend. Today this can be worked around using QEMU's NBD (Network Block Device) support, but its performance is low since it is network protocol.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Preamble

1.1 Introduction

QEMU is the most popular open source virtual machine (VM) emulator application. QEMU uses Kernel Virtual Machine (KVM) driver in Linux to accelerate VM performance using CPU's virtualization extensions. QEMU can use virtio framework to emulate virtual I/O devices such as Disk and NIC for the VMs. For example, virtio-blk is used by QEMU to provide block devices (i.e., disk) to VMs. Normally QEMU is responsible for emulating VM's I/O devices.

vhost is a protocol for devices accessible via inter-process communication. vhost is used by QEMU to optimize performance of virtio devices or offload virtio device functionality to other components (either in the kernel or a separate application).

QEMU can connect virtio-blk disks to external processes that act as vhost-user-blk device backends. This makes it possible for QEMU VMs to access disks managed by SPDK or other software-defined storage appliances. QEMU itself does not offer a vhost-user-blk device backend and hence depends on other applications like SPDK (Storage Performance Development Kit) for providing vhost-user backends. However, the QEMU block layer has a number of features that make QEMU desirable as a software-defined storage appliance in its own right. For example, multiple VMs could safely access a shared qcow2 disk image file with one of the QEMUs acting as the vhostuser-blk device backend. Today this can be worked around using QEMU's NBD (network block device) support, but its performance is low since it is a network protocol.

1.2 Plan of Implementation

The goal is to add a vhost-user-blk device backend to QEMU so that disks can be exported to other processes. Specifically, we want to use the vhost-use-blk backend in QEMU to share a disk with multiple VMs instead of NBD based approach. We will use the following steps:

• Understand libvhost-user which is QEMU's library for implementing vhost-user device backends.

• Implement a vhost-user-blk device backend using the libvhost-user library.

• Use the vhost-user-blk device backend to implement sharing of a disk among multiple VMs.

• Evaluate the storage performance and compare it with NBD based solution.

1.3 Problem Statement

Implement a vhost-user-blk device backend inside QEMU so guests can efficiently access shared disk images.

QEMU itself does not offer a vhost-user-blk device backend although the QEMU block layer has a number of features that make QEMU desirable as a software-defined storage appliance in its own right. For example, multiple VMs could safely access a shared qcow2 disk image file with one of the QEMUs acting as the vhost-user-blk device backend. Today this is can be worked around using QEMU's NBD support, but its performance will always be lower since it is a network protocol.

The goal is to add a vhost-user-blk device backend to QEMU so that disks can be exported to other processes.

1.4 Objective of the Project

• To develop the vhost-user-blk backend in QEMU which will enable QEMU to act as a software defined storage appliance without using third party vhost targets like SPDK.

• To enable the sharing of a disk among VMs using the vhost-user-blk backend while improving the performance compared to the NBD based solution.

• Explore other storage services which can be implemented inside QEMU using the vHost framework (e.g., storage deduplication)

Chapter 2

Literature Survey

In order to get required knowledge about various concepts related to the present application, existing literature were studied. Some of the important conclusions were made through those are listed below.

•Accelerating-VM-Access-with-Storage-Performance-Developer-Kit-SNIA-SDCEMEA-2018 By Anu H Rao (Product Manager) Intel.

It was a seminar given by Anu H Rao at SDC-EMEA (Storage Developer Conference) in that he has explained about the working of different Virtual Machine Internals and about how to accelerate storage performance in a virtual machine.

• https://www.qemu.org/
Official Site and Documentation of QEMU.
• http://linux-kvm.org/
Official Site and Documentation of Linux-Kernel Virtual Machine.
• https://wiki.libvirt.org/page/Virtio virtIO
Official Page and Documentation.
• https://spdk.io/doc/vhost_processing.html
Official SPDK site and explained about how vHost internally work with SPDK.

Chapter 3

Theoretical Background

Theoretical background highlighting some topics related to the project work is given below. The description contains several topics which are worth to discuss and also highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

3.1 Virtualization

In computing, virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

Different types of hardware virtualization include:

- Full virtualization – almost complete simulation of the actual hardware to allow software environments, including a guest operating system and its apps, to run unmodified.
- Paravirtualization – the guest apps are executed in their own isolated domains, as if they are running on a separate system, but a hardware environment is not simulated. Guest programs need to be specifically modified to run in this environment.

3.2 Full Virtualization vs Paravirtualization

Paravirtualization works differently from the full virtualization. It doesn’t need to simulate hardware for the virtual machines. The hypervisor is installed on the physical server (host) and the guest OS is installed into the environment. Virtual guests are aware that it has been virtualized, unlike the full virtualization (where the guest doesn’t know that it has been virtualized) to take advantage of the functions. In this virtualization method, guest source codes will be modified with sensitive information to communicate with the host. Guest Operating Systems require extensions to make API calls to the hypervisor. In full virtualization, guests will issue a hardware call but in paravirtualization, guest will directly communicate with the host (hypervisor) using the drivers.

