Continuous and Concurrent Network Connection for Hardware Virtualization

by

Devarani Kumarasan

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Technology (Hons)
(Information & Communication Technology)

DECEMBER 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Continuous and Concurrent Network Connection

for

Hardware Virtualization

by

Devarani Kumarasan

A project dissertation submitted to the

Information Communication Technology Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)

(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,

[illegible]

(Ms Nazleeni Samiha Binti Haron)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

DEVARANI KUMARASAN

ABSTRACT

This project addresses the network connectivity in virtualization for cloud computing. Each Virtual Machine will be able to access the network concurrently and obtains continuous internet connectivity without any disruption. This project proposes a new method of resource sharing which is the Network Interface Card (NIC) among the Virtual Machines with each of them having the full access to it with near-native bandwidth. With this, could computing can perform resource allocation more effectively. This will be essential to migrate the each Operating System (Virtual Machine) that resides on one physical machine to another without disrupting its internet or network connection.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

CHAPTER 1

INTRODUCTION

1.1 PROJECT BACKGROUND

Cloud computing has attracted major attention of the computing world recently. Many groups of people from developers to businessman has shown a major interest in the development of cloud computing. However, the fact that cloud computing is still at the infant stage cannot not be denied. It still has a lot more room for improvement. One important part of cloud computing is Virtualization where it is required in order to enable cloud computing. There are four virtualization techniques in common use today, namely guest operating systems, shared kernel, hypervisor and kernel level. Firstly is the Guest Operating System Virtualization. In this approach, the physical host system runs a standard unmodified operating system. The Shared Kernel Virtualization, also known as system level or operating system virtualization, takes the advantage of the architectural design of the Linux and Unix based operating systems. Under kernel level virtualization the host operating system runs on a specially modified kernel which contains extensions designed to manage and control multiple virtual machines each containing a guest operating system. Finally is the Hypervisor Virtualization. Under hypervisor virtualization, a program known as the hypervisor (also known as the Virtual Machine Monitor or VMM) runs directly on the hardware of the host system. Discussing about virtualization, we can never overlook the importance of network, thus, this also makes network to become a major part of cloud computing. This proposal will propose the project of implementation of continuous network connectivity in virtualization for cloud computing.

1.2 PROBLEM STATEMENT

As computers became more ubiquitous however, it became apparent that simply time-sharing a single computer was not always ideal. For organizations that could easily afford it, they simply purchased multiple computer systems to mitigate these pitfalls. Most organizations at the same time were not so fortunate to be able to purchase multiple computer systems, it was also recognized that purchasing multiple computers was often wasteful, as having more computers made it even harder to optimize them fully. Putting into consideration that having multiple computers is still required, problem of high cost and low optimization can be tackled through cloud computing. Virtualization is a vital part in the infrastructure layer of cloud computing model. With virtualization, one physical system can run multiple Operating Systems. However, problem occurs when each of this Operating Systems want to access to the network with the close to native bandwidth, continuously and concurrently. This is the main issue that will be looked into in this project.

1.3 PROJECT OBJECTIVE

The objectives of this project are:

- To cater concurrent network access with network virtualization for cloud computing.
- To implement continuous network connection in virtualization for cloud computing.

1.4 SCOPE OF STUDY

The project covers only the virtualization part of the cloud computing. Cloud computing implements various types of virtualization depending on its requirement. This project will only concentrate upon hardware-based full virtualization. The network is a vital in virtualization as to manage the resources to the optimum level through migration. Thus, research will be done on the best method to cater concurrent network access and continuous network connection.

Critical review and analysis will be conducted to develop an optimized system that can enhance the network connectivity. Thorough various types of configuration will be done in order to investigate alternative setup solutions. The outcomes of the various configurations will be evaluated, tested, validated and compared with the current results of the cloud computing system.

CHAPTER 2

LITERATURE REVIEW

2.1 CLOUD COMPUTING

Cloud computing is the deployment of a new technology where the resources such as the CPU and storage are provided as general utilities that can be leased and released by users through the internet in an on-demand basis [1]. Cloud computing describes computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Cloud computing comes into focus only when you think about what IT always needs: a way to increase capacity or add capabilities without investing in new infrastructure [2]. Cloud computing encompasses any subscription-based or pay-per-use service that, in real time over the Internet, extends IT's existing capabilities. Cloud computing is often related to virtualization which becomes a vital part of virtualization where it is almost impossible to rectify the effectiveness of cloud computing without its presence.

2.2 VIRTUALIZATION

IBM developed the virtual machine concept as a way of time-sharing very expensive mainframe computers. The virtual machine concept allows the same computer to be shared as if it were several. IBM defined the virtual machine as a fully protected and isolated copy of the underlying physical machine's hardware [3]. IBM designed their virtual machine systems with the goal that applications, even operating systems, run in the virtual machine would behave exactly as they would on the original hardware [4]. Intel ® Virtualization Technology (Intel ® VT) [5] provides greater flexibility and maximum system utilization by consolidating multiple environments into a single server, workstation, or PC. With fewer systems required for the same tasks, Intel ®VT delivers:

- Simplified resources management increasing IT efficiency.
- Greater systems reliability and availability reducing corporate risk and real-time losses from downtime.
- Lower hardware acquisition costs with increases utilization of the machines that already available.

With support from the processor, chipset, BIOS, and enabling software, Intel VT improves traditional software-based virtualization. Taking advantage of offloading workloads to system hardware, these integrated features enable virtualization software to provide more streamlined software stacks and "near native" performance characteristics. Virtualization solutions enhanced by Intel VT allow a platform to run multiple operating systems (OSs) and applications as independent virtual machines, allowing one computer system to function as multiple "virtual" systems.

According to the Figure 1, it shows that the typical server system (left) where each Operating System runs on top of one physical systems. With the implementation of virtualization, now we can run more than one, in this case five, operating systems (virtual machines) on top of one physical server (right). The system runs a program to manage the virtual machines which is known as Virtual Machine Manager or hypervisor [6].

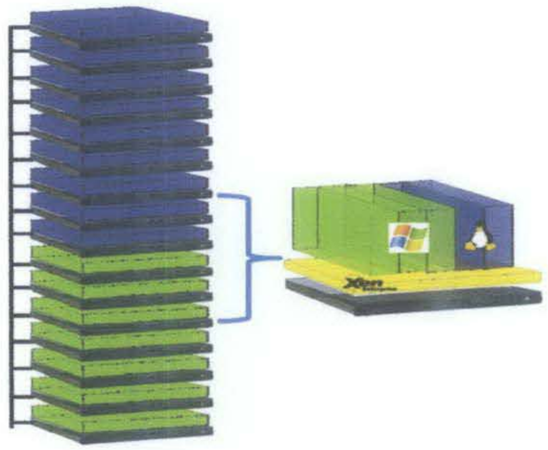


Figure 1: The layout of the server system before and after Virtualization

The hypervisor that is purposed for this project is XEN. Xen is a virtualization system supporting Paravirtualization (PV) and Hardware-assistant Full Virtualization (HVM). Paravirtualization means that the OS knows that it is actually a virtual machine sitting on a hypervisor and the OS, drivers and other resources are modified for the paravirtualized environment. Whereas, fully virtualized (HVM) provides a virtual replica of the system's hardware so that the OS and software may run on the virtual hardware exactly as they would on the original hardware. This makes PV performs better because the drivers have been modified for shared resources but HVMs more flexible.

Choosing the best approach to implement virtualization for this project is important. There are four virtualization techniques in common use today, namely guest operating systems, shared kernel, hypervisor and kernel level [7].

Firstly is the Guest Operating System Virtualization. In this approach, the physical host system runs a standard unmodified operating system. On this operating system, a vitualization application or software which executes in much the same way as any other applications such as a word processor or spreadsheet or media player would run on the system. It is within this virtualization application that one or more virtual machines are created to run the guest operating system or the VM. This application starts, stops, and manages each of the VM and essentially controls the access to physical hardware resources on behalf of each individual VM. The VM thinks that it is running directly on the system hardware, rather than in a VM within an application. Examples of guest OS virtualization technology include VMware Server and VirtualBox. Figure 2 illustrates this technology:

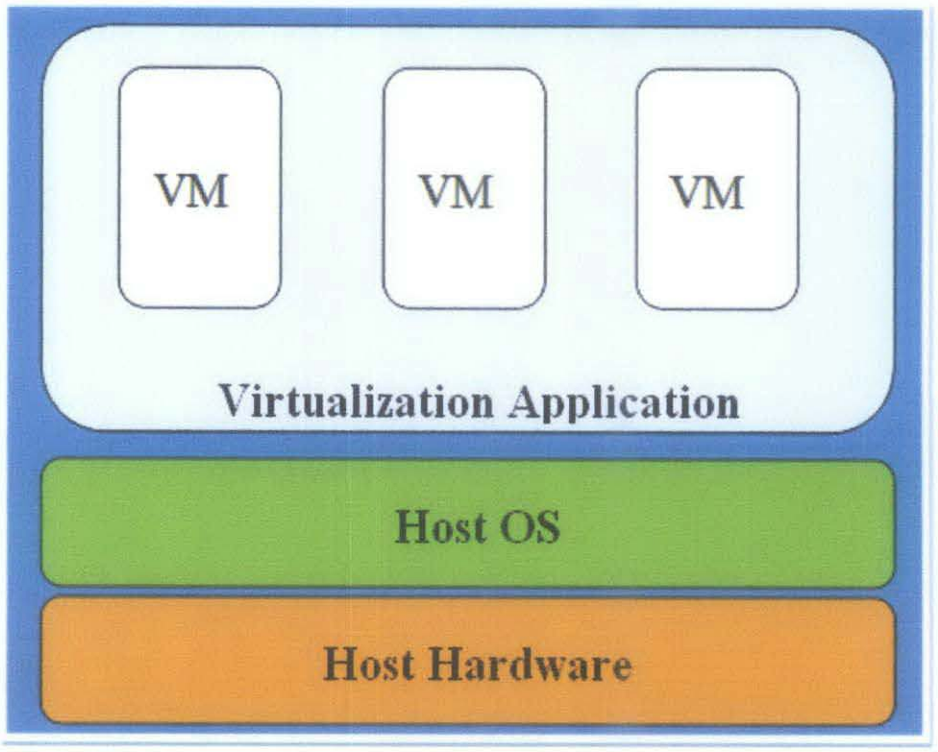


Figure 2: Guest Operating System Virtualization

The Shared Kernel Virtualization, also known as system level or operating system virtualization, takes the advantage of the architectural design of the Linux and Unix based operating systems. Under shared kernel virtualization the virtual guest systems each have their own root file system but share the kernel of the host operating system. This type of virtualization is made possible by the ability of the kernel to dynamically change the current root file system to a different root file system without having to reboot the entire system. The single disadvantage of this form of virtualization is the fact that the guest operating systems must be compatible with the version of the kernel which is being shared. Examples of the shared kernel virtualization are Linux VServer, Solaris Zones and Containers, FreeVPS and OpenV. The figure below explains the structure of this technology:

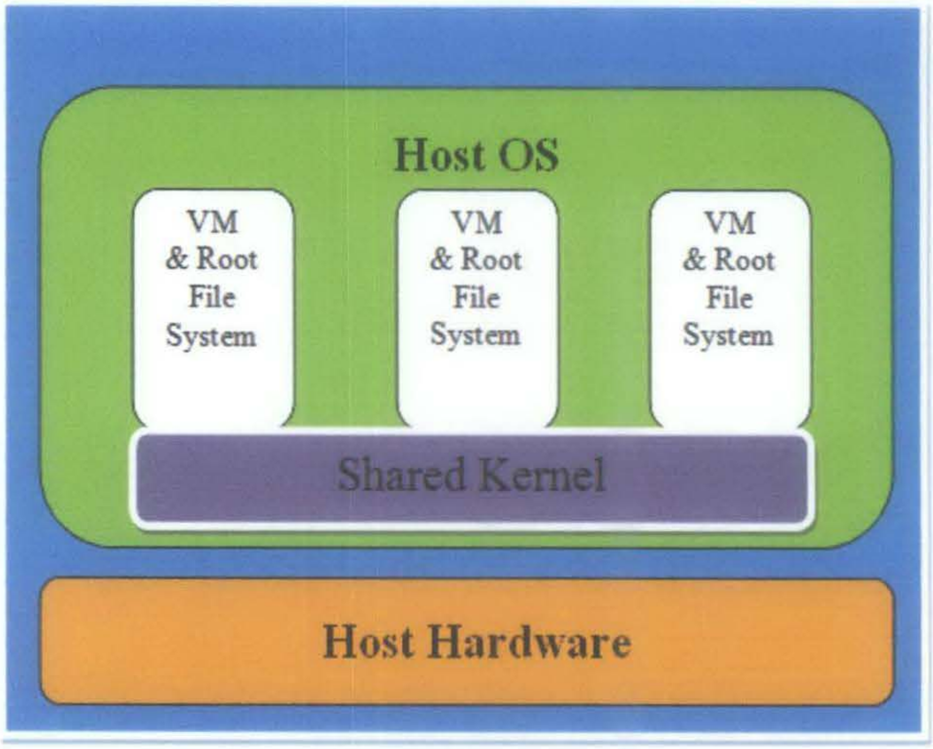


Figure 3: Shared Kernel Virtualization

Under kernel level virtualization the host operating system runs on a specially modified kernel which contains extensions designed to manage and control multiple virtual machines each containing a guest operating system. Unlike shared kernel virtualization each guest runs its own kernel, although similar restrictions apply in that the VMs must have been compiled for the same hardware as the kernel in which they are running. Examples of kernel level virtualization technologies include User Mode Linux (UML) and Kernel-based Virtual Machine (KVM) [8]. The following figure provides an overview of the kernel level virtualization:

