

# **Clustering of Boot Servers for High Availability in Linux Compute Clusters using Kerrighed**



Submitted by:

**Ali Muhammad**

Supervised by:

**Dr Athar Mahboob**

THESIS

Submitted to:

Department of Electronics and Power Engineering,  
Pakistan Navy Engineering College Karachi,  
National University of Sciences and Technology, Islamabad  
In partial fulfillment of the requirements for the degree of  
**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**  
With Specialization in Communication

May 2012

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

In the name of ALLAH,  
The Most Beneficent, the Most Merciful

# Acknowledgement

First of all I thank ALMIGHTY ALLAH for his divine help at every single moment of my life. I am deeply indebted to my supervisor Dr. Athar Mahboob whose constant guidance helped me in successful completion of my research work. I would like to thank my family whose constant inspiration and guidance kept me motivated and focused on my work. My thanks are also extended to the EPE Department of the University for a Nice Atmosphere, kind treatment and support.

At last but not the least, I would like to thank all of Guidance Committee members for their support and guidance.

- Cdr. Dr. M. Farhan PN
- Dr. Arshad Aziz
- Dr. Sameer Qazi

# Abstract

High-availability cluster or failover clusters are collections of computers that empower applications and services that can be unfailingly utilized with a minimum of down-time and elimination of single point of failure (SPOF). For High Performance computing clusters the boot server is an SPOF. We have been able to achieve our original objective which was to remove the single point of failure of any cluster based solution carrying only one boot server. We have successfully setup two nodes high availability cluster for compute cluster's boot server. The boot server and its services are now highly available to outside world and are able to boot any of the cluster nodes through PXE boot. For data synchronization between the boot servers we have used the Distributed Replicated Block Device (DRBD). DRBD is configured on both boot server nodes. The replicated device is formatted with Oracle Cluster File System 2. The latest version of Oracle Clusterware i.e. 11gR2 is installed and configured in this thesis work. Oracle Clusterware is used for high availability of virtual IP so that in case of failure of any one boot server, there should be no downtime in network. We have built the Kerrighed version 2.4.4 which creates a Single Server Image compute cluster based on configured policy. We have performed the required test scenarios to check the High Availability of Clusterware, Kerrighed performance, High Availability of shared disks and boot infrastructure of nodes through PXE boot on network through network card. We present the results of all these tests and provide conclusions and future recommendations.

# Table of contents

Acknowledgement .....	iii
Abstract .....	iv
List of Figures .....	viii
List of Appendices .....	ix
List of Tables .....	x
CHAPTER 1 .....	1
INTRODUCTION .....	1
1.1    Mission.....	1
1.2    Background Study.....	2
1.2    Linux Operating System .....	3
1.4    Clusters and types of Clusters .....	4
1.4.1    High Performance Compute Cluster (Beowulf).....	4
1.4.2    Load Balancing Clusters.....	4
1.4.3    High Availability Clusters .....	5
1.4.4    Storage Clusters .....	5
1.5    The Cluster Building Blocks .....	6
1.5.1    Cluster Nodes.....	7
1.5.2    Cluster Interconnect .....	7
Chapter 2 .....	9
BOOTING INFRASTRUCTURE .....	9
2.1    The Cluster Boot Server .....	9
2.2    Bootting Procedure .....	10
2.2.1    DHCP .....	11
2.2.2    TFTP .....	12
2.3    Setting up Bootting Infrastructure .....	13
2.3.1    Installation of necessary packages .....	13
2.3.2    Amendment of Important Files .....	13
CHAPTER 3.....	14
DRBD, OCFS2 AND NFS SETUP .....	14

3.1	DRBD .....	14
3.1.1	DRBD Features .....	15
3.1.2	Building, Installing and Configuring DRBD .....	15
3.2	OCFS2.....	16
3.2.1	Introduction .....	16
3.2.2	Installation of OCFS2 & OCFS2 tools .....	17
3.2.3	Updating of Configuration Files .....	17
3.2.4	Configure Distributed lock manager (DLM) .....	18
3.3	Configuring NFS .....	18
CHAPTER 4 .....		19
CLUSTERWARE .....		19
4.1	Clusterware.....	19
4.2	Benefits of using Clusterware.....	19
4.3	Real Application Clusterware in our case .....	20
4.4	Preparation for Installation of Clusterware .....	21
4.4.1	Hardware Requirements for Boot Server Nodes .....	21
4.4.2	Network Requirements for Boot Server Nodes .....	21
4.5	Steps for Installation and Configuration of Oracle Clusterware 11gR2.....	22
4.5.1	Installation of Necessary Packages .....	22
4.5.2	Updating of Files.....	22
4.5.3	Adding of Groups and Users .....	22
4.5.4	Creation of Some Directories.....	23
4.5.5	Setting up NFS Share and its Export and Mounting .....	23
4.5.6	Installation of Oracle Clusterware Latest Release .....	23
4.5.7	Guide line for Oracle Installer .....	23
CHAPTER 5 .....		25
SINGLE SYSTEM IMAGE CLUSTER .....		25
5.1	Introduction .....	25
5.2	Mosix .....	26
5.3	OpenMosix .....	26
5.4	OpenSSI .....	26
5.5	Kerrighed.....	26