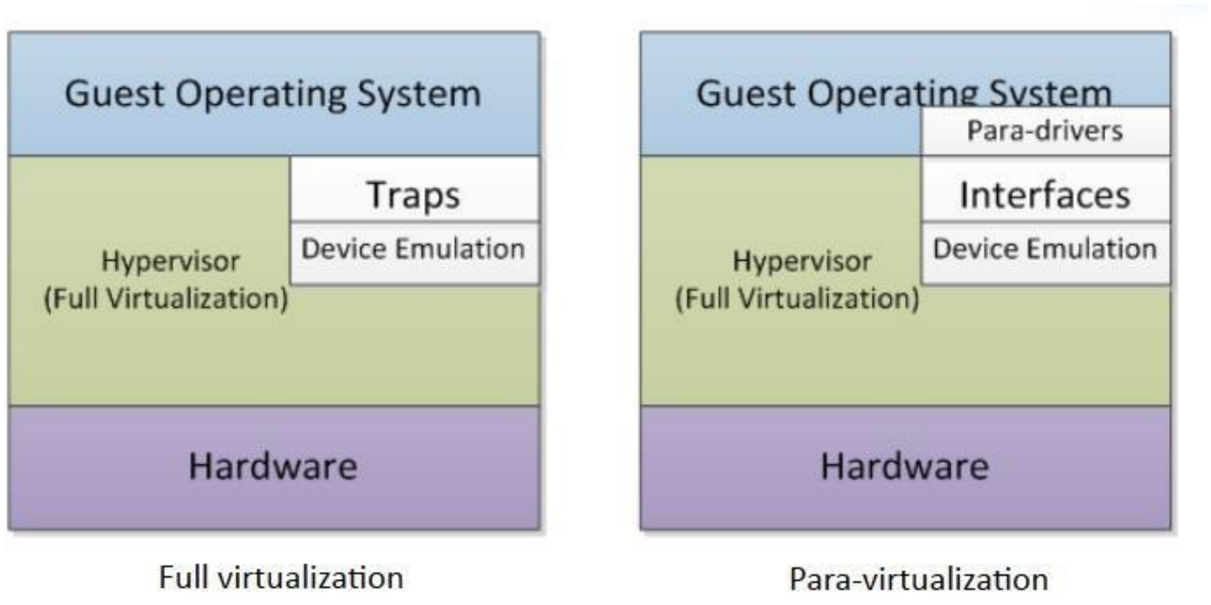


Fig 3.2: Full Virtualization vs Para-virtualization

3.3 Storage Performance Development Kit

The storage Performance Development Kit (SPDK) provides a set of tools and libraries for writing high performance, scalable, user-mode storage applications. It achieves high performance through the use of number of key techniques.

- Moving all the necessary drivers into user space, which avoids system call and enables zero copy access from the application.
- Polling hardware for completions instead of relying on interrupts, which lowers both total latency and latency variance.
- Avoiding all locks in the I/O path, instead of relying on message passing.

3.4 QEMU

QEMU (Quick EMUlator) is a free and open-source emulator that performs hardware virtualization. QEMU is a hosted virtual machine monitor: it emulates the machine's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating system. It also can be used with KVM to run virtual machines at near-native speed (by taking advantage of hardware extensions such as Intel VT-x). QEMU can also do emulation for user-level processes, allowing applications compiled for one architecture to run on another.

3.4.1 Operating Modes

QEMU has multiple Operating Modes:

• User-mode Emulation

In this mode QEMU runs single Linux or Darwin/macOS programs that were compiled for a different instruction set. System calls are thunked for endianness and for 32/64 bit mismatches. Fast cross-compilation and cross-debugging are the main targets for user-mode emulation.

• System Emulation

In this mode QEMU emulates a full computer system, including peripherals. It can be used to provide virtual hosting of several virtual computers on a single computer. QEMU can boot many guests operating systems, including Linux, Solaris, Microsoft Windows, DOS, and

• KVM Hosting

Here QEMU deals with the setting up and migration of KVM images. It is still involved in the emulation of hardware, but the execution of the guest is done by KVM as requested by QEMU.

• **XEN Housing**

QEMU is involved only in the emulation of hardware; the execution of the guest is done within Xen and is totally hidden from QEMU.

3.5 Kernel Virtual Machine

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc. KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The userspace component of KVM is included in mainline QEMU, as of 1.3.

3.6 virtIO

VirtIO is a standardized interface which allows virtual machines access to simplified "virtual" devices, such as block devices, network adapters and consoles. Accessing devices through VirtIO on a guest VM improves performance over more traditional "emulated" devices, as VirtIO devices require only the bare minimum setup and configuration needed to send and receive data, while the host machine handles the majority of the setup and maintenance of the actual physical hardware.

3.6.1 Technical Details

VirtIO devices appear, to the guest VM, to be normal PCI devices with a specific VendorID and DeviceID. All VirtIO devices have a Vendor ID of 0x1AF4, and have a DeviceID between 0x1000 and 0x103F. The type of VirtIO device (Network Adapter, Block Device, etc.) can be determined by the Subsystem ID field in the PCI Configuration Space for the device. The currently defined types are:

Subsystem	Name
01	Network Card
02	Block Device
03	Console
04	Entropy Source
05	Memory Ballooning
06	IO Memory
07	RPMSG
08	SCSI Host
09	9P Transport
10	MAC802.11 WLAN

Fig 3.6.1: Defined subsystem types

3.7 QEMU Server Architecture

• One QEMU application acts as the vHost target (a.k.a. QEMU Server) which provides the shared disk access to other QEMU VMs.

• Server QEMU provides access to the shared disk to its own Virtual Machine.

• Other QEMU VMs (clients) access the shared disk through the QEMU Server using vHost protocol.

Chapter 4

System Requirements Specification

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. In order to fully understand one's project, it is very important that they come up with an SRS listing out their requirements, how are they going to meet it and how will they complete the project. SRS also functions as a blueprint for completing a project with as little cost growth as possible. SRS is often referred to as the parent document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

4.1 System Requirement

Hardware System Configuration

• Processor - Intel i5-6200U @2.30 GHz (As fast as possible to run multiple VMs)

• Memory – 8GB (Preferred 16 GB)

• Disk – 256GB Solid Disk Drive for Faster Experience

Software System Configuration

• Operating System – Linux

• Language – C

• Virtual Machine Framework – KVM/QEMU

• Various Operating System to Run as Guest OS.

4.2 Technology Used

• Virtualisation

• Storage Virtualisation

• SPDK (Storage Performance Development Kit)

• DPDK (Data Plain Development Kit)

Chapter 5

System Analysis

Analysis is the process of finding the best solution to the problem. System analysis is the process by which we learn about the existing problems, define objects and requirements and evaluates the solutions. It is the way of thinking about the organization and the problem it involves, a set of technologies that helps in solving these problems. Feasibility study plays an important role in system analysis which gives the target for design and development.