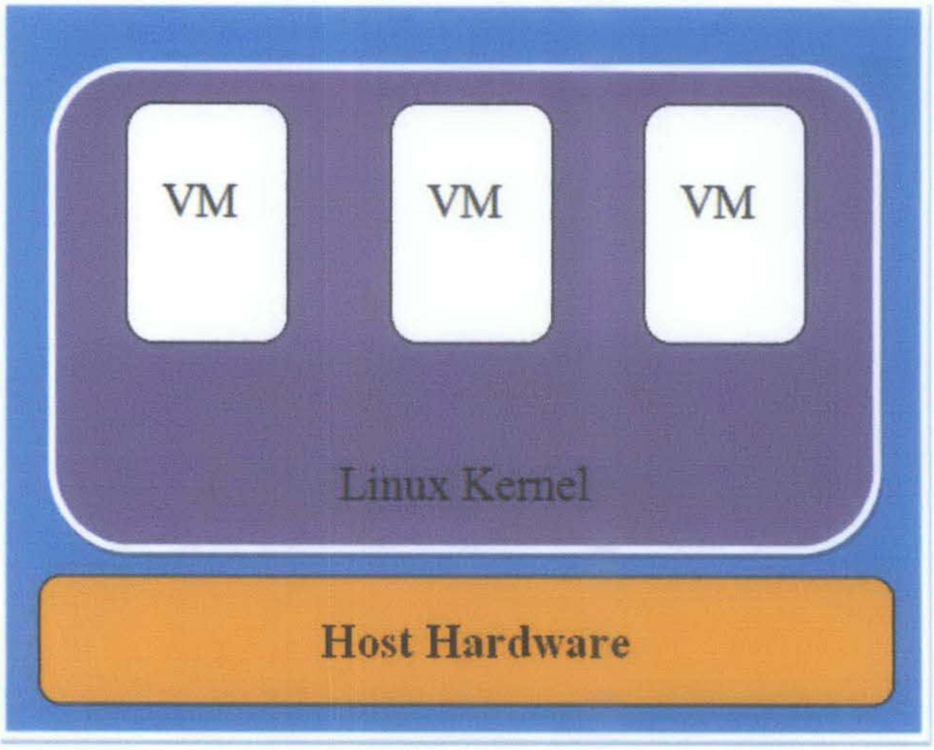


Figure 4: Kernel Level Virtualization

Finally is the Hypervisor Virtualization. Under hypervisor virtualization, a program known as the hypervisor (also known as the Virtual Machine Monitor or VMM) runs directly on the hardware of the host system. The task of this hypervisor is to handle resource and memory allocation for the virtual machines in addition to providing interfaces for higher level administration and monitoring tools. As outlined in Figure 5, in addition to the VMs, an administrative operating system or management console also runs on top of the hypervisor allowing the virtual machines to be managed by a system administrator. Hypervisor based virtualization solutions include Xen, VMware ESX Server and Microsoft's Hyper-V technology. Different types of solution for resources and memory allocation for hypervisor virtualization are paravirtualization, full virtualization and hardware virtualization.

For paravirtualizaton, the kernel of the VM is modified specifically to run on the hypervisor. This typically involves replacing any privileged operations of the CPU with calls to the hypervisor. The hypervisor in turn performs the task on behalf of the guest kernel. This limits the support to open source operating systems such as Linux which may be freely alters and proprietary operating systems where the owners have agreed to make the necessary code modifications to target a specific hypervisor. The drawback is the VM's kernel unable to communicate directly with the hypervisor resulting in greater performance levels than other virtualization approaches.

Full Virtualization provides support for unmodified guest operating systems which means the VM's kernel is not altered to run on a hypervisor. Thus, it still executes privileged operations. The hypervisor provides PCU emulation to handle and modify privileged and protected CPU operations made by unmodified guest operating system kernels. However, this emulation process requires both time and system resources to operate, resulting in inferior performance levels when compared to those provided by paravirtualization.

The Hardware Virtualization leverages virtualization features built into the latest generations of CPUs from both Intel and AMD. These technologies, known as Intel VT and AMD-V respectively, provide extensions necessary to run unmodified guest virtual machines without the overheads inherent in full virtualization CPU emulation. In simple terms, these new processors provide an additional privilege mode in which the hypervisor can operate essentially leaving the guest operating systems unmodified.

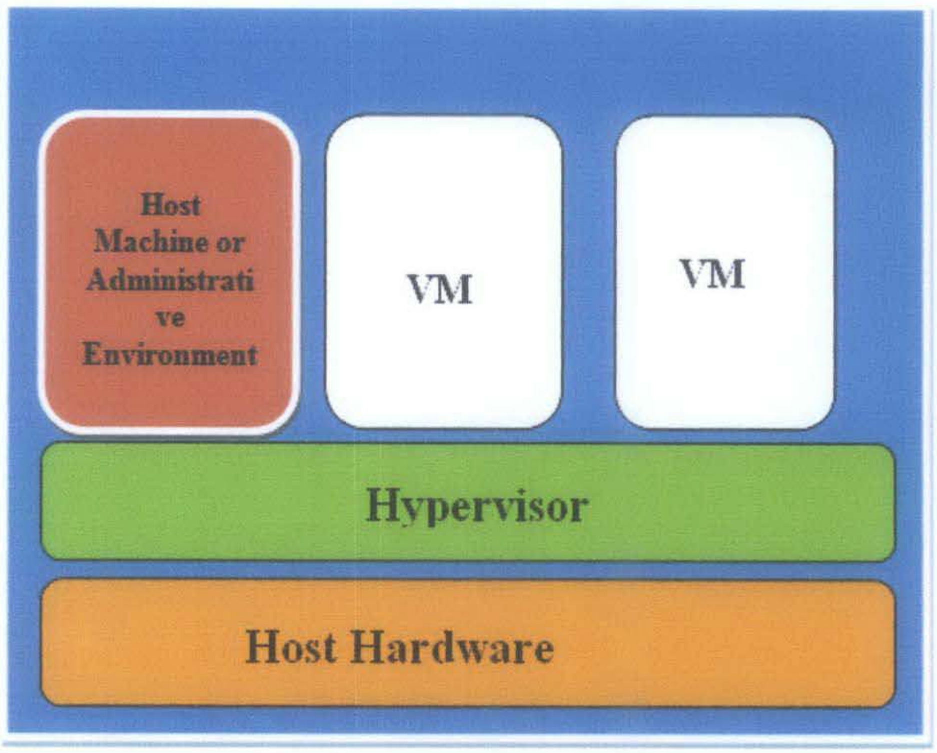


Figure 5: Hypervisor Virtualization

HVM running on Xen hypervisor will be implemented for this project. Figure 6 shows how the actual virtualized environment looks like. The hypervisor lies on top of the hardware and the OSes runs on top of the hypervisor. Each OS has to go through the hypervisor to access the hardware or resources.

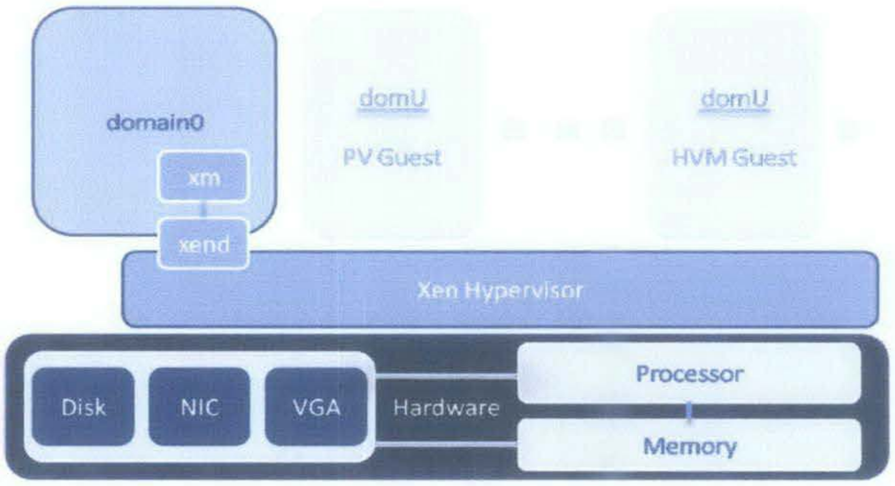


Figure 6: Complete Virtualization environment

2.3 LIVE MIGRATION

Now, with the virtual machines installed in the system, the migration of virtual machines can be done. Migration is the ability to easily move a virtual machine across the network from one physical host to another can be useful for a number of different administrative tasks such as load balancing or dealing with scheduled maintenance. Xen provides integrated relocation, or migration support. It helps manage the process of preparing, transporting, and resuming guests from one nearby host to another.

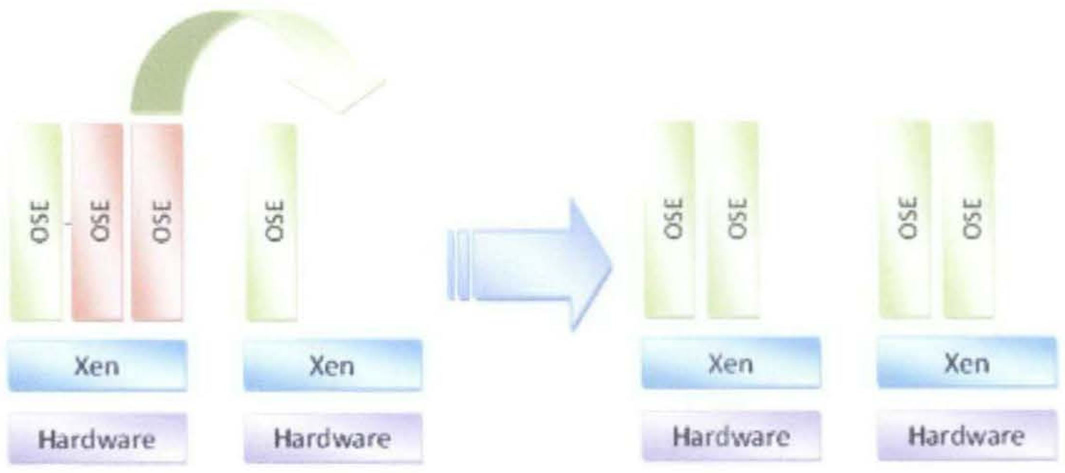


Figure 7: Live Migration Concept

There are many types of migration: cold static relocation, warm static migration, and live migration [9]. Cold static relocation is accomplished manually without the help of Xen's integrated migration facility. There are two ways of accomplishing this goal. The first occurs when both hosts share underlying storage (network attached storage). The second method involves manually copying the configuration file and file systems from the host to the target hardware. Warm static migration, or regular migration, of a guest domain is the combined process of pausing the execution of that guest's processes on its original host, transferring its memory and processes from its origin host to a destination host, and resuming its execution on the destination host. As for this project, live migration will be applied. The mere ability to migrate a guest domain from one physical host to another is beneficial, but performing migration by temporarily suspending and then restoring a guest's state is not suitable in all applications. Live migration, enables a domain to be migrated while it is in operation and without the interruption of its services or connections, as illustrated in Figure 7. Live migration of a guest is the act of seamlessly moving its execution to the new physical host, including redirecting established and future network connections away from its original and to its new location.

2.4 VT-D FOR PCI PASSTHROUGH

Once the live-migration of the virtual machines can be conducted impeccably, the isolation of NIC can be implemented. This basically means making use the Intel VT for Directed I/O (VT-d). VT-d is a technique to give a virtual machine exclusive access to a Peripheral Component Interconnect (PCI) function using the IOMMU provided by Xen for VT-d [10]. Exclusive access here means the virtual machine can easily access to the hardware without passing through the hypervisor (Xen). Thus, the virtual machine has the complete access to the hardware that is being passthrough. That particular hardware is hidden from the Xen and other OSes. Two Network Interface Cards (NIC) will be used where one of the NICs will be connected the VM to the system with bridge. The other NIC will be hidden from the hypervisor (Xen) and host but assigned to the virtual machine. The second NIC will bypass the hypervisor and thus, the virtual machine has the full access to the NIC. The importance of this NIC passthrough can be seen in the later stage.