5.6	Kerrighed Features.....	27
CHAPTER 6.....		28
BREAKING RSA USING MULTI QUADRATIC SEIVE ALGORITHM .....		28
6.1	MULTIPLE POLYNOMIAL QUADRATIC SIEVE.....	28
6.2	Public-Key Cryptosystems .....	28
6.3	RSA.....	29
6.3.1	Key Generation .....	29
6.3.2	Encryption .....	29
6.3.3	Decryption .....	30
6.3.4	RSA Security .....	30
6.4	RSA Numbers.....	30
6.5	Factorization Results.....	31
CHAPTER 7.....		32
TEST SCENARIOS AND RESULTS .....		32
7.1	Storage Cluster Testing.....	33
7.2	High Availability Test on Boot Servers .....	35
	Test 7.2.1 .....	35
	Test 7.2.2 .....	35
7.3	Diskless Boot System Testing .....	36
7.4	Kerrighed Testing .....	39
Summary of High Availability Tests.....		41
RESULTS.....		42
CONCLUSION.....		43
Glossary.....		44
APPENDIX-A .....		45
APPENDIX-B.....		47
APPENDIX-C.....		51
APPENDIX-D .....		55
APPENDIX-E.....		56
REFERENCES .....		59

## List of Figures

Figure 1: Clustering of Boot Server .....	1
Figure 2: Operating System Family Share.....	3
Figure 3: High Availability Cluster.....	5
Figure 4: Storage Cluster.....	6
Figure 5: Generic view of Cluster Components.....	7
Figure 6: Cluster Boot Server Components.....	10
Figure 7: Normal Booting Procedure.....	11
Figure 8: PXE Booting Procedure.....	11
Figure 9: Distributed Replicated Block Device .....	14
Figure 10: Oracle Cluster File System 2 .....	17
Figure 11: Grid Infrastructure Configuration Success .....	24
Figure 12: View of a SMP.....	25
Figure 13: Kerrighed.....	27
Figure 14: Memory Replication (1) .....	33
Figure 15: Memory Replication (2) .....	34
Figure 16: HA of Boot Servers (1).....	35
Figure 17: Diskless Booting Test (1) .....	37
Figure 18: Diskless Booting Test (2) .....	38
Figure 19: Diskless Booting Test (3) .....	38
Figure 20: Multiple msieve processes performing sieving on the Kerrighed cluster .....	40



## List of Appendices

Appendix A: .....	45
Appendix B: .....	47
Appendix C: .....	51
Appendix D: .....	55
Appendix E: .....	56

## List of Tables

Table 1: RSA.....	37
Table 2: Performance of MPQS Algorithm on 19 Nodes Cluster.....	39
Table 3: Test Scenarios.....	40
Table 4: Storage Cluster Testing Analysis.....	42
Table 5: Diskless Booting.....	47
Table 6: High Availability Test Summary.....	51

# CHAPTER 1

## INTRODUCTION

### 1.1 Mission

This research work is based on the previous research conducted by [Mahboob, 2009] on “High Performance Linux Clusters for breaking RSA” in which a design of Linux based HS clusters is presented. In his research he identified the use of Linux based clustering solution which advances the performance and speed and also reduces the processing time for breaking RSA. He presented a recipe for building a Kerrighed cluster. This research work is an extension to [Mahboob, 2009] and eliminates the Single point of failure of boot server node by offering a solution of highly available cluster boot servers. The clustering of boot server which is the objective of this research is shown in Figure 1.