5.1 Feasibility Study

All systems are feasible when provided with unlimited resource and infinite time. But unfortunately this condition does not prevail in practical world. Soit is both necessary and prudent to evaluate the feasibility of the system at the earliest possible time. Months or years of effort, thousands of rupees and untold professional embarrassment can be averted if an illconceived system is recognized early in the definition phase. Feasibility & risk analysis are related in many ways. If project risk is great, the feasibility of producing quality software is reduced. In this case three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

5.1.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead tohigh demands being placed on the client. Thedeveloped system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.2 Analysis

5.2.1 Performance Analysis

For the complete functionality of the project work, the project is run with the help of healthy networking environment. Performance analysis is done to find out whether the proposed system. It is essential that the process of performance analysis anddefinition must be conducted in parallel.

5.1.2 Technical Analysis

System is only beneficial only if it can be turned into information systems that will meet the organization’s technical requirement. Simply stated this test of feasibility asks whether the system will work or not when developed & installed, whether there are any major barriers to implementation. Regarding all these issues in technical analysis there are several points to focus on:-

Changes to bring in the system: All changes should be in positive direction, there will be increased level of efficiency and better customer service.

Required skills: Platforms & tools used in this project are widely used. So the skilled manpower is readily available in the industry.

Acceptability: The structure of the system is kept feasible enough so that there should not be any problem from the user’s point of view.

5.1.3 Economical Analysis

Economic analysis is performed to evaluate the development cost weighed against the ultimate income or benefits derived from the developed system. For running this system, we need not have any routers which are highly economical. So the system is economically feasible enough.

Chapter-6

System Design

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customer's requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

6.1 System development methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

6.1.1 Model phases

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation,, Analysis, Design, Implementation, Testing and maintenance.

Requirement Analysis: This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.

System Design: Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm**,** data structure**,** software architecture etc.

Coding: In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

Implementation: The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation

Testing: In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.

Maintenance: The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

6.1.2 Advantages of the Waterfall Model

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.

- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

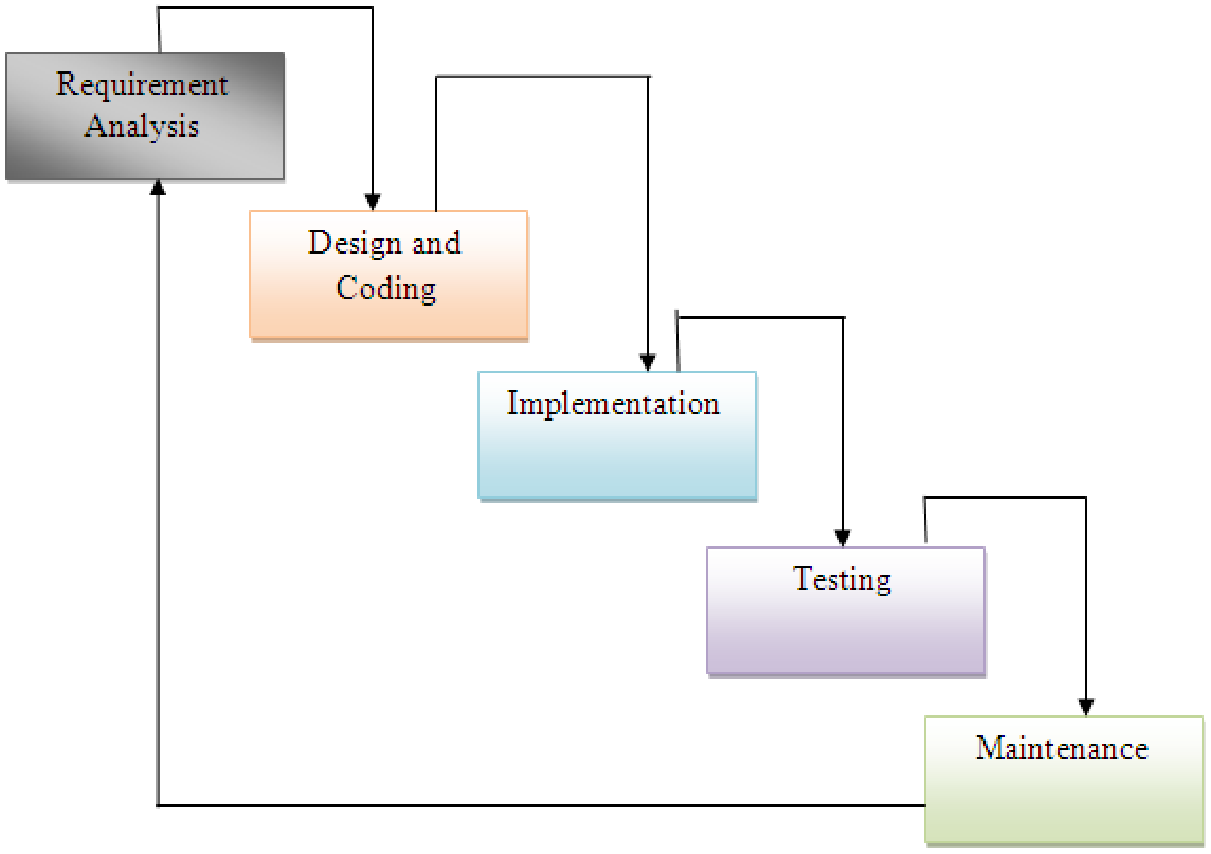
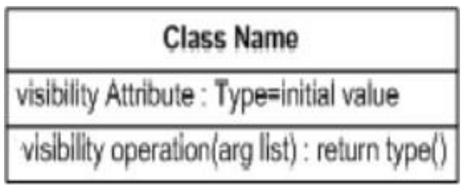
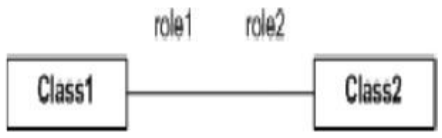
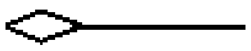