2.5 NETWORK BONDING

When the passthrough is complete, network bonding will be implemented. The Linux bonding driver provides a method for aggregating multiple network interface into a single logical "bonded" interface, multiplying the bandwidth [11] [12]. Before implementing bonding, it is important to understand Link Aggregation. Link aggregation is used to increase the speed of a link by bundling network cables/ports in parallel. There are a lot of other term of link aggregation including Ethernet/network bonding, NIC teaming, link bundling, and trunking. By achieving network bonding, we can address tow main problems: bandwidth limitations, and lack of redundancy.

There are many bonding mode and Mode 1 which is active-backup or Active-backup policy is the one that is related to this project . Refer to Appendix 1 for the list of other bonding modes. According to this mode, only one slave (primary slave) out of two bonded interfaces (network interface) will be active. The other slave (secondary slave) becomes active if, and only if, the active slave fails [13].

2.6 LIVE MIGRATION WITH NIC PASSTHROUGH (VT-D) AND NETWORK BONDING

Now the live migration can be repeated. To implement video streaming with live migration, the VM would stream a video that is stored in the storage system through the network. During the video streaming, it can be seen that the bandwidth of the video will be close to 1.0 Gb. While the video was being streamed, the VM will detach its first NIC as live migration cannot be done with the NIC still attached to the network. Once the first NIC is detached, the second NIC will take over to make sure continuous network connection. Though the bandwidth drops, the video streaming still continues without any disruption. Live migration begins and the time it takes for the VM to migrate depends on its size [14]. During the live migration, the user of the VM would not see any difference and would not realize that live migration is being conducted in the background. Once the live migration is completed, the network interface has to be attached to the system again manually. The bandwidth will again improve back to almost the native bandwidth.

2.7 SINGLE ROOT I/O VIRTUALIZATION (SR-IOV)

The final stage would be the SR-IOV. SR-IOV which stands for Single Root I/O Virtualization is a specification that allows a PCIe device to appear to be multiple separate physical PCIe device. SR-IOV works by introducing the idea of *physical functions (PFs)* and *virtual functions (VFs)* [15]. Physical functions (PFs) are full-featured PCIe functions; virtual functions (VFs) are “lightweight” functions that lack configuration resources. SR-IOV requires support in the BIOS as well as in the operating system instance or hypervisor that is running on the hardware. SR-IOV will require the use of 10Gb Ethernet Card. These 10Gb bandwidth capability can be divided to create ten virtual NIC. Each virtual network will be assigned to different virtual machines on the same system. This means, the system will no more require multiple NICs and in fact will still obtain the almost 1Gb network bandwidth.

2.8 RELATED STUDIES

According to Lou in his paper Network I/O Virtualization for Cloud Computing [16], has done similar project. He performed the steps including virtualization, live migration, PCI passthrough and network bonding. However, the difference between this project and Lou's is that the bonding and the SR-IOV will be conducted in a different manner. Lou still uses the 1Gb Ethernet card and uses the SR-IOV to assign each Virtual Machines with the virtualized network card. Although each Virtual Machine appear to own the network access, but it does not have the full control over the network card as the Virtual Machines are still sharing the network card among themselves following a scheduling process. This disables the concurrent network access as well as continuous network connection. This project tackles these issues by dividing the 10Gb Ethernet card to 1Gb and assigning each Virtual Machines to 1Gb network capability with using the SR-IOV method as well.

Another research done by Dong [17], optimization of Network I/O Virtualization was done with efficient Interrupt Coalescing and Virtual Receive Side Scaling. In this paper, the performance challenges in network I/O virtualization is analyzed and observed that the conventional network I/O virtualization incurs excessive virtual interrupts to guest VMs, and the backend driver in the driver domain is not parallelized and cannot leverage underlying multi-core processors. Motivated by the above observations, optimizations were proposed through: efficient interrupt coalescing for network I/O virtualization and virtual receive side scaling to effectively leverage multi-core processors. Those optimizations in Xen and extensive performance evaluation was done. The experimental results revealed that the proposed optimizations significantly improve network I/O virtualization performance and effectively tackle the performance challenges. However, this method only optimized the allocation of network I/O but does not enable continuous and concurrent network.

In another paper on A Modeling of Network I/O Efficiency in Xen Virtualized Clouds by Pu *et al* [18],based on the server side and client side, two groups of equations are proposed to mathematical modeling the exchanged memory pages in Xen I/O channel. According to the designed experiments, the proposed functions calculated the number of exchanged memory pages accurately, with an average error rate around 6%, even in multiple VMs scenario. Meanwhile, the effects of varying MTU (Maximum Transmission Unit) size respectively based on our experimental results and proposed equations were illustrated and analyzed. Afterwards, an effective improvement for optimizing network I/O performance is also proposed in this paper. This method also only focuses on optimization instead of providing dedicated network connectivity for each VM.

Dong Y. et al [19], based on the first implementation of network device driver, several optimizations were applied to reduce virtualization overhead. Then, comprehensive experiments to evaluate SR-IOV performance were carried out and compared with paravirtualized network driver. The results show SR-IOV can achieve line rate (9.48Gbps) and scale network up to 60 VMs at the cost of only 1.76% additional CPU overhead per VM, without sacrificing throughput. It has better throughout, scalability, and lower CPU utilization than paravirtualization. This research which is on SR-IOV shows that it can be implemented to achieve the objective of this project which is to enable concurrent network connection.

CHAPTER 3

METHODOLOGY

3.1 PHASE 1: SETUP: HARDWARE REQUIREMENT

This project requires two server systems (System1 and System2). Both System1 and System2 have to be of similar server system. Another server system that is Network File System (NFS) which acts as shared server. The forth system is the Control System can be of any platform which will be used to access all the other three system from one point. PuTTy will be used to connect to System1, System2 and NFS system from the Control System.

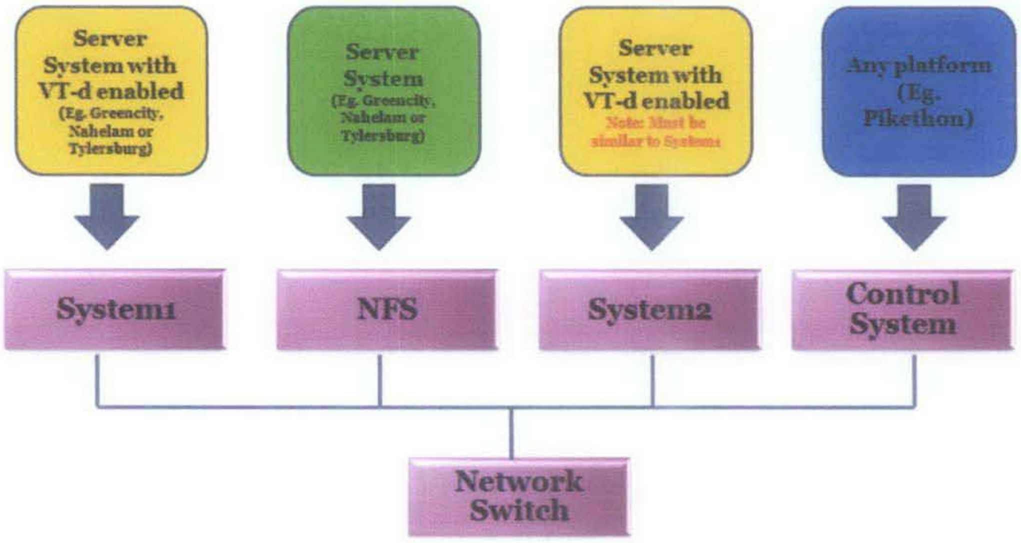

Server System with VT-d enabled (Eg. Greencity, Nahelam or Tylersburg)
Server System (Eg. Greencity, Nahelam or Tylersburg)
Server System with VT-d enabled
Note: Must be similar to System1
Any platform (Eg. Pikethon)
System1
NFS
System2
Control System
Network Switch

Figure 8: Virtualization Hardware Setup

The system is set as shown in the Figure 8 above. All the systems are connected to the network switch. System1 andSystem2 have two separate Network Interface Cards (NIC). Both NICs are connected to the network switch. The Figure 9 below shows how each system is connected to the network switch.

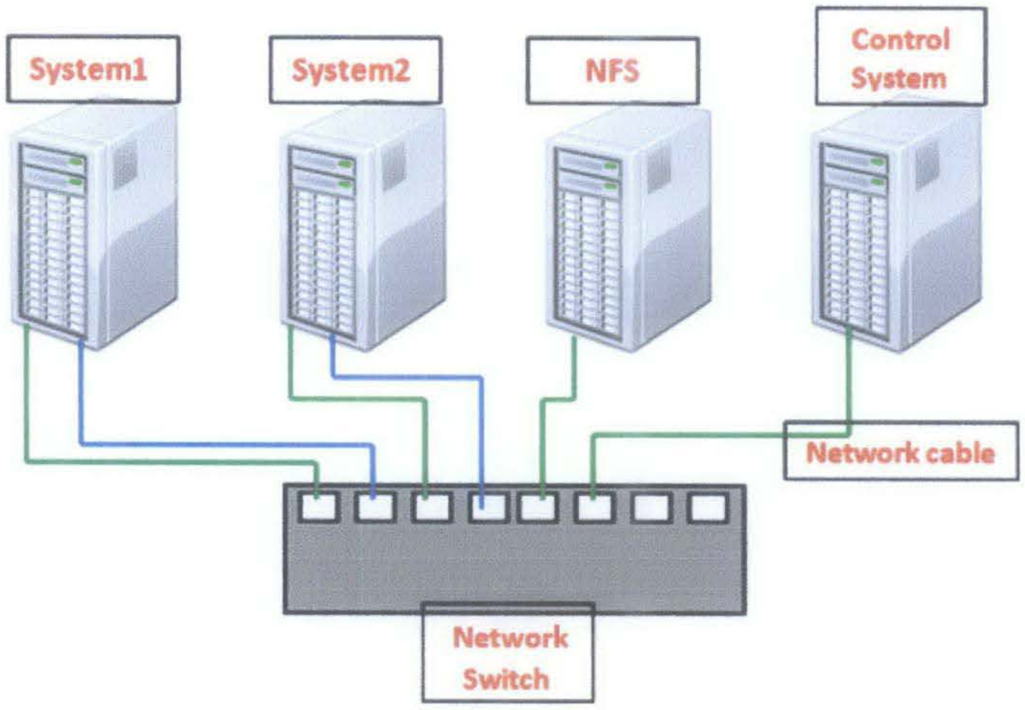


Figure 9: Virtualization Network Setup

3.2 PHASE 2: SETUP: SOFTWARE REQUIREMENT

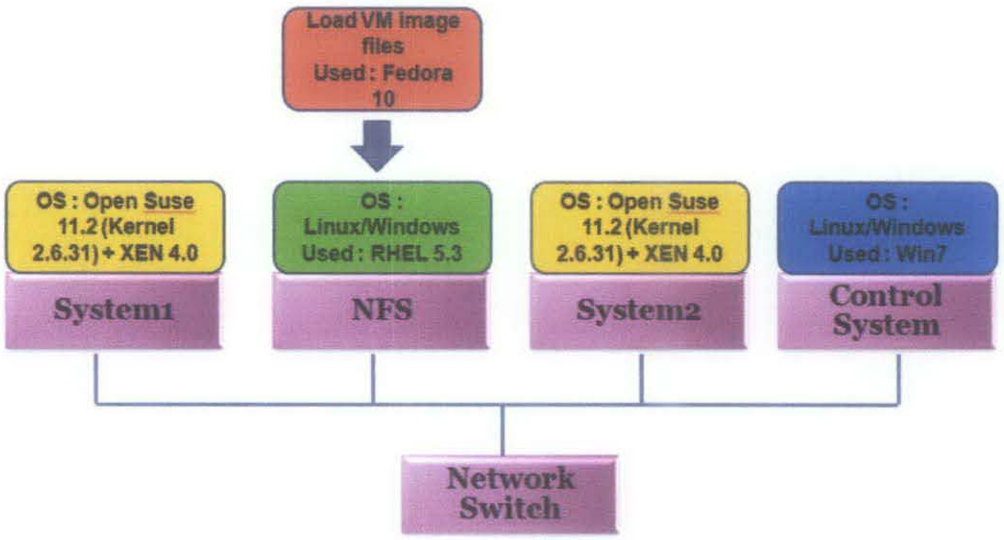

Load VM image files Used : Fedora 10
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
OS : Linux/Windows Used : RHEL 5.3
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
OS : Linux/Windows Used : Win7
System1
NFS
System2
Control System
Network Switch

Figure 10: Virtualization Software Setup

Figure 10 shows the software setup of this project. System1 and System2 are installed with OpenSuse 11.2 operating system with Xen 4.0 hypervisor. System2 must be installed and setup similar to System1. The NFS system can be installed with any operating system. For this project, we have installed RHEL 5.3. As for the Control System, it can be installed with any operating system. I have used Windows7. The image files of the Virtual Machine, for this project Fedore Core 10, is uploaded to the NFS.