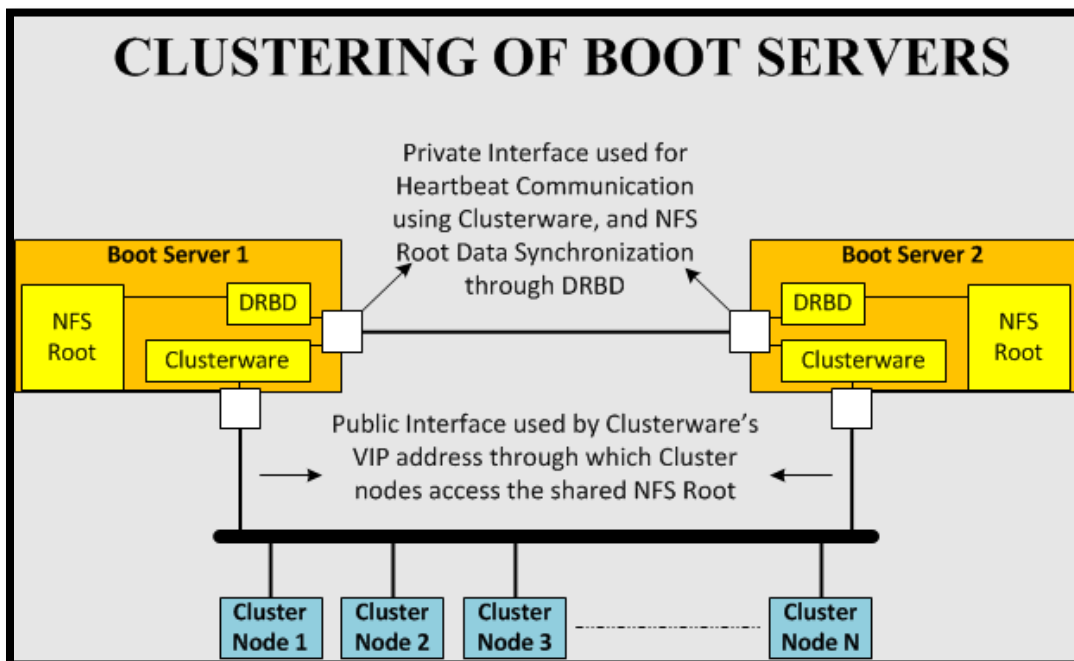


Figure 1: Clustering of Boot Server

## 1.2 Background Study

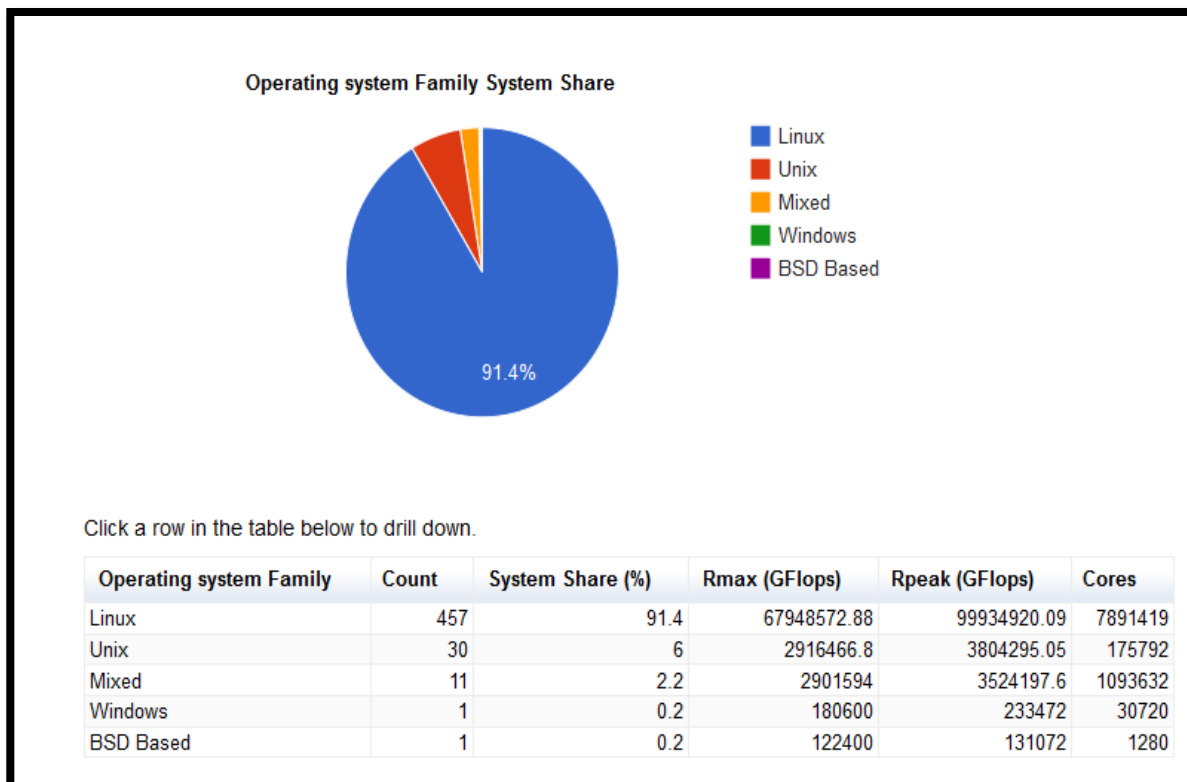
Computer clustering is getting fame for last few decades as it fulfills the desire to get more horsepower and reliability over a single computer by orchestrating a number of low cost computers [Mahboob, 2009]. Two main benefits of using Clusters technology are the scalability and availability. The demand of high computational and high performance computer clusters is increasing day by day in scientific and business domains, where there is a requirement of concurrent processing and computational load handling. Similarly high availability is a prime issue to these domains as the down-time due to maintenance or failover cannot be tolerated at any cost because it leads to loss of revenue. Availability can be achieved by using fail-over technique i.e. In case of failure of any node; the healthy nodes continue to work normally. Another concept is of “Hot-plugging” which is used in clustering. Hot-plug nodes can leave or join the cluster during normal execution of processes.

Linux Operating system has an extraordinary contribution in providing open source kernels to run many open source software on variety of hardware platforms. [Mahboob, 2009] Many clustering solutions have been developed exclusively for Linux. Linux computers on a network can appear as a cluster to cooperate with each other by means of remote shell execution and shared directories. Using handy codes with well-defined standards clues to progressively proficient use of clusters [Bader 2001]

Single System Image cluster offers a sight of one big machine with all-in-one integration of applications and resources. Kerrighed is one of the famous single system image projects which propose the vision of a single Symmetric Multiprocessing system on top of a cluster. [Kerrighed, 2004] Kerrighed allows a global process scheduler that can be dynamically configured and permits the shifting of assignments from overloaded nodes to under loaded nodes.

## 1.2 Linux Operating System

Oracle Enterprise Linux version 5.7 was selected for this research work as Linux is one of the most commonly used operating system as well as most widely supported platform for cluster software. One of the reasons of such a big fame of Linux is its open source feature which gives an opportunity to researchers and giant organizations to route open source software on broad variety of hardware platforms.