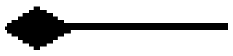
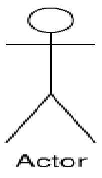
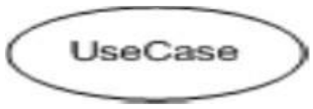
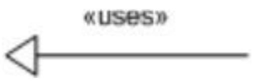
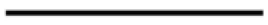
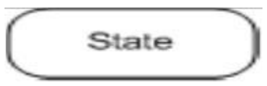
Fig 6.1: Waterfall model

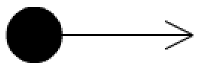
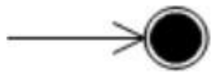
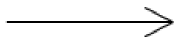
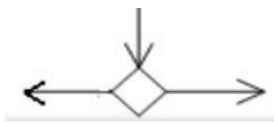
6.2 Design Using UML

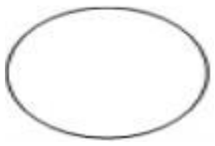
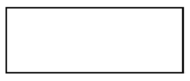
Designing UML diagram specifies, how the process within the system communicates along with how the objects with in the process collaborate using both static as well as dynamic

UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail. Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

Sl. No	Symbol Name	Symbol	Description
1	Class	 Class Name visibility Attribute : Type=initial value visibility operation(arg list) : return type()	Classes represent a collection of similar entities grouped together.
2	Association		Association represents a static relation between classes.
3	Aggregation		Aggregation is a form of association. It aggregates several classes into a single class.

4	Composition		Composition is a special type of aggregation that denotes a strong ownership between classes.
5	Actor		Actor is the user of the system that reacts with the system.
6	Use Case		A use case is an interaction between system and the external environment.
7	Relation (Uses)		It is used for additional purpose communication.
8	Communication		It is the communication between use cases.
9	State		It represents the state of process. Each state goes through various flows.

10	Initial State		It represents initial state of object.
11	Final State		It represents final state of object.
12	Control Flow		It represents decision making process for object.
13	Decision Box		It represents the decision making process from a constraint.

14	Data Process / State		A circle in a DFD represents a state or process which has been triggered due to some other event or action.
15	External Entity		It represents external entity such as Keyboard, sensors, etc which are used in the system.

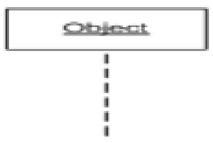
16	Transition		It represents any communication that occurs between processes.
17	Object Lifeline		Object lifeline represents the vertical dimension that object communicates.
18	Message		It represen ts messages exchanged.

Table 6.2: Symbols used in UML

6.3 Sequence Diagram

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

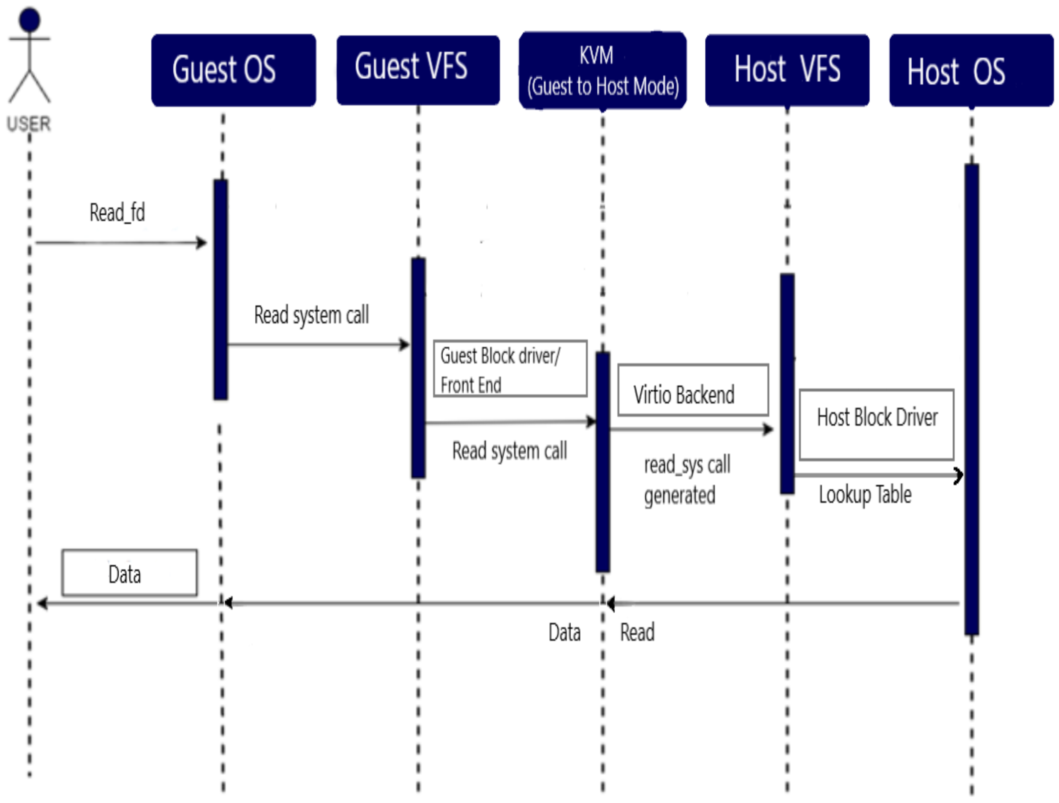


Fig 6.3 Sequence Diagram

The sequence diagram shows the set of events that occur once the input file has been given. The first step is to parse, where the required data will be put into a structure. Then the adjacency matrix is created which will be used when the lookup table is created and also while computing the final sequence. Once the final sequence is generated, the final code is generated which also uses the structure that was created during parsing. The user will get the generated code for the file that he has inputted.