3.3 PHASE 3: SETUP: CREATING THE VIRTUAL MACHINE

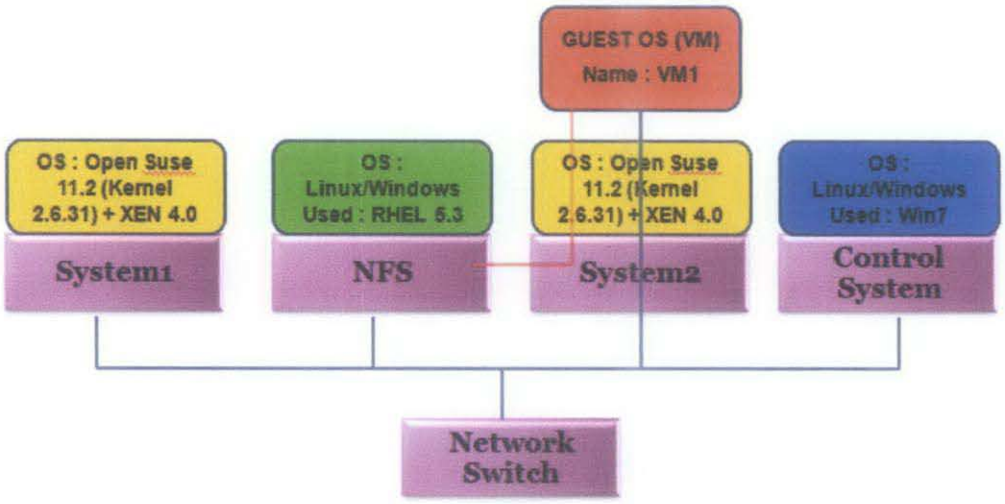

GUEST OS (VM)
Name : VM1
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
System1
OS : Linux/Windows Used : RHEL 5.3
NFS
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
System2
OS : Linux/Windows Used : Win7
Control System
Network Switch

Figure 11: Virtual Machine Setup

The Figure 11 above shows that the Virtual Machine (VM) is installed on the System2. Xen hypervisor is enabled. The VM's image file is loaded from the NFS system and installed on top of the Xen hypervisor. The VM is then named as VM1.

3.4 PHASE 4: LIVE MIGRATION

Live migration of the Virtual Machine from System1 to System2 will be performed. The Figure 12 shows how live migration is conducted for this project. When live migration is done, the VM will disconnect its network connection form System2 and is migrated to System1 and then establishes the network connection again through System1.

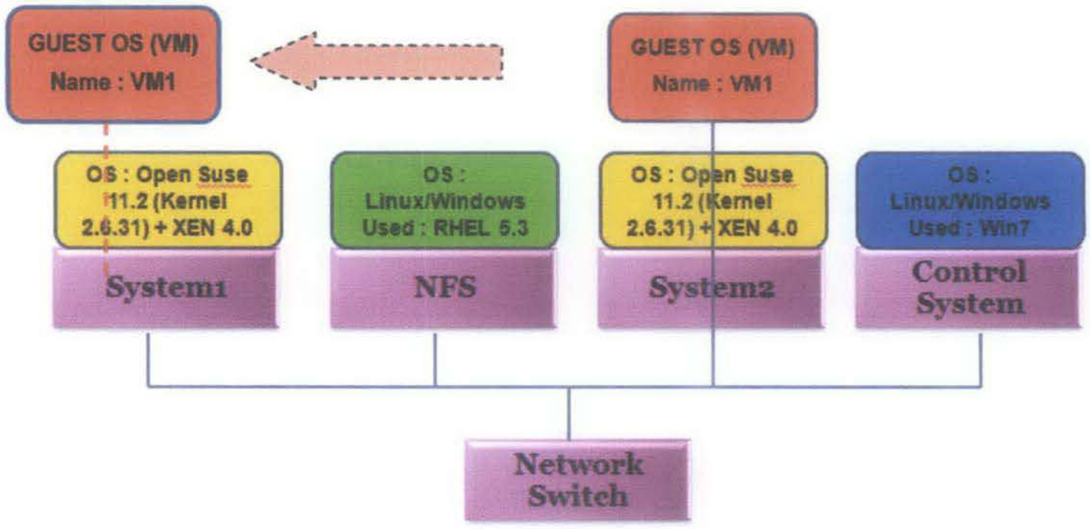

GUEST OS (VM)
Name : VM1
GUEST OS (VM)
Name : VM1
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
OS : Linux/Windows Used : RHEL 5.3
OS : Open Suse 11.2 (Kernel 2.6.31) + XEN 4.0
OS : Linux/Windows Used : Win7
System1
NFS
System2
Control System
Network Switch

Figure 12: Live Migration

3.5 PHASE 5: VT FOR DIRECTED I/O (VT-D) PCI PASSTHROUGH

For this project the PIC that will be passthrough is the Network Interface Card (NIC).

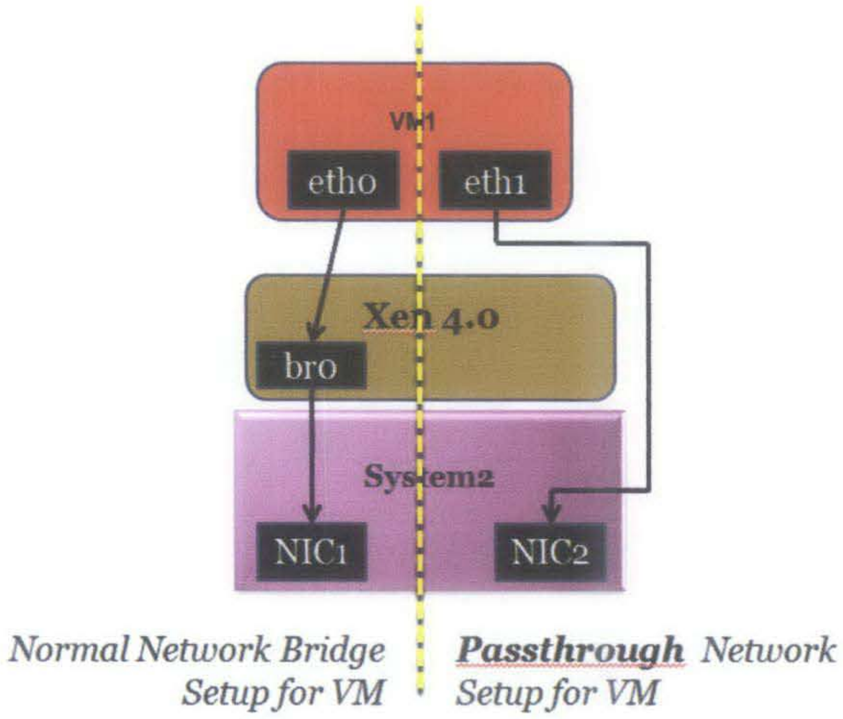

VM1
etho
eth1
Xen 4.0
bro
System2
NIC1
NIC2
Normal Network Bridge Setup for VM
Passthrough Network Setup for VM

Figure 13: NIC Passthrough Structure

The Figure 13 shows how VT-d was implemented. In order to implement NIC passthrough, two NIC card is required. One NIC card (eth0) will be connected the VM to the system with bridge (shown in the left side of the yellow line in the Figure X). The other NIC (eth1) will be created and then hidden from the hypervisor and host but assigned to the VM (shown in the right side of the yellow line in the Figure X). From the figure it can be seen that the second NIC bypassed the hypervisor. Thus, the VM has the full access to the NIC.

3.6 PHASE 6: NETWORK BONDING

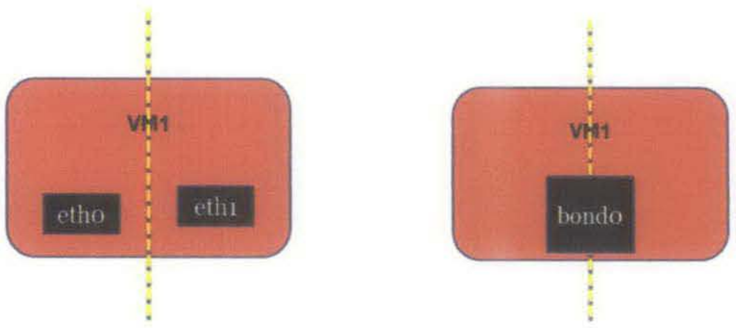


Figure 14: Network Bonding Structure

According to the Figure 14, eth1 that was created during NIC passsthrough, will be combined or aggregated to eth0 and renamed it bond0. bond0 will appear to be one logical network port and the bandwidth will be close to the native bandwidth (almost 1Gb).

3.7 PHASE 7: LIVE MIGRATION WITH NIC PASSTHROUGH (VT-D) AND NETWORK BONDING

Figure 15 shows the structure of the complete system. Now, all the steps from phase 4 to phase 6 can be repeated. The output will be monitored. Further research and analysis will be done based on the output and the outcome will be noted for further improvisation of the system.

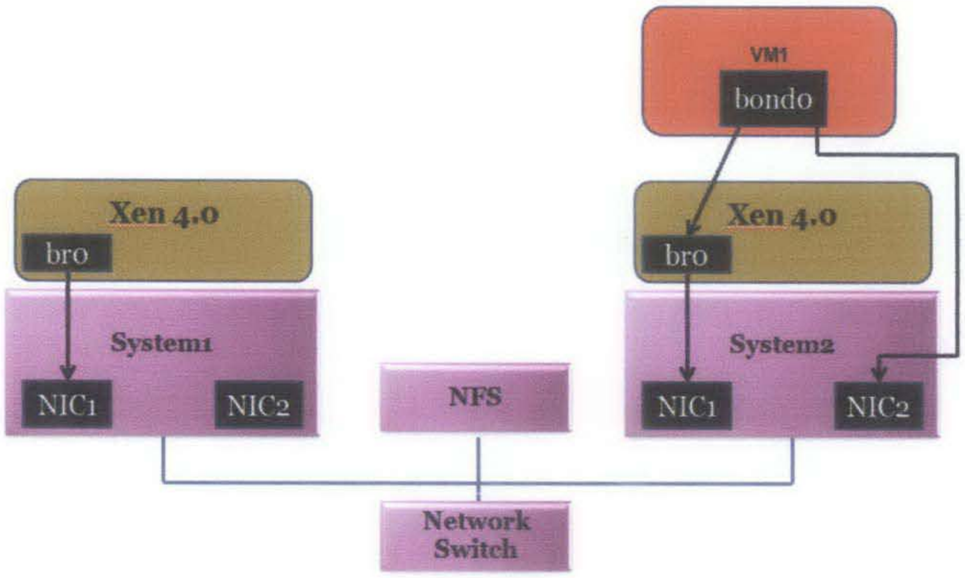


Figure 15: Complete Structure After Network Bonding

3.8 PHASE 8: SINGLE ROOT I/O VIRTUALIZATION (SR/IOV)

SR-IOV should be implemented with the 10Gb Ethernet cards replacing the existing 1Gb Ethernet cards. However, since the high cost of 10Gb Ethernet cards defers from obtaining it, the same stimulation will be done to the 1Gb card as a prove of concept. This part of the project will divide the 10Gb capacity to 1Gb each and assigned directly to the Virtual Machines. Thus, each of the Virtual Machines will have continuous network connectivity and each of the Virtual Machines can be accessed concurrently. Each Virtual Machine also will be able to have near-native bandwidth of 1Gb.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 PROJECT DEVELOPMENT

4.1.1. Phase 1: Setup: Hardware Requirement

The hardware setup of this project is to simulate the real-world environment. A simple network system is created with two server systems, System1 and System2, a Network File System (NFS) and also a Control System. All these systems are connected to a network switch and given static IP address to create a single network. The real-world environment will be more complex with many server systems and clients requesting for the service from the server within a very large network.

4.1.1.1Server Systems

Both the server system, System1 and System2 must be VT-d enabled. VT-d is Intel Virtualization Technology for Directed I/O. A general requirement for I/O virtualization model is the ability to isolate and restrict device accesses to the resources owned by the Virtual Machine. Thus, to implement this, the server systems must be of Intel's processor and both of the systems must be identical. However, not all Intel processors are VT-d enabled. Thus, before beginning with this project, it is vital to ensure that the system is VT-d enabled. The list of Intel processors that supports VT-d is included in Appendix C. AMD processor also has I/O virtualization which is the AMD-Vi which provides the same function.