**Figure 2: Operating System Family Share**

**Source: [Top500, 2011]**

Figure 2 shows Operating System Family share statistics which declares Linux share of 91.4% out of all other operating systems in range. [Top500, 2011]

## **1.4 Clusters and types of Clusters**

A Cluster is an assortment of stand-alone computer nodes which are grouped together to solve any specific problem. [Cluster, 2012] There are different kinds of Clusters available out of which we will debate following four most popular types:

1. High Performance Computing Cluster
2. Load Balancing Cluster
3. High Availability Cluster
4. Storage Cluster

### **1.4.1 High Performance Compute Cluster (Beowulf)**

High Performance computing cluster as the name suggests solves the computing problems. Applications like data mining, parallel processing, simulations and computer graphical rendering falls under the category of this type of cluster's scope of work. The name Beowulf originally referred to a specific computer assembled in 1994 by Thomas Sterling and Donald Becker at NASA with 16 sets of DX4 PCs and 10 Mb/s Ethernet. Compute clusters are built on the principle of Message passing interface, In MPI scheme every cluster node runs its distinct operating system and have its own physical memory available.

### **1.4.2 Load Balancing Clusters**

In Load balancing clusters the forward-facing nodes of cluster accept the requests from end users and then forward the requests to the back office server nodes. The requests are actually circulated among multiple nodes with any defined strategy to balance the load. The strategy is maintained by a network administrator and it is based on the certain requirement. An example of such strategy can be: a certain amount of load will be borne by each node and when it exceeds with allowed load, it can be dynamically adjusted or balanced with other nodes.

### 1.4.3 High Availability Clusters

High availability clusters are very famous because they avoid downtime of services due to breakdown of any node software or hardware. The single point of failure is avoided by provision of High Available cluster nodes with redundancy as shown in Figure 3

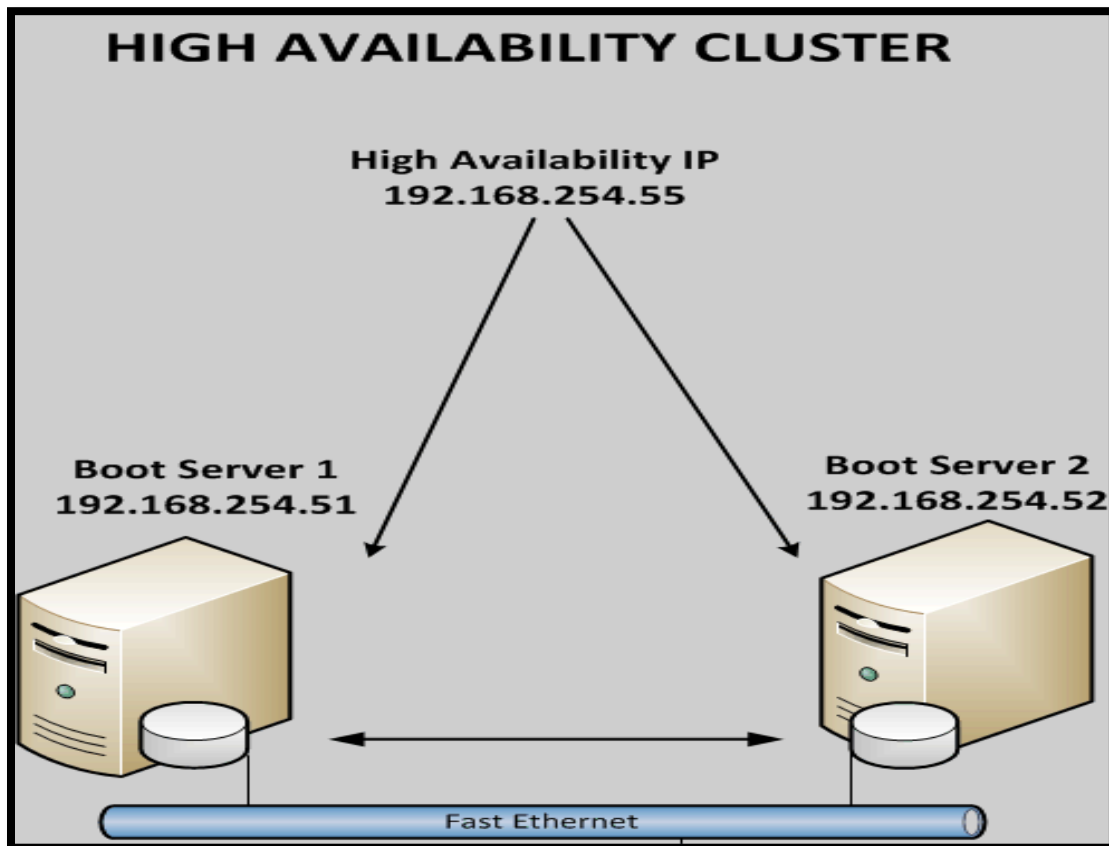


Figure 3: High Availability Cluster

### 1.4.4 Storage Clusters

Storage Clusters use Redundant Array of Independent disks (RAID) technology. The key objective of a storage cluster is accessibility by means of redundancy to end users. The next level technology is storage area networks (SANs). In this research work, the storage cluster used is Distributed Replicated Block device (DRDB).