Chapter 7

Implementation

The following are the commands that are required for the working of the project.

7.1 Commands

- **The command to start the QEMU machine:**

```
sudo qemu-system-x86_64 -machine q35,usb=off -m 2048 -name ubuntu16 -drive
file=/var/lib/libvirt/images/ubuntu16.img,format=raw,if=none,id=drive-virtio-disk0-device
virtio-blk-pci,scsi=off,drive=drive-virtio-disk0,id=virtio-disk0-bios
/usr/local/share/qemu/bios.bin -enable-kvm
```

- **This is followed by starting the VNCviewer-**

  ```
  vncviewer 127.0.0.1:port
  ```

- **To start and close the QEMU console**

  ```
  ctrl+alt+2
  ctrl+alt+1
  ```

- **In Server QEMU Console:**

  runinfoblock - to see the name of the disk

  ```
  runvhost_user_server_start /tmp/vhost-user-blk_vhost.socket
  ```

- **Starting the second Virtual Machine in Ubuntu 2**

```
sudo qemu-system-x86_64 -machine q35,usb=off -m 2048 -name ubuntu2 -drive
file=/var/lib/libvirt/images/ubuntu2.img,format=raw,if=none,id=drive-virtio-disk0 -device
```

virtio-blk-pci,scsi=off,drive=drive-virtio-disk0,id=virtio-disk0 -bios /usr/local/share/qemu/bios.bin -enable-kvm -chardevsocket,id=char1,path=/tmp/vhost-user-blk_vhost.socket -device vhost-user-blk-pci,id=blk0,chardev=char1 -object memory-backend-memfd,id=mem,size=2048M,share=on -numanode,memdev=mem

- **To mount the disk**

  sudo mount -o ro,noload /dev/vdb1 /mnt

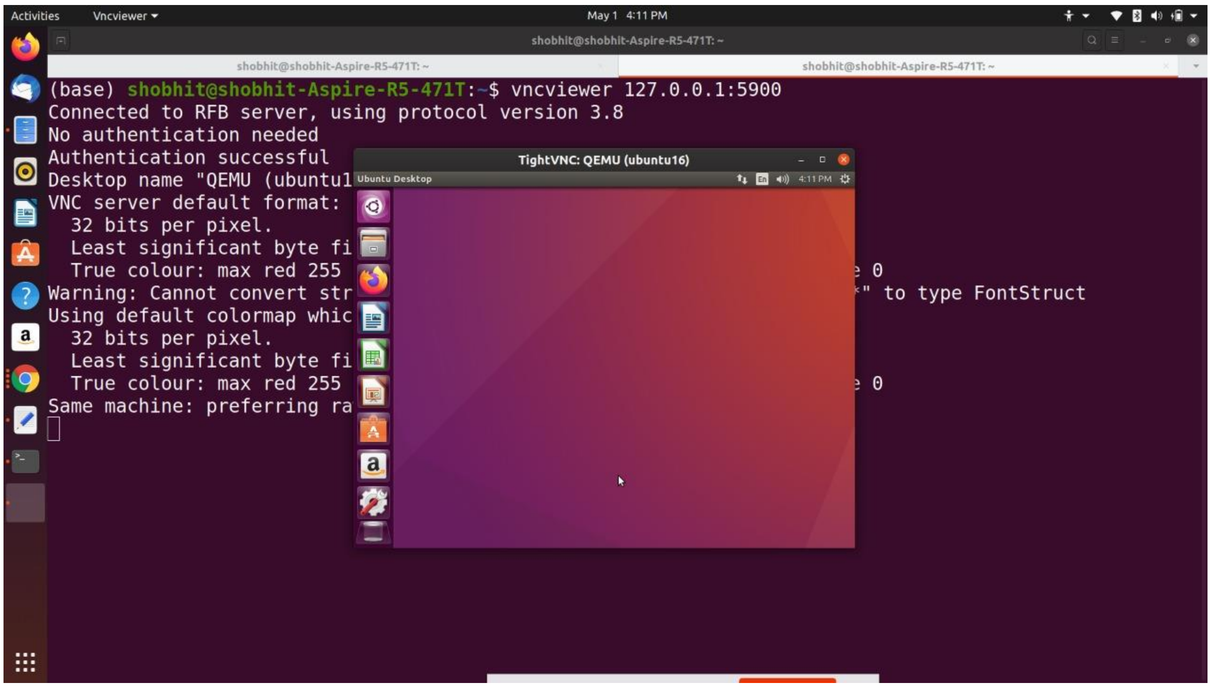
  This will enable us to access the server's file.

Chapter-8

Results and Performance Analysis

In this section, the snapshots shows the implementation of the vhost-user-blk device backend inside QEMU so guests can efficiently access shared disk images.