4.1.1.2 Network File System (NFS)

As for the NFS, it allows a system to share directories and files with others over a network. By using NFS, the server systems can access files on remote systems almost as if they were local files. One of the notable benefits that NFS can provide is the server systems can use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network. This attributes becomes very important for live migration that will be implemented in this project to verify the continuous and concurrent network connectivity. NFS servers will be used as a storage repository for all the Virtual Machine (VM) images and other video files that will be used throughout live migration and video streaming. When VMs are stored on non-shared storage, they cannot be live migrated between systems. With non-shared storage, the VMs are stored in that storage can only be seen by one virtualization server. Shared remote storage such as NFS allows a VM to be migrated to multiple virtualization servers.

4.1.1.3 Control System

The control system is only an additional system that functions as central system that manages all the other systems in the network. This is useful to ease the project implementation.

4.1.1.4 Additional Network Interface Card

For this system, the Network Interface Card (NIC) must be Single Root I/O Virtualization (SR-IOV) enabled. The list of SR-IOV enabled NIC is listed in Appendix D. The SR-IOV is fairly a new technology to the market. Thus, the integrated NIC that comes with the machines or server systems are not SR-IOV enable. Thus, this additional NIC has to be inserted to use the SR-IOV function.

4.1.2. Phase 2: Setup: Software Requirement

The machines are installed with Linux Operating Systems. The main reason why Linux is used is that it can be downloaded for free as they are open source. Another reason is that the operating system is flexible and can be easily rendered as the Linux source codes are freely distributed.

4.1.2.1 Server System with OpenSUSE 11.2

Both the server systems, System1 and System2 are installed with OpenSUSE 11.2 operating system. The main reason for choosing OpenSUSE 11.2 is that it has Xen 3.4 hypervisor that comes with it. Enabling this Xen hypervisor is simple and straightforward using OpenSUSE 11.2 with a click of a button. The Xen hypervisor is then upgraded to 4.0 as the performance of the VM is smoother and better. The I/O such as the mouse pointer movement and keyboard input lags in the VM with Xen 3.2. The video streaming also shows that the video is not smooth but shuddering instead. All these problems can be overcome with upgrading the hypervisor to Xen 4.0. Another option is that OpenSUSE 12.1 is recently announced and it supports Xen 4.1 hypervisor.

4.1.2.2 Other Systems

The requirements of the other systems are more flexible. Any operating systems can be installed. As for the NFS system, RedHat Enterprise Linux 5 (RHEL5) is used as it supports NFS. Other operating systems that support NFS can also be used. The Central System can be installed with any operating system and not necessarily of Linux. The VM images are stored in the NFS. The VM can be of any Linux operating system.

4.1.3. Phase 3: Setup: Creating the Virtual Machine

At this point, only one VM is created to verify the first objective which is enabling continuous network connectivity. The VM is created in either System1 or System2 by editing the .hvm file which is the guest configuration file that is stored in the NFS. This .hvm points to the image of the operating system that is used as the VM. Sample .hvm file can be referred to Appendix B.

4.1.3.1 Xen hypervisor.

For this project, the Xen hypervisor is chosen. Xen is the most suitable virtualization approach for this system. As explained earlier, there are many types of virtualization approach. The hypervisor virtualization was chosen that implements hardware virtualization. The hardware virtualization is enabled by the Intel VT and also AMD-V which is required to perform PCI or Network Card passthrough.

4.1.4. Phase 4: Live Migration

The capability of virtual machine live migration brings benefits such as improved performance, manageability and fault tolerance, while allowing workload movement with a short service downtime. Moving the contents of a VM's memory from one physical host to another can be approached in any number of ways. However, when a VM is running a live service such as video streaming, it is important that this transfer occurs in a manner that balances the requirements of minimizing both downtime and total migration time [20]. The former is the period during which the service is unavailable due to there being no currently executing instance of the VM; this period will be directly visible to clients of the VM as service interruption. The latter is the duration between when migration is initiated and when the original VM may be finally discarded and hence, the source host may potentially be taken down for maintenance, upgrade or repair.

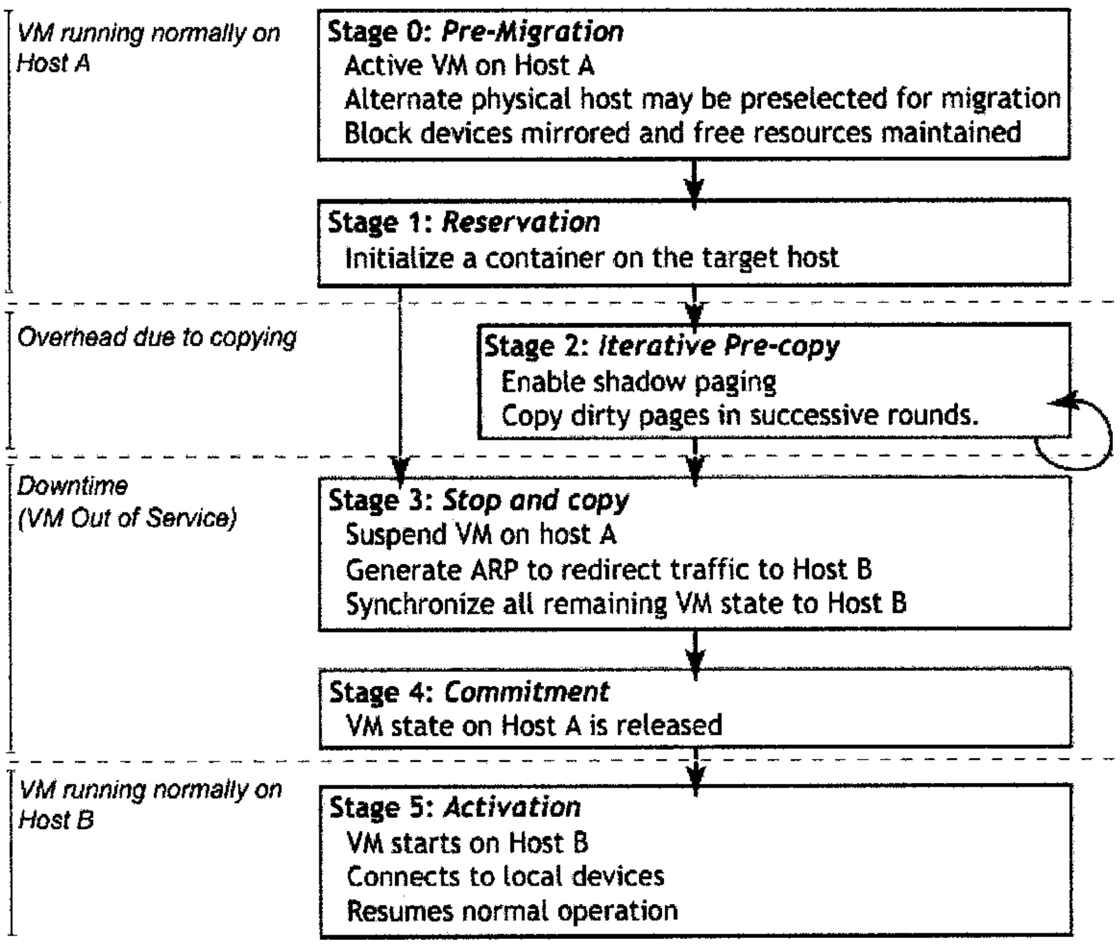

VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime
(VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
Resumes normal operation

Figure 16: Migration Timeline [21]

$$\begin{aligned} &Total\ Migration\ Time \\ &\quad = Initialisation + Reservation + \sum_i Pre - copy_i + Stop - and \\ &\quad - copy + Commitment + Activation \end{aligned}$$

Where, Initialization + Reservation is the Pre-migration Overhead, and Commitment + Activation is the Post-migration Overhead.

$$Total\ Downtime\ = Stop - and - copy + Commitment + Activation$$

Where, Commitment + Activation is the Post-migration Overhead.

4.1.4.1 Upgrading Xen 3.2 to Xen 4.0

The Xen performance with Xen 3.2 is lower compared to Xen 4.0. The difference can be seen in the lagging of I/O input such as mouse keystroke input and keyboard input. The video performance also varies between Xen 3.2 and Xen 4.0. The total live migration time and total downtime for live migration of VM is showed in the table below:

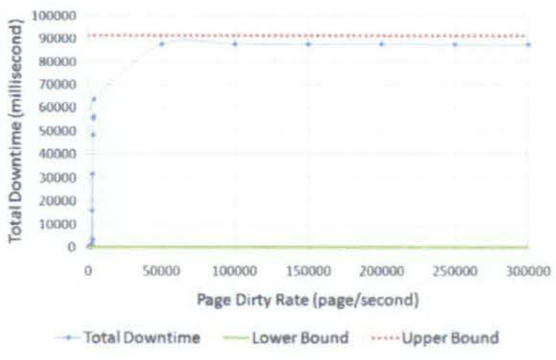
Measurement	Total Migration Time	Total Downtime
Xen 3.2	**60 sec**	**4 sec**
Xen 4.0	**32 sec**	**<1 sec**

Table 1: Xen 3.2 and Xen 4.0 Live Migration Performance

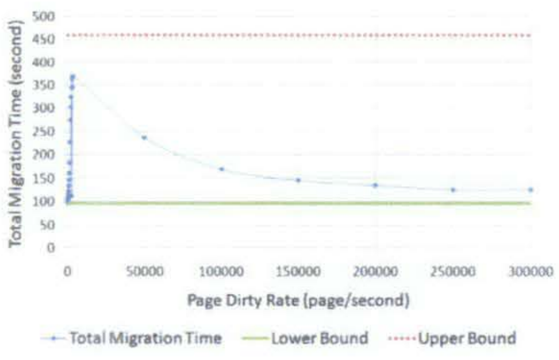
The table above shows the improvement in Total Migration Time and Total Downtime improvement for Xen 4.0. The total Migration Time is, 32 seconds, almost half of the time required for Xen 3.2. The Total Downtime is less than 1 second which is almost not visible to the client. The Total Downtime is measured with video streaming and counting the time that the video pauses before resuming and continues to play. The Total Downtime or service unavailability for Xen 4.0 is less than 1 second which is almost not visible.

4.1.4.2 Parameters Affecting Migration

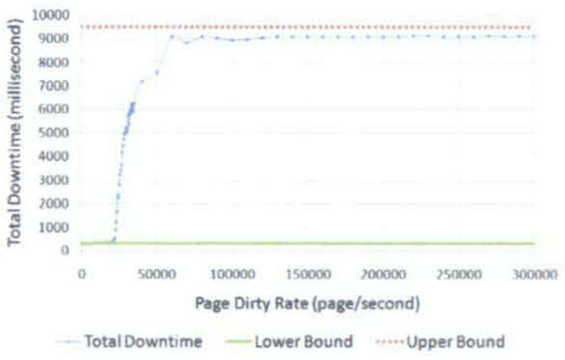
There are two major factors that need to be studied for a good migration modeling. The factors that impact the total migration time and downtime are migration link bandwidth and page dirty rate. Migration link bandwidth is the most influential parameter as link capacity us inversely proportional to total migration time and downtime. Whereas, the page dirty rate is the rate at which memory pages in the VM are modifies which, in turn, directly affects the number of pages that are transferred in each of the pre-copy iteration. Higher page dirty rates results in more data being sent per iteration which leads to longer total migration time. On top of that, higher page dirty rates results in longer VM downtime as more pages need to be sent in the final transfer round in which the VM is suspended. The Figure 17 shows the effect of varying the page dirty rate on total migration time and downtime for each link speed with static VM size = 1024 MB.