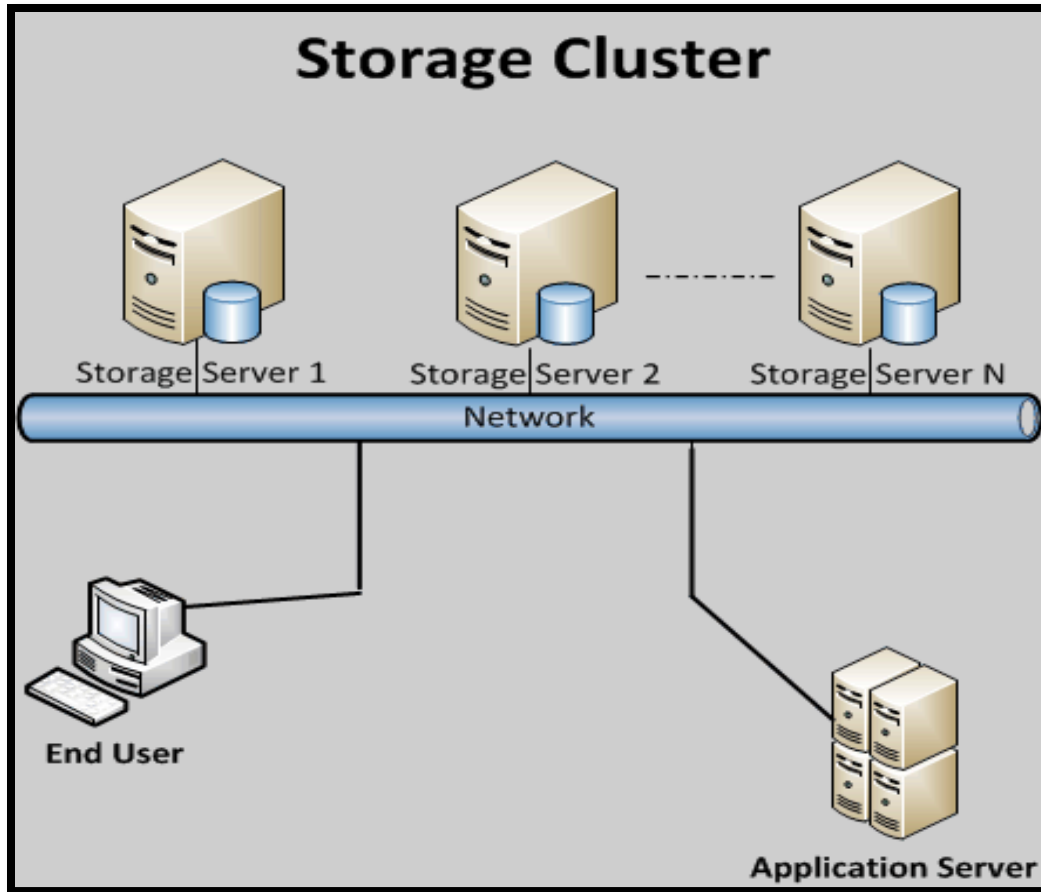


Figure 4: Storage Cluster

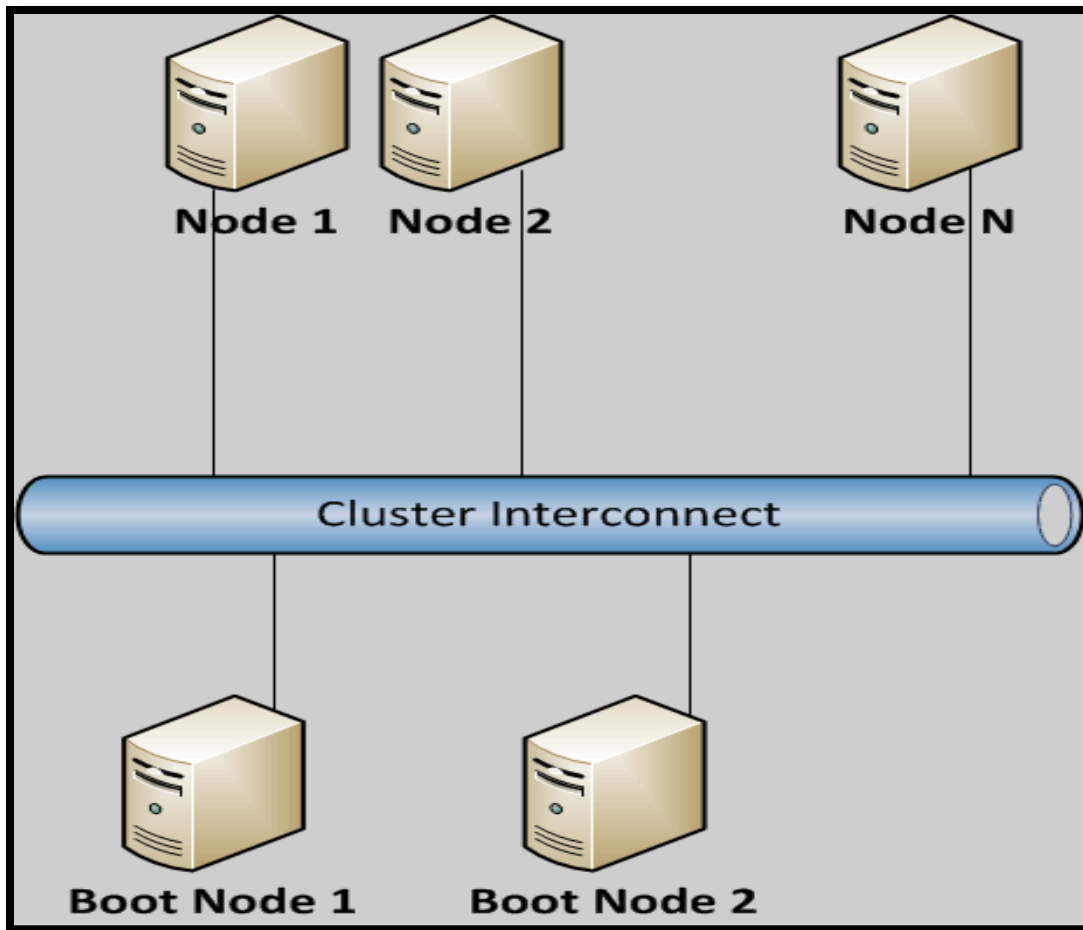
## 1.5 The Cluster Building Blocks

Following are the main building blocks of a cluster solution.