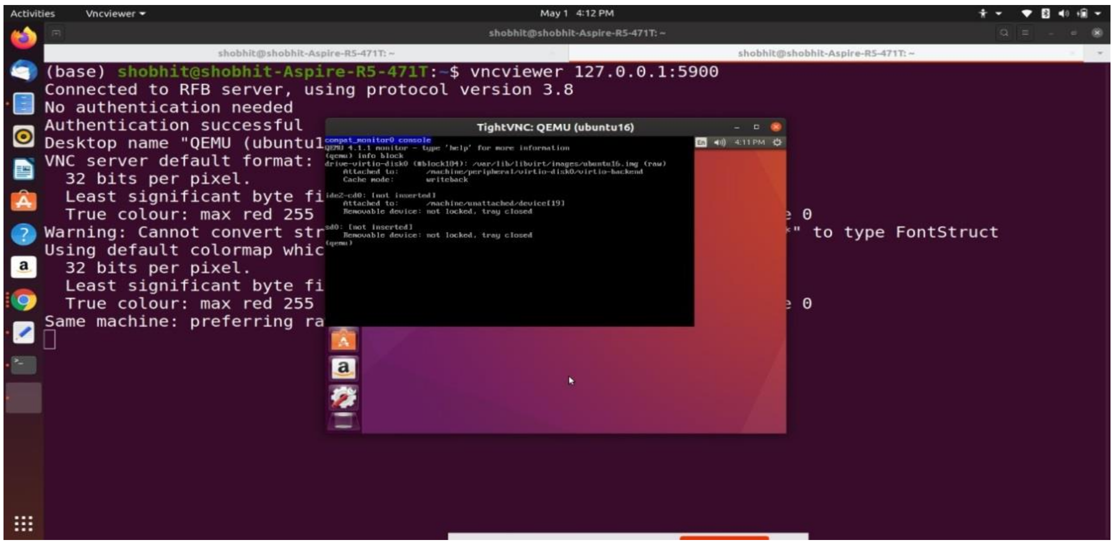
8.1 Snapshots



Activities
Vncviewer
May 1 4:11 PM
shobhit@shobhit-Aspire-R5-471T: ~
shobhit@shobhit-Aspire-R5-471T: ~
shobhit@shobhit-Aspire-R5-471T: ~
TightVNC: QEMU (ubuntu16)
Ubuntu Desktop
4:11 PM

```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5900
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop name "QEMU (ubuntu1
VNC server default format:
  32 bits per pixel.
  Least significant byte fi
  True colour: max red 255                            e 0
Warning: Cannot convert str                          *" to type FontStruct
Using default colormap whic
  32 bits per pixel.
  Least significant byte fi
  True colour: max red 255                            e 0
Same machine: preferring ra
```

Fig 8.1: The first Virtual Machine



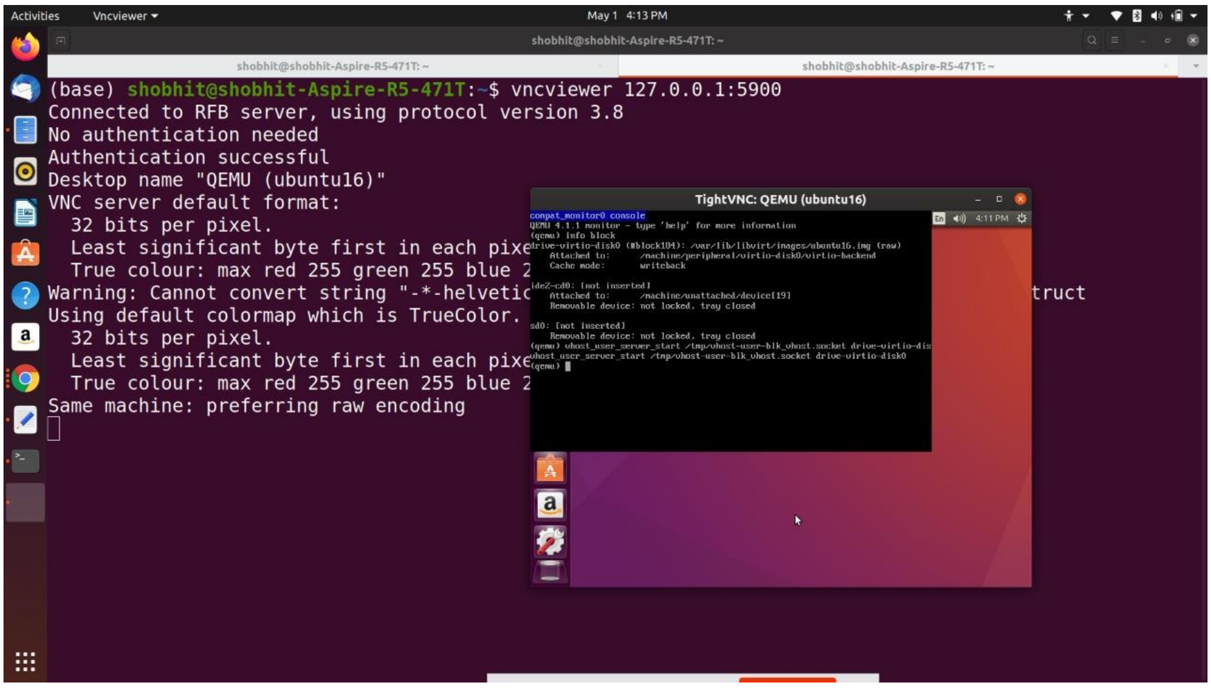
```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5900
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop name "QEMU (ubuntu1
VNC server default format:
  32 bits per pixel.
  Least significant byte fi                                    e 0
  True colour: max red 255
Warning: Cannot convert str                                   *" to type FontStruct
Using default colormap whic
  32 bits per pixel.
  Least significant byte fi
  True colour: max red 255                                     e 0
Same machine: preferring ra
```

```
compat_monitor0 console
QEMU 4.1.1 monitor - type 'help' for more information
(qemu) info block
drive-virtio-disk0 (#block104): /var/lib/libvirt/images/ubuntu16.img (raw)
    Attached to:      /machine/peripheral/virtio-disk0/virtio-backend
    Cache mode:       writeback

ide2-cd0: [not inserted]
    Attached to:      /machine/unattached/device[19]
    Removable device: not locked, tray closed

sd0: [not inserted]
    Removable device: not locked, tray closed
(qemu)
```