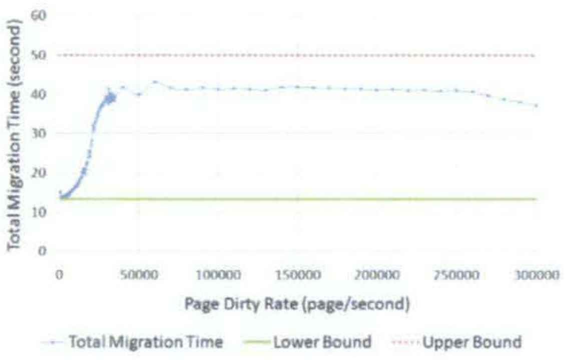
(a) 100 Mbps Total Downtime



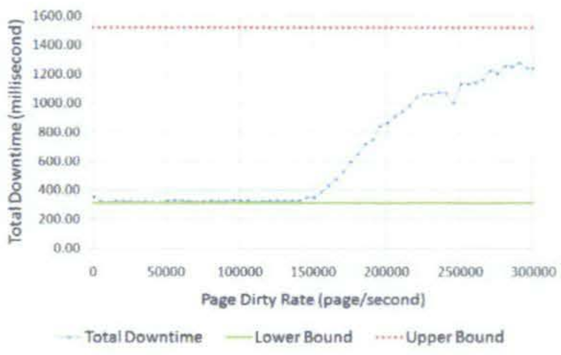
(b) 100 Mbps Total Migration Time



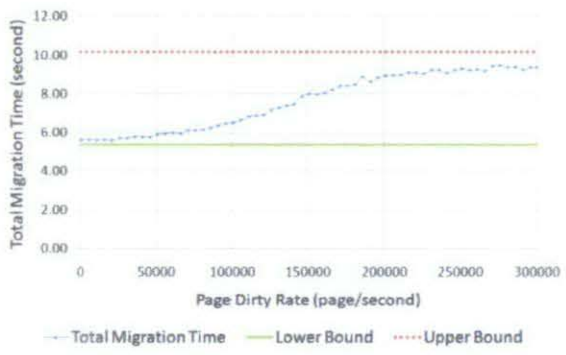
(c) 1 Gbps Total Downtime



(d) 1 Gbps Total Migration Time



(e) 10 Gbps Total Downtime



(f) 10 Gbps Total Migration Time

Figure 17: Link Capacity and Migration Performance [21]

The Figures 17 explains the choice of Network Interface Card of 1Gbps for the host System1 and System2 which is also the integrated Network Interface card. The Figure shows the effect of varying the page dirty rate on total migration time and downtime for each link speed. If the page dirty rate is below link capacity, the migration sub-system is able to transfer all modified pages in a timely fashion, resulting in a low total migration time and downtime. On the other hand, if the page dirty rate starts approaching link capacity, migration performance degrades significantly.

4.1.5. Phase 5: VT for Directed I/O (VT-d) PCI passthrough

Virtualization allows the creation of multiple virtual machines on a single server. This consolidation maximizes server hardware utilization, but server applications require a significant amount of I/O performance. Software based I/O virtualization methods use emulation of the I/O devices. With this emulation layer the VMM provides a consistent view of a hardware device to the VMs and the device can be shared amongst many VMs. However, it could also slow down the I/O performance devices. Thus, VT-d can provide solution for the loss of native performance or of native capability of a virtualized I/O device by directly assigning the device to a VM. By doing this, VM is able to have full control over a I/O without going through the VMM, in other word, direct access.

4.1.5.1 Requirements for Xen PCI passthru to HVM guest [22]

Hardware IOMMU (Intel VT-d or AMD IOMMU) is required from the CPU/motherboard/chipset/BIOS. To verify you have IOMMU support enabled:

- Check if IOMMU (Intel VT-d or AMD IOMMU) is enabled in the system BIOS. Some BIOSes call this feature "IO virtualization" or "Directed IO".
- If running Xen 3.4.x (or older version) it is required to add iommu=1 flag (or vtd=1 in even older versions) for Xen hypervisor (xen.gz) to grub.conf and reboot.

- Xen 4.0.0 and newer versions enable IOMMU support as a default if supported by the hardware and BIOS, no additional boot flags required for the hypervisor.
- "xm dmesg" or the Xen hypervisor boot messages can be used to check if "IO virtualization" gets enabled.

4.1.5.2 VT-d for Continuous Network Connection

For this project, VT-d is implemented to the I/O which is also a PCI device, the Network Interface Card (NIC). By directly assigning this NIC to the VM, the VM will have full control over the NIC without disruption as the particular NIC is hidden from Xen hypervisor, the host system and also other VMs. Two network cards are used; the integrated NIC is assigned to the Host Operating System, the additional NIC is assigned to the VM. This enables the VM to have a dedicated NIC and does not require sharing the NIC with other VM. If the NIC is being shared by multiple VMs, the NIC allocation will be based on certain scheduling system. Thus, the each VM will not have a continuous network connection as it has to wait for its turn to be allocated with the resources based on the scheduling system.

The list of the NIC (with two ports each) in the Host Operating System of System1 before PCI passthrough will be as the following:

```
01:00.0 Ethernet controller: Intel Corporation 82575EB Gigabit Network Connection (rev 02)
01:00.1 Ethernet controller: Intel Corporation 82575EB Gigabit Network Connection (rev 02)
03:00.0 Ethernet controller: Intel Corporation 82571EB Gigabit Ethernet Controller (rev 06)
03:00.1 Ethernet controller: Intel Corporation 82571EB Gigabit Ethernet Controller (rev 06)
```

After PCI passthrough, the NIC with the PCI ID of 03:00.0 and 03:00.1 will be hidden from the Host Operating System of System1:

```
01:00.0 Ethernet controller: Intel Corporation 82575EB Gigabit Network Connection (rev 02)
01:00.1 Ethernet controller: Intel Corporation 82575EB Gigabit Network Connection (rev 02)
```

This PCI or NIC is then assigned directly to the VM for full access.

4.1.6. Phase 6: Network Bonding

By achieving network bonding, two main problems can be addressed: bandwidth limitations, and lack of redundancy. Since there is two NIC in both System1 and System2, both of these NIC can be bonded together to form a single connection that improves the bandwidth.

The iPerf testing shows the following performance with network bonding and the additional NIC is attached to the VM and also without.

4.1.6.1 Improved Bandwidth with Network Bonding

iPerf is a tool to measure the bandwidth and the quality of a network link. By default, the iPerf client connects to the iPerf server on the TCP port 5001 and the bandwidth displayed by iPerf is the bandwidth from the client to the server. As for the project, System1 performed as the Server while VM performed as the Client. The results are showed in the table below:

	WITHOUT NETWORK BONDING				WITH NETWORK BONDING			
	Server		Client		Server		Client	
	Transfer	Bandwidth	Transfer	Bandwidth	Transfer	Bandwidth	Transfer	Bandwidth
TEST1	*36.7 MBytes*	*30.8 Mbits/sec*	*36.7 MBytes*	*30.8 Mbits/sec*	*1.09 GBytes*	*936 Mbits/sec*	*1.09 GBytes*	*936 Mbits/sec*
TEST2	*30.3 MBytes*	*27.6 Mbits/sec*	*30.3 MBytes*	*27.6 Mbits/sec*	*1.10 GBytes*	*940 Mbits/sec*	*1.10 GBytes*	*940 Mbits/sec*

Table 2: Bandwidth Results for Network Bonding through iPerf Testing

Below are the sample output obtained through iPerf:

The iPerf result **without** network bonding:

On VM (client):

```
------------------------------------------------------------
Client connecting to 192.168.1.11, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 192.168.1.61 port 37201 connected with 192.168.1.11 port 5001
[ID]    Interval        Transfer Bandwidth
[ 3]    0.0-10.0sec          36.7 MByte     30.8 Mbits/sec
```

On System1 (server):

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
 [ 4] local 192.168.1.11 port 5001 connected with 192.168.1.61 port 37201
[ID]    Interval        Transfer Bandwidth
[ 4]    0.0-10.0sec          36.7 Mbytes    30.8 Mbits/sec
```

The iPerf result **with** network bonding:

On VM (client):

```
------------------------------------------------------------
Client connecting to 192.168.1.11, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[ 3] local 192.168.1.61 port 37201 connected with 192.168.1.11 port 5001
[ID]    Interval        Transfer Bandwidth
[ 3]    0.0-10.0sec         1.09 GBytes     936 Mbits/sec
```

On System1 (server):

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
 [ 4] local 192.168.1.11 port 5001 connected with 192.168.1.61 port 37201
[ID]    Interval        Transfer Bandwidth
[ 4]    0.0-10.0sec         1.09 Gbytes     936 Mbits/sec
```

4.1.7. Phase 7: Live Migration with NIC Passthrough (VT-d) and Network Bonding

This step is a proof of concept to show that live migration for VM is possible without continuous network connectivity. For live migration to happen, the NIC that is attached to the VM has to be disconnected before beginning the live migration. Thus, the Network bonding becomes important. When the Master NIC is detached (also the additional NIC that is attached to the VM), the Slave NIC will take over and begin the live migration. This shows that there is a network connection throughout the live migration.

4.1.7.1 Video Streaming to Show Continuous Network Connectivity

In order to visually see the continuous network connection for the VM, a video streaming is performed. A high-definition video that is stored in the NFS is streamed to the VM. The dedicated NIC is detached, live migration is performed, and the NIC is attached again in the destination system. While all these are being performed, the video streaming continues to play without disruption. The Figure below shows that video streaming is being performed while live migration without being interrupted.

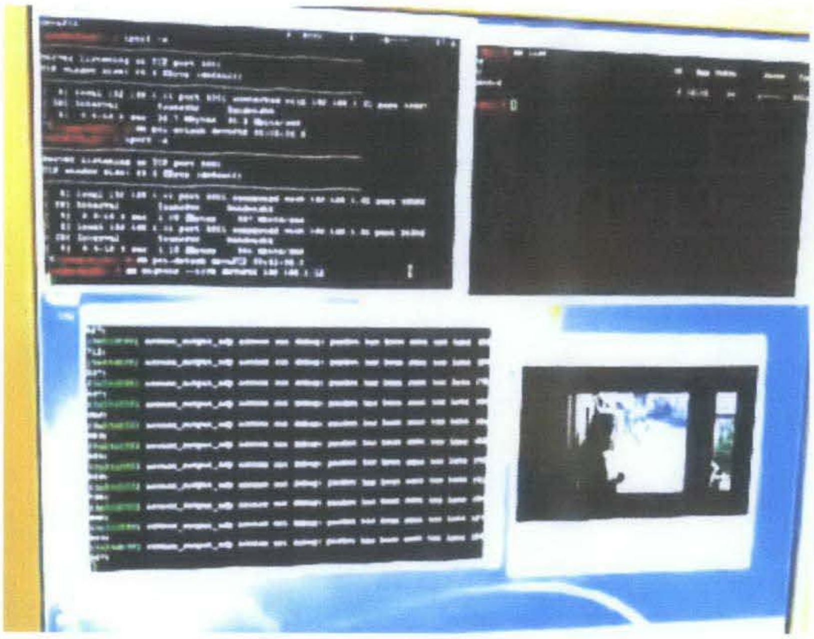


Figure 18: Video Streaming with Live Migration

4.1.8. Phase 8: Single Root I/O Virtualization (SR-IOV)

The NIC passthrough shows that a NIC can be assigned directly to a VM. However, in a situation where multiple VMs available, it is not practical to add multiple physical NIC. For example, if the server system has the capability to rum 10 VMs and all the VM requires continuous network connection at the same time, it is not practical to add 10 additional NIC and assign each to the VM. Therefore, SR-IOV is used.

4.1.8.1 SR-IOV for Concurrent Network Connectivity

SR-IOV is a specification that allows a PCIe device to appear to be multiple separate physical PCIe devices. Thus, by only adding only one NIC, all the 10 VMs in the example above can be assigned directly to a virtual NIC. However, not all NIC supports SR-IOV. This is because SR-IOV is a fairly new technology to the market. Most of the NIC that supports SR-IOV are of 10Gbps bandwidth [23]. There is only one 1Gbps bandwidth NIC in the market currently. This portion of the project is unable to be developed due to the difficulties to find the SR-IOV enabled NIC in the market. The 10Gbps NIC is required so that it can be divided into 1Gbps Virtual Functions (VF) and ensures good network performance. Further testing will be conducted after the implementation of SR-IOV [24].

4.1.8.2 Testing with SR-IOV implemented

First of all, 10 VMs will be created into a single system, System1 and the live migration with video streaming will be conducted for each of the VM concurrently. The iPerf testing will be conducted to verify the bandwidth for each VM's network connection throughout the live migration and also before and after the completion of live migration.

The next test is the LMBench test. LMBench was developed specifically to measure the performance of core kernel system calls and facilities, such as file access, context switching, and memory access. LMBench has been particularly effective at establishing and maintaining excellent performance in these core facilities in the Linux kernel. LMBench test will be useful to verify the overall performance of the host operating systems as well as all the VMs.

Another test that will be conducted is using the traffic generator the generate network traffic and test the network performance. An example of this software is the Network Traffic Generator and Monitor. This software was designed to generate and monitor IP/ICMP/TCP/UDP traffic from clients to servers to stress test routers, servers and firewalls under extreme network loads. It is a very simple and fast program which can emulate true client/server activity. This software also has the ability to create network traffic so that a true measurement can be taken by our network monitoring tool or external tools.