1. Cluster Boot Nodes
2. Cluster Member Nodes
3. Cluster Interconnect
4. Cluster Operating System software,
5. Middleware Software

Figure 5 shows the basic sight of the cluster nodes, cluster interconnect and the cluster boot server.





**Figure 5: Generic view of Cluster Components**

### **1.5.1 Cluster Nodes**

The Cluster node is made up of following components:

1. Hardware Components of a Cluster Node.
2. Software Components of a Cluster Node

The hardware components consist of the network interface card, the system bus, CPU and memory. The software components involves Operating system, middleware software e.g. Kerrighed and system libraries.

### **1.5.2 Cluster Interconnect**

The cluster Interconnect network offers the inter-node communication between nodes. Speed and latency of the interconnection network are the main limitations of the

clustering technology. High rate interconnect network can advance the overall performance of cluster solution.

This thesis has been organized as follows: **Chapter 1** gives an overview of Linux operating system, Clusters and their numerous types, the main constituents of a cluster based solution including particulars of cluster nodes and cluster interconnect. **Chapter 2** presents an architectural view of a booting solution and the essential commands and scripts for establishment of such boot setup. **Chapter 3** involves the technical details of DRBD, OCFS2 and steps for configuring Network file system are enlightened with relevant code. **Chapter 4** sketches our approach to prepare a high available cluster of boot nodes. In this chapter we have debated the Clusterware concept along with oracle based Clusterware solution and its benefits, pre-requisites for installation of Oracle Clusterware and step by step guide for configuration of Oracle Clusterware 11gR2. **Chapter 5** tells about single system image type of cluster.it includes history and various flavors of SSI cluster including Mosix, OpenMosix, OpenSSI and Kerrighed. Steps for compiling Kerrighed kernel and starting kerrighed service on cluster nodes have also been discussed in this chapter. **Chapter 6** gives the details of our working on Multiple Polynomial Quadratic Sieve [MPQS] algorithm for integer factorization. We have factored a 100 decimal digit integer from RSA Factoring Challenge by using MPQS algorithm running on our 20 node Kerrighed SSI cluster within seven hour time. **Chapter 7** presents the different test scenarios and the results, we have tested High availability of boot server nodes, the real time data synchronization on boot server nodes, diskless booting of cluster nodes over network and running of Kerrighed scheme on our cluster nodes. In this chapter we have tested an application "msieve" running on the cluster and observed the behaviour of Kerrighed with respect to stability and performance. We have performed a failover test while the "msieve" factorization was in progress and have shown that Kerrighed cluster does not hang to boot server failure.

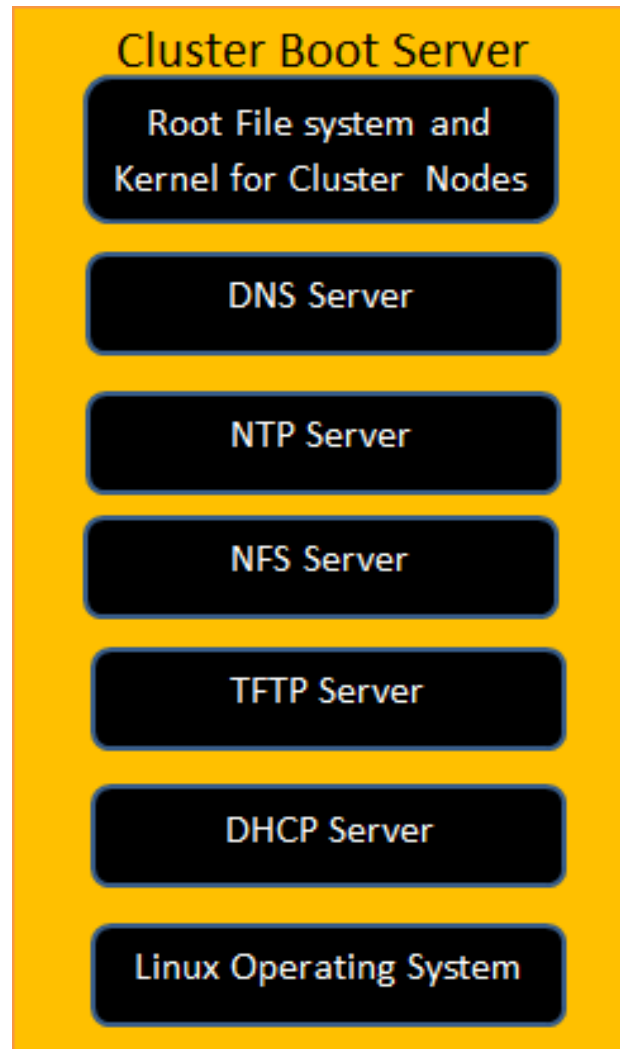
# Chapter 2

## BOOTING INFRASTRUCTURE

This chapter covers the technical knowledge of Booting infrastructure and provides the basic details of a Cluster boot server. The main building blocks of a boot server are discussed in addition to the main ingredients which we have configured in this research to prepare boot servers. What's the need of having a boot node in any cluster solution is primarily debated and the solution of redundant boot server nodes is presented to eliminate single point of failure. PXE boot technology is explained in this chapter covering the comparison of traditional booting procedures with PXE boot. We have discussed a recipe to setup boot infrastructure to prepare cluster nodes to boot over network card.