Fig 8.2: QEMU Console to see the shared disk



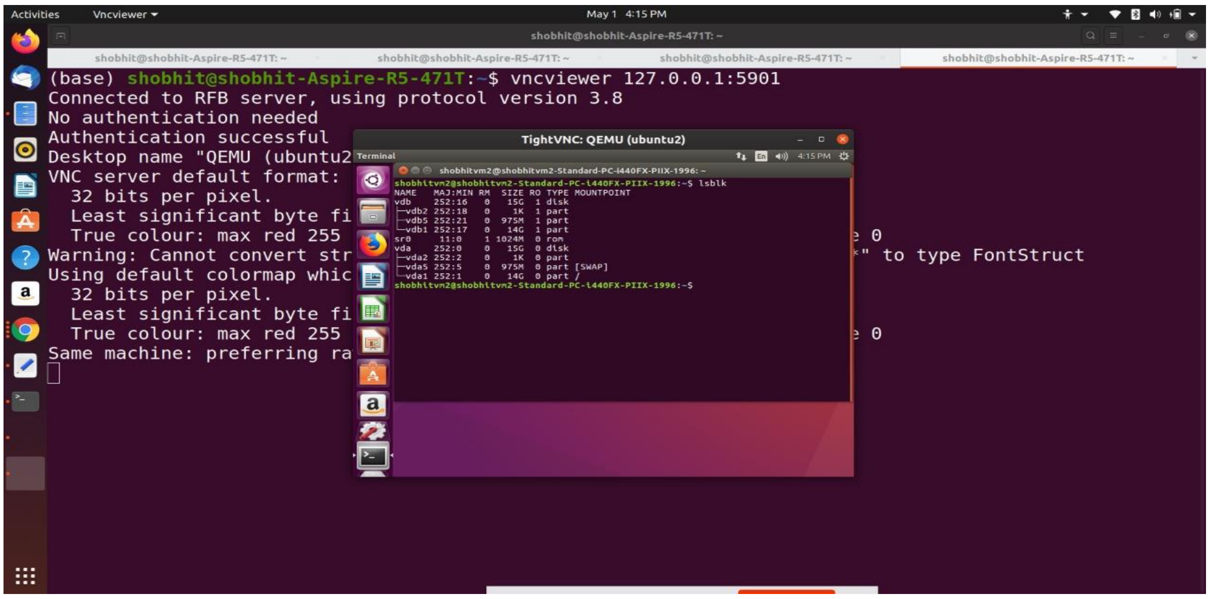
```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5900
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop name "QEMU (ubuntu16)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixe
  True colour: max red 255 green 255 blue 2
Warning: Cannot convert string "-*-helvetic                    truct
Using default colormap which is TrueColor.
  32 bits per pixel.
  Least significant byte first in each pixe
  True colour: max red 255 green 255 blue 2
Same machine: preferring raw encoding
```

```
compat_monitor0 console
QEMU 4.1.1 monitor - type 'help' for more information
(qemu) info block
drive-virtio-disk0 (#block104): /var/lib/libvirt/images/ubuntu16.img (raw)
    Attached to:      /machine/peripheral/virtio-disk0/virtio-backend
    Cache mode:       writeback

ide2-cd0: [not inserted]
    Attached to:      /machine/unattached/device[19]
    Removable device: not locked, tray closed

sd0: [not inserted]
    Removable device: not locked, tray closed
(qemu) vhost_user_server_start /tmp/vhost-user-blk_vhost.socket drive-virtio-dis
vhost_user_server_start /tmp/vhost-user-blk_vhost.socket drive-virtio-disk0
(qemu) 
```

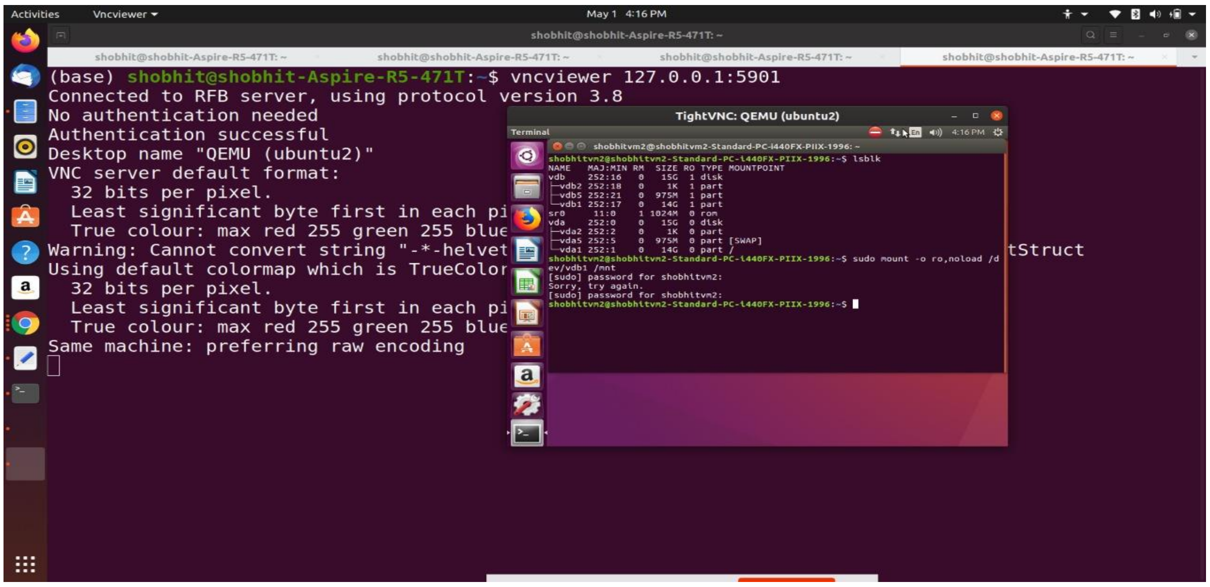
Fig 8.3: Starting the server


Activities
Vncviewer
May 1 4:15 PM
shobhit@shobhit-Aspire-R5-471T: ~
TightVNC: QEMU (ubuntu2)
Terminal
4:15 PM
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996: ~

```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5901
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop name "QEMU (ubuntu2
VNC server default format:
  32 bits per pixel.
  Least significant byte fi
  True colour: max red 255                                  e 0
Warning: Cannot convert str                                 *" to type FontStruct
Using default colormap whic
  32 bits per pixel.
  Least significant byte fi
  True colour: max red 255                                  e 0
Same machine: preferring ra
```

```
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vdb    252:16   0   15G  1 disk
├─vdb2 252:18   0    1K  1 part
├─vdb5 252:21   0  975M  1 part
└─vdb1 252:17   0   14G  1 part
sr0     11:0    1 1024M  0 rom
vda    252:0    0   15G  0 disk
├─vda2 252:2    0    1K  0 part
├─vda5 252:5    0  975M  0 part [SWAP]
└─vda1 252:1    0   14G  0 part /
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996:~$
```