4.2 PROTOTYPE

4.2.1 Complete System Prototype

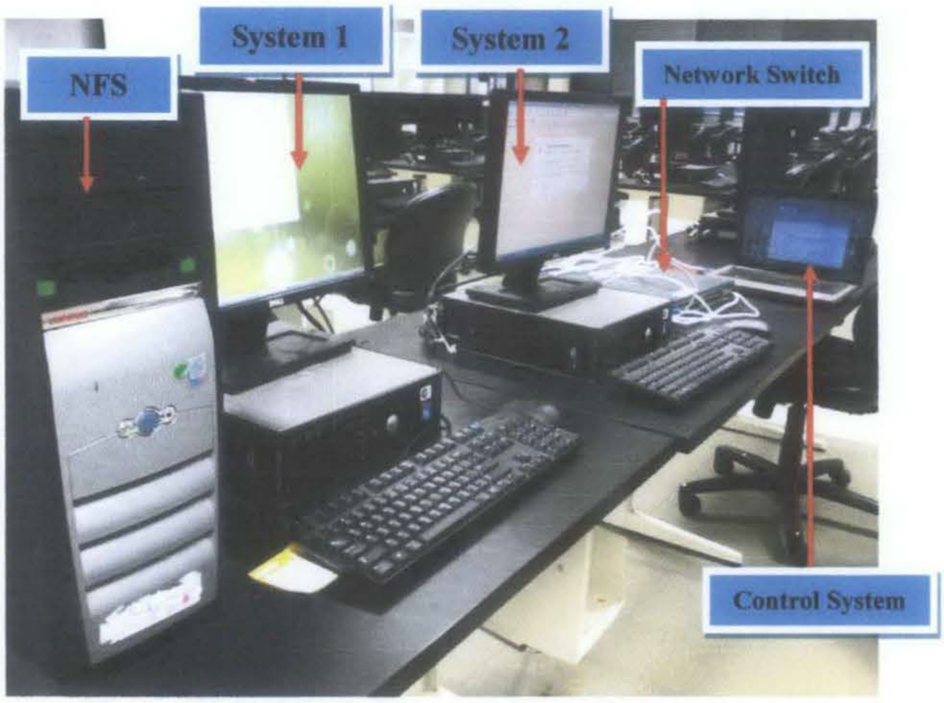


Figure 19: The hardware setup

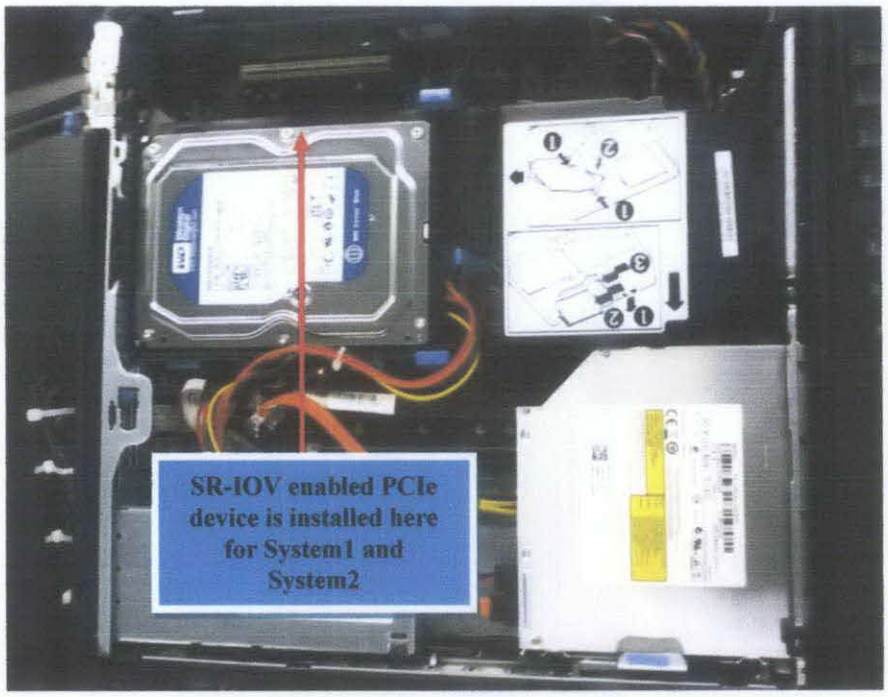

SR-IOV enabled PCIe device is installed here for System1 and System2

Figure 20: The PCIe Slot for SR-IOV enabled NIC

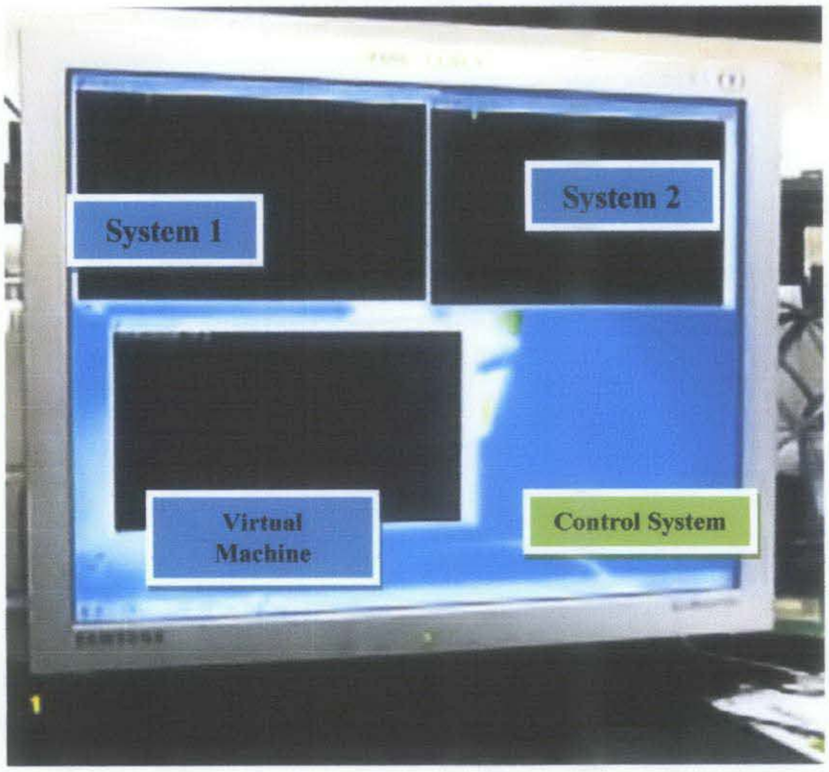


Figure 21: The Control System That Accesses System1, System2 and VM

4.3 LIMITATION AND CHALLENGES

4.3.1. Hardware Limitation

One of the significant limitations for this project is the hardware limitation. The virtualization technology, VT-d and SR-IOV are new technology in the market. The first challenge was finding a system or machine that has processor with VT-d enabled for System1 and System2. However, not all the new processors support the VT-d. It has to be Intel processor as well. Another hardware limitation is the SR-IOV enabled NIC. This NIC could not be purchased as it is not easily available in the market. This NIC has to be book and purchased from a particular vendor. This can be time consuming and also costly. This limitation is beyond control, thus alternative action will be taken or will be implemented in the future.

4.3.2. Other Limitation

Other limitation also includes related to the internet connection which has proxy enabled with username and password. Resolving this matter was time consuming, however was resolved with the help of the lab technician. Another limitation is the VT-d enabled machines are limited in the laboratory and accesses to these machines are constrained to a short period of time. This causes limited time for the project to be developed and gaining the output.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

In conclusion, virtualization in cloud computing has a lot of room for improvement. If the proposed project is implemented to the cloud computing environment, the functions can be optimized and resource allocation can be done without any network disruption. Live migration is crucial in virtualization as well as network connectivity. The tests conducted such as the live migration with video streaming shows that the continuous network connectivity for virtual machines can be achieved while live migration. The main contribution of this project will be the near-native internet connectivity which is yet to be achieved with the current virtualization and cloud computing. This was proved with the iPerf testing that was done after PCI passthrough and network bonding. However, the concurrent network connectivity is yet to be implemented due to hardware support and other hardware limitations which are required for this part of the project to be implemented.

5.2 FURTHER ENHANCEMENTS

Virtualization, cloud computing and SR-IOV is a booming technology that has received the attention of the Information Technology industry. The hardware limitation related to these technologies will be eliminated in near future. Thus, this provides an easier implementation of these technologies and also to gain the advantage from it. The SR-IOV is not able to be implemented in the project currently due to hardware and cost limitation. Once SR-IOV is implemented, the continuous and concurrent network connectivity can be achieved.

On top of that, the completion of this project faced some difficulties due to issues related to hardware compatibility. Therefore, in-depth research on hardware support for SR-IOV, VT-d and also other Virtualization and Cloud Computing requirements will be done. Through this thorough research, the compiled list of hardware requirements, restrictions and most suitable hardware will be listed.

The future planning for this project is to implement it into a various Cloud Computing environment such as the Infrastructure as a Service architecture, Platform as a Service architecture or Software as a Service architecture.

REFERENCES

[1] Buyya R., Chee S. Y., Venugopal S., Broberg J., Brandic I.. "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility". *Future Generation Computer Systems*, 2008, p599-616.

[2] Zhang Q., Cheng L., "Boutaba R.. Cloud computing: state-of-the-art and research challenges". *J Internet Serv Appl*, 2010, 1, p7-18.

[3] R.J. Creasy, "The Origin of the VM/370 Time Sharing System," *IBM Journal of Research and Development*, vol. 25, no. 5, p. 483, 1981.

[4] Rose R.. "Survey of System Virtualization Techniques". Available: http://robertwrose.com/vita/rose-virtualization.pdf. Last accessed 20th February 2011, March, 2004.

[5] Uhlig, R.; Neiger, G.; Rodgers, D.; Santoni, A.L.; Martins, F.C.M.; Anderson, A.V.; Bennett, S.M.; Kagi, A.; Leung, F.H.; Smith, L. (2005). "Intel Virtualization Technology", 38(5), 48-56

[6] P. Barham et al., "Xen and the Art of Virtualizatio". *Proc. 19th ACM Symp. Operating Systems Principles*, ACM Press, 2003, pp. 164-17

[7] Juan C. Dueñas, José L. Ruiz, Félix Cuadrado, Boni García, Hugo A. Parada G., "System Virtualization Tools for Software Development," *IEEE Internet Computing*, vol. 13, no. 5, pp. 52-59, Sep./Oct. 2009, doi:10.1109/MIC.2009.115

[8] Li S., Hao Q., Xiao L., Xu Q.. "Optimizing Network Virtualization in Kernel-based Virtual Machine". *The 1st International COnference on Information Science and Engineering (ICISE2009)* , 2009, p282-285.

[9] Stage A., Setzer T.. "Network-aware migration control and scheduling of differentiated virtual machine workloads". *ISE'09 Workshop.* May 2009, p9-14,.

[10] O. Cherkaoui, and E. Halima, "Network Virtualization Under User Control". *International Journal of Network Management*, March/April 2008, Vol. 18, No.2, pp.147-158.

[11] Zhang J., Li X., Guan H.. "The Optimization of Xen Network Virtualization". *2008 International Conference on Computer Science and Software Engineering*, 2009, p431-437.

[12] Aust S., Kim J., Davis P., Yamaguchi A., Obana S.. "Evaluation of Linux Bonding Features". *Institute of Electrical and Electronic Engineers (IEEE)*, 2008.

[13] T. Anderson, L. Peterson, S. Shenker, and J. Turner. "Overcoming the Internet Impasse through Virtualization", *COMPUTER*, 2005,. pp. 34–41.

[14] Jebala M., Letaifa A. B., Tabbane S.. "A Survey of Live Migration in Virtual Network Environment (VNE)". *Institute of Electrical and Electronic Engineers (IEEE)*, 2010, p351-354

[15] Lui J.. "Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support". *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium*, 2010, p1-12.

[16]Lou Y.. (July 2010). Network I/O Virtualization for Cloud Computing. computer.org/ITPro. 12 (5), p36-41.

[17]Dong Y., Xu D., Zhang Y., Lioa G.. "Optimizing Network I/O Virtualization with Efficient Interrupt Coalescing and Virtual Receive Side Scaling". *2011 IEEE International Conference on Cluster Computing*, 2011,. p26-34.

[18]Pu X., Liu M., Jin J., Cao Y.. "A Modeling of Network I/O Efficiency in Xen Virtualized Cloud". *Electronics, Communications and Control (ICECC), 2011 International Conference.*, 2011, p1831-1834.

[19]Dong Y., Yang X., Li X., Li J., Tian K., Guan H.. "High Performance Network Virtualization with SR-IOV". *Institute of Electrical and Electronic Engineers (IEEE)*, 2009.

[20]Jin H., Gao W., Wu S., Shi X., Wu X., Zhou F.. "Optimizing the live migration of virtual machine by CPU scheduling". *Journal of Network and Computer Appication,* 2010. p1-9.

[21]Liu Z., Qu W., Liu W., Li K.. "Xen Live Migration with Slowdown Scheduling Algorithm". *The 11th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2010, p215-221.

[22] Akoush S., Sohan R., Rice A., Moore A.W., Hopper A.. "Predicting the Performance of Virtual Machine Migration". *2010 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2010, (13), p37-46.