### 2.1 The Cluster Boot Server

Cluster boot server is a solution that allows the cluster nodes to load the required software easily. This solution offers plug and play feature to add cluster nodes. It is very important to know that boot server is not the part of single server image cluster and its role is just to facilitate the cluster nodes in accessing and loading their software and necessary configuration. In this research work we are using Oracle Enterprise Linux 5.7 on boot servers. The cluster boot server provides the infrastructure software to all cluster nodes and ensures that identical operating system and libraries are made available to all cluster nodes. We can call boot server as a cluster manager since it manages configuration parameters. To cope up the single point of failure issue of boot server we are using two redundant boot server nodes in this thesis. Figure 6 shows the most significant components of the cluster boot server which we will configure in this research.

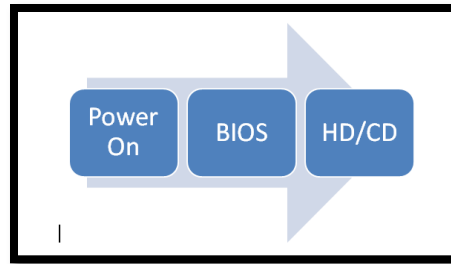


**Figure 6: Cluster Boot Server Components**

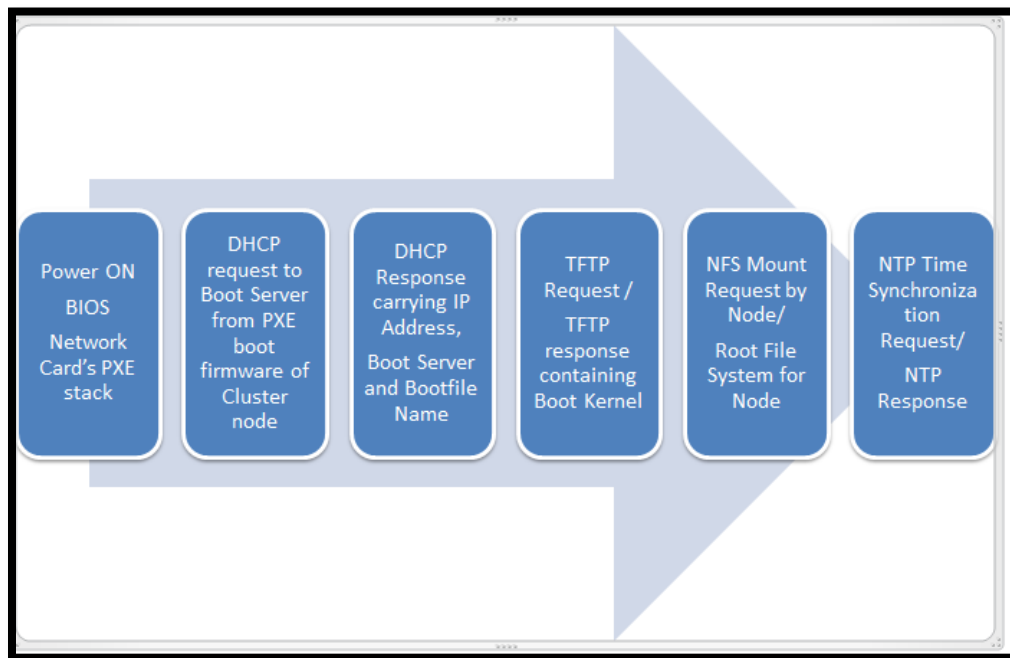
Adopted from [Mahboob, 2009]

## 2.2 Booting Procedure

PXE (Pre-boot eXecution Environment) is a technology that enables the cluster nodes to boot over the network and makes them fault free and plug & play. This method of booting was created way back in 1999 and makes it possible to circumvent the normal boot procedure. The normal boot procedure is shown in Figure: 7, The PXE booting scheme is shown in Figure: 8.



**Figure 7: Normal Booting Procedure**



**Figure 8: PXE Booting Procedure**

**Adoption: [Mahboob, 2009]**

### 2.2.1 DHCP

Dynamic Host Configuration Protocol (DHCP) systematizes the required transfer of network parameters to network devices. DHCP is an important protocol because it makes it so comfortable to add new machines to any network. In order to connect to a network; a query message is send to a DHCP server by DHCP client. The DHCP server maintains a list of different IP addresses and other essential information about clients e.g. the domain name and the default gateway. When a valid request is launched, the server assigns the node an IP address and other configuration parameters like default gateway and subnet

mask. The query is naturally initiated straightaway after booting, and essentially complete before the client can start IP-based communication with other hosts. The DHCP server can have any of the three procedures for allocation of IP addresses:

➤ **Dynamic allocation:**

- ✓ Assignment of range of IP addresses to DHCP by any network administrator.
- ✓ During network initialization request of client for an IP address from the DHCP server.
- ✓ Allocation of IP address to client.

➤ **Automatic allocation:**

- ✓ Assignment of free IP address to a client from the defined range of IP addresses.
- ✓ DHCP server keeps a track of past IP address which was allotted to nodes and it favorably assigns the same IP address to the client which it was having earlier.