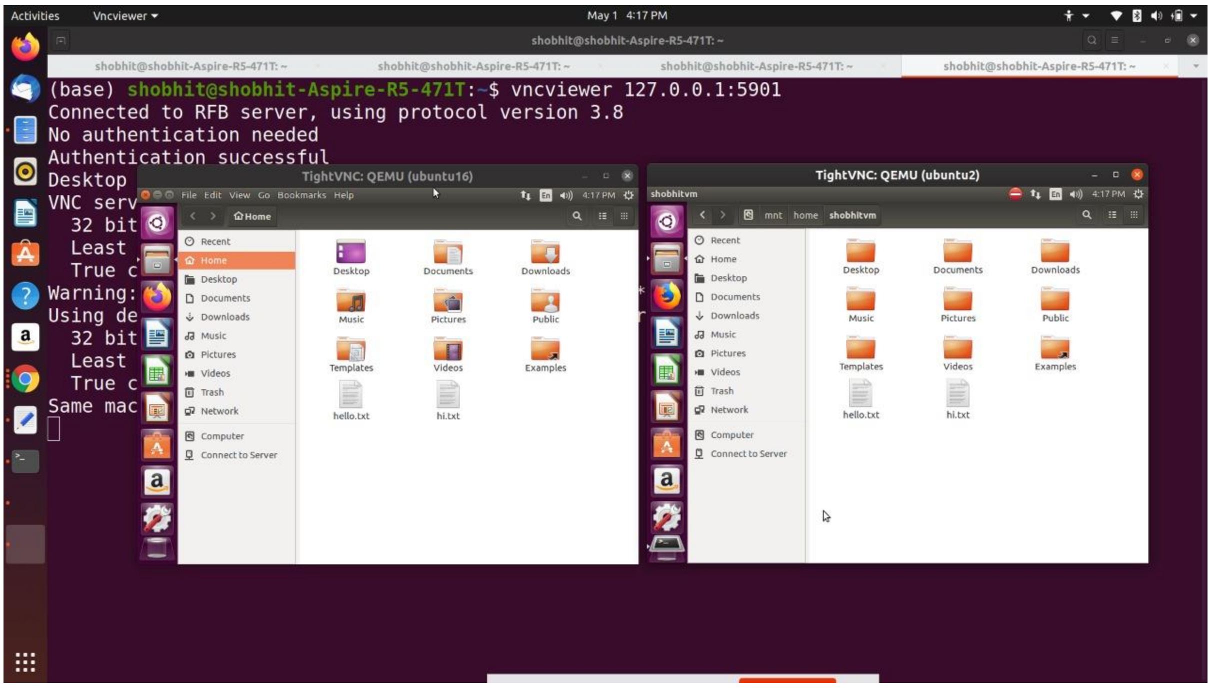
Fig 8.4: List of block disks


Activities
Vncviewer
May 1 4:16 PM
shobhit@shobhit-Aspire-R5-471T: ~
TightVNC: QEMU (ubuntu2)
Terminal
4:16 PM
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996: ~

```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5901
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop name "QEMU (ubuntu2)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pi
  True colour: max red 255 green 255 blue
Warning: Cannot convert string "-*-helvet                                    tStruct
Using default colormap which is TrueColor
  32 bits per pixel.
  Least significant byte first in each pi
  True colour: max red 255 green 255 blue
Same machine: preferring raw encoding
```

```
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996:~$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vdb    252:16   0   15G  1 disk
├─vdb2 252:18   0    1K  1 part
├─vdb5 252:21   0  975M  1 part
└─vdb1 252:17   0   14G  1 part
sr0     11:0    1 1024M  0 rom
vda    252:0    0   15G  0 disk
├─vda2 252:2    0    1K  0 part
├─vda5 252:5    0  975M  0 part [SWAP]
└─vda1 252:1    0   14G  0 part /
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996:~$ sudo mount -o ro,noload /d
ev/vdb1 /mnt
[sudo] password for shobhitvm2:
Sorry, try again.
[sudo] password for shobhitvm2:
shobhitvm2@shobhitvm2-Standard-PC-i440FX-PIIX-1996:~$ 
```

Fig 8.5: Mounting to the server disk



```
(base) shobhit@shobhit-Aspire-R5-471T:~$ vncviewer 127.0.0.1:5901
Connected to RFB server, using protocol version 3.8
No authentication needed
Authentication successful
Desktop
VNC serv
  32 bit
  Least
  True c
Warning:
Using de
  32 bit
  Least
  True c
Same mac
```

Fig 8.6: Shared disks images being accessed by the Virtual Machines

Chapter-9

Conclusion and Future Scope

In the project we implemented a vhost-user-block backend service in QEMU. The vhost-user-block backend service was used to enable a QEMU to act as a server and share a virtual disk with another client QEMU VM. This solution should provide better performance compared to existing NBD based solution in QEMU. As part of future work, we plan to evaluate our implementation’s performance and compare it to NBD based solution. We also plan to extend the implement to enable multiple QEMU clients to connect to the QEMU server to enabling sharing virtual disks between more than 2 VMs. If the QEMU client caches the shared disk data, it doesn’t immediately see the updates made by other clients or QEMU server. We plan to implement more consistent updates to the shared disk so that each client can see the updated data immediately.

References

1. Official Website of QEMU https://www.qemu.org/

2. Official Website of Linux-KVM http://linuxkvm.org/

3. virtio official page https://wiki.libvirt.org/page/Virtio

4. Virtualised I/O with vhost-user https://spdk.io/doc/vhost_processing.html