[23] Sang F. L., Lacombe E., Nicomette V., Deswarte Y.. "Exploiting an I/OMMU vulnerability". *2010 5th International Conference on Malicious and Unwanted Software*, 2010, p7-14..

[24] Suzuki J., Hidaka Y., Higuchi J., Baba T., Kami N., Yoshikawar T.. (2010). "Multi-Root Share of Single-Root I/O Virtualization (SR-IOV) Compliant PCI Express Device". *2010 18th IEEE Symposium on High Performance Interconnect*, 2010, p25-31.

APPENDICES

APPENDIX A: NETWORK BONDING MODE

mode=0 (balance-rr)
Round-robin policy: Transmit packets in sequential order from the first available slave through the last. This mode provides load balancing and fault tolerance.

mode=1 (active-backup)
Active-backup policy: Only one slave in the bond is active. A different slave becomes active if, and only if, the active slave fails. The bond's MAC address is externally visible on only one port (network adapter) to avoid confusing the switch. This mode provides fault tolerance. The primary option affects the behavior of this mode.

mode=2 (balance-xor)
XOR policy: Transmit based on [(source MAC address XOR'd with destination MAC address) modulo slave count]. This selects the same slave for each destination MAC address. This mode provides load balancing and fault tolerance.

mode=3 (broadcast)
Broadcast policy: transmits everything on all slave interfaces. This mode provides fault tolerance.

mode=4 (802.3ad)
IEEE 802.3ad Dynamic link aggregation. Creates aggregation groups that share the same speed and duplex settings. Utilizes all slaves in the active aggregator according to the 802.3ad specification.

Pre-requisites:
1. Ethtool support in the base drivers for retrieving the speed and duplex of each slave.
2. A switch that supports IEEE 802.3ad Dynamic link aggregation.
Most switches will require some type of configuration to enable 802.3ad mode.

mode=5 (balance-tlb)
Adaptive transmit load balancing: channel bonding that does not require any special switch support. The outgoing traffic is distributed according to the current load (computed relative to the speed) on each slave. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed receiving slave.

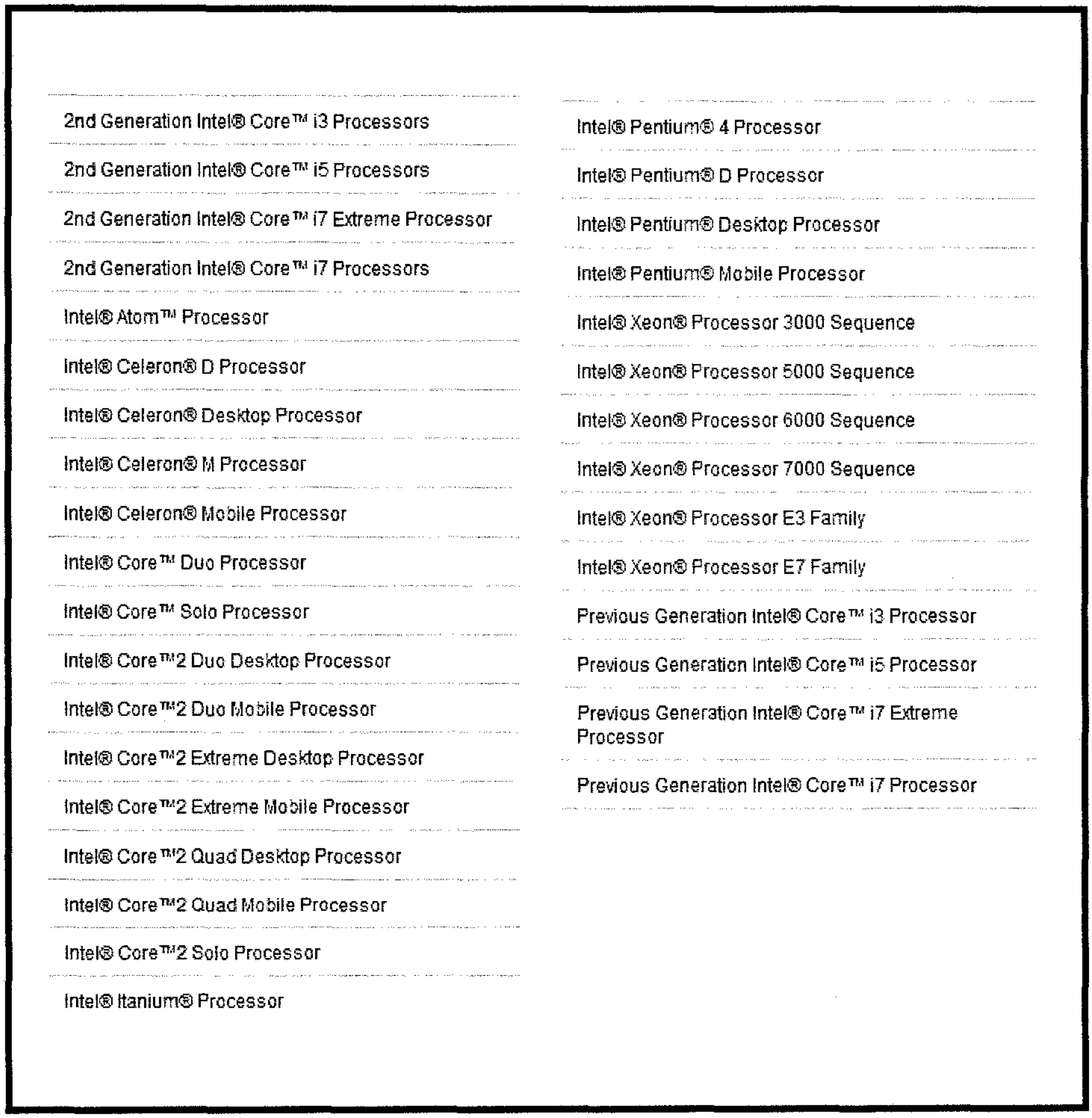
Prerequisite:
Ethtool support in the base drivers for retrieving the speed of each slave.

mode=6 (balance-alb)
Adaptive load balancing: includes balance-tlb plus receive load balancing (rlb) for IPV4 traffic, and does not require any special switch support. The receive load balancing is achieved by ARP negotiation. The bonding driver intercepts the ARP Replies sent by the local system on their way out and overwrites the source hardware address with the unique hardware address of one of the slaves in the bond such that different peers use different hardware addresses for the server.

APPENDIX B: SAMPLE .hvm CONFIGURATION FILE

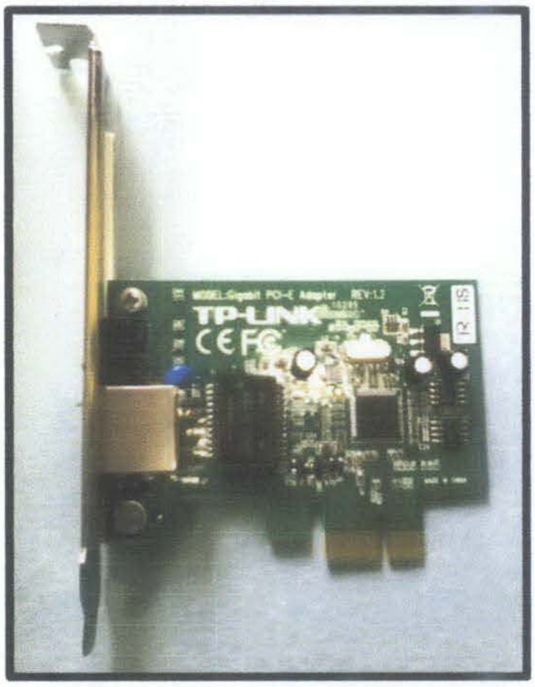
```
# vi vm_config_file.hvm
import os, re
arch = os.uname()[4]
if re.search('64', arch):
    arch_libdir = 'lib64'
else:
    arch_libdir = 'lib'
kernel = "/usr/lib/xen/boot/hvmloader"
builder='hvm'
memory = 512
shadow_memory=8
name = "myGuestOS_name"
vcpus=2
vif = [ 'type=ioemu, mac=00:16:3e:00:34:11, bridge=br0' ]
disk = [ 'file:/mnt/NFS /ia32e_fc10.img,hda,w', ',hdc:cdrom,r' ]
device_model = '/usr/' + arch_libdir + '/xen/bin/qemu-dm'
opengl=1
vnc=1
vncpasswd=''
stdvga=0
serial='pty'
usbdevice='mouse'
```

APPENDIX C: VT-d ENABLED INTEL PROCESSOR



- 2nd Generation Intel® Core™ i3 Processors
- 2nd Generation Intel® Core™ i5 Processors
- 2nd Generation Intel® Core™ i7 Extreme Processor
- 2nd Generation Intel® Core™ i7 Processors
- Intel® Atom™ Processor
- Intel® Celeron® D Processor
- Intel® Celeron® Desktop Processor
- Intel® Celeron® M Processor
- Intel® Celeron® Mobile Processor
- Intel® Core™ Duo Processor
- Intel® Core™ Solo Processor
- Intel® Core™2 Duo Desktop Processor
- Intel® Core™2 Duo Mobile Processor
- Intel® Core™2 Extreme Desktop Processor
- Intel® Core™2 Extreme Mobile Processor
- Intel® Core™2 Quad Desktop Processor
- Intel® Core™2 Quad Mobile Processor
- Intel® Core™2 Solo Processor
- Intel® Itanium® Processor
- Intel® Pentium® 4 Processor
- Intel® Pentium® D Processor
- Intel® Pentium® Desktop Processor
- Intel® Pentium® Mobile Processor
- Intel® Xeon® Processor 3000 Sequence
- Intel® Xeon® Processor 5000 Sequence
- Intel® Xeon® Processor 6000 Sequence
- Intel® Xeon® Processor 7000 Sequence
- Intel® Xeon® Processor E3 Family
- Intel® Xeon® Processor E7 Family
- Previous Generation Intel® Core™ i3 Processor
- Previous Generation Intel® Core™ i5 Processor
- Previous Generation Intel® Core™ i7 Extreme Processor
- Previous Generation Intel® Core™ i7 Processor

APPENDIX D: SAMPLE Gigabit PCIe NETWORK ADAPTER/CARD

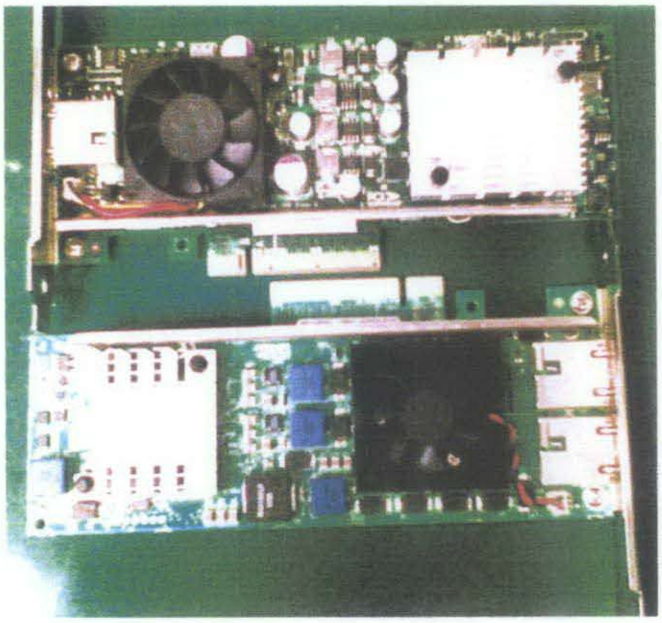


TP-LINK Gigabit PCIe Network Adapter (Model No. TG-3468)

APPENDIX D: SR-IOV ENABLED PCIe DEVICES

Intel ® Ethernet Server Adapter X520-DA2
Intel ® Ethernet Server Adapter X520-SR1
Intel ® Ethernet Server Adapter X520-SR2
Intel ® Ethernet Server Adapter X520-LR1
Intel ® Ethernet Server Adapter X520-T2
Intel ® Gigabit ET Dual Port Server Adapter
Intel ®Gigabit ET2 Quad Port Server Adapter

List of Intel SR-IOV enabled PCIe Devices



SR-IOV enabled PCIe Network Card (Intel ® Gigabit ET Dual Port Server Adapter)

APPENDIX F: PROJECT KEY MILESTONE

Tasks	FYP1 (week)					FYP2 (week)									
	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10
Research - Comparing Journals	■	■													
Research - Hardware			■												
Research - Software				■											
Final Design					■										
System Setup - Hardware						■	■	■	■	■					
System Setup - Network											■				
System Setup - Software												■			
Live Migration Implementation													■		
VT-d (PCI passthrough)													■		
Network Bonding														■	
SR-IOV															■
Final Review & Improvement															■