➤ **Static allocation:**

- ✓ Manual assignment of IP addresses from a prepared list of IP addresses and MAC address.
- ✓ This list is manually prepared perhaps by a network administrator.
- ✓ Only those clients will be allotted an IP address whose MAC addresses are already listed in the table.
- ✓ It is also called the Static DHCP.

### **2.2.2 TFTP**

Trivial File Transfer Protocol (TFTP) is a file transfer protocol. As the name suggests it is very simple protocol. TFTP is typically used for transfer of boot files or configuration files between nodes. TFTP provides no authentication hence required very less amount of memory for implementation. Being an element of the Preboot Execution Environment, TFTP resides in the firmware of the host's network card. In TFTP, any new data transfer is started with a client's request to write or read a file, the connection is opened and the file is sent only if the request is granted by the server. An acknowledgment packet is required before the sending of subsequent packets and the sender just reserves last packet

in hand for retransmission if required. The three modes of transfer supported by TFTP are netascii, octet, and mail.

## 2.3 Setting up Booting Infrastructure

### 2.3.1 Installation of necessary packages

Following packages are required to install for setting up boot infrastructure:

```
yum install tftp-server xinetd dhcp syslinux
```

### 2.3.2 Amendment of Important Files

Edit the following files as shown in Appendix-A

1. /etc/dhcpd.conf
2. /etc/xinetd.d/tftp

Enable xinetd service on system startup

```
chkconfig xinetd on  
service xinetd start
```

Enabling DHCP server at system startup and start dhcp server:

```
chkconfig dhcpd on  
service dhcpd start
```

# CHAPTER 3

## DRBD, OCFS2 AND NFS SETUP

This Chapter covers the study of Distributed Replicated Block Device, Oracle Cluster File System 2 and Network File System. DRBD is studied and configured as one of the objective of this research work i.e. real time data synchronization on boot server nodes. The data replication task is successfully accomplished and we have ended up with not only high availability of boot server nodes but with the high availability of shared memory.

### 3.1 DRBD

The Distributed Replicated Block Device (DRBD) is a software-based, replicated storage solution which mirrors the content of block devices like hard disks, partitions and logical volumes between hosts. DRBD mirrors data in following forms:

1. **Mirroring of data in real time:** In real time mirroring replication occurs continuously while applications amend the data on the device.
2. **Mirroring of data transparently:** The applications are not required to be responsive that the data is being stored on multiple hosts.

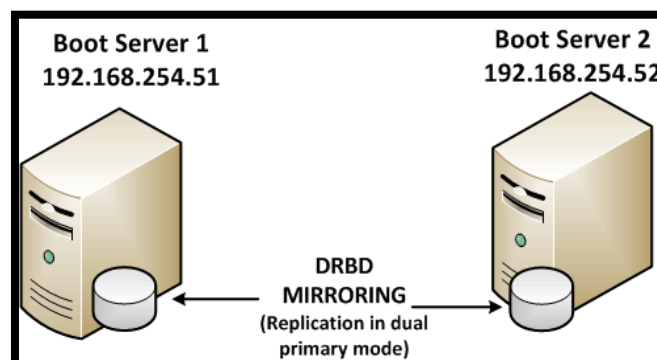


Figure 9: Distributed Replicated Block Device



## 3.1.1 DRBD Features

### Single-primary mode

In this scheme a resource remains in the primary role on only one cluster node at any instant of time, because only one cluster node is authorized to manipulate the data at any moment. Single-primary mode can be used with any conventional file system for example: ext3, ext4 and XFS. Setting up DRBD in single-primary mode is the acknowledged approach for high availability clusters.

### Dual-primary mode

In this scheme a resource remains in the primary role on both cluster nodes all the time. This makes the concurrent access to the data possible; this mode requires the use of a shared cluster file system that utilizes a distributed lock manager for example: GFS and OCFS2. Dual-primary mode of DRBD is noteworthy in Load-balancing clusters which require simultaneous data access from two nodes.

## 3.1.2 Building, Installing and Configuring DRBD

### 3.1.2.1: Laying the foundation – LVM over RAID1 setup

On our both server nodes (OL57-SRV1 & OL57-SRV2) the added hard drive is usually detected as /dev/sdb. Here we create 1 logical volume ‘**shared**’ that takes 100% of space on volume group **vg0**. [DRBD1, 2011] Specific commands for building physical volume and volume group are given in Appendix-B

### 3.1.2.2: Installation of DRBD and DRBD tools

Latest version of DRBD i.e. 8.4 will be installed along with DRBD tools using the commands given in Appendix-B [DRBD2, 2011].

### 3.1.2.3: Updating Configuration files

Configuration files which will be updated are given in following list:

1. /etc/drbd.d/global\_common.conf
2. /etc/drbd.conf
3. /etc/shared\_config

Boot Node 1:

```
drbdadm -- --overwrite-data-of-peer primary shared
```

Check DRBD sync status on both nodes as follows:

```
watch -n 1 cat /proc/drbd
```

After the sync is complete, execute the following command to make the second node primary as well:

Boot Node 2:

```
drbdadm primary shared
```

Enable DRBD on startup as follows:

```
chkconfig drbd on
```

## 3.2 OCFS2

### 3.2.1 Introduction

OCFS2 or Oracle Cluster File System 2 is an open source cluster file system which was acknowledged into Linux kernel 2.6.16. OCFS2 provides an enterprise-class substitute to patented cluster file systems and offers both high performance and high availability.

Since OCFS2 provides local file system semantics, so it can be used with any application. In this research work, the shared partition is formatted with OCFS2 to prepare it for high availability and data sharing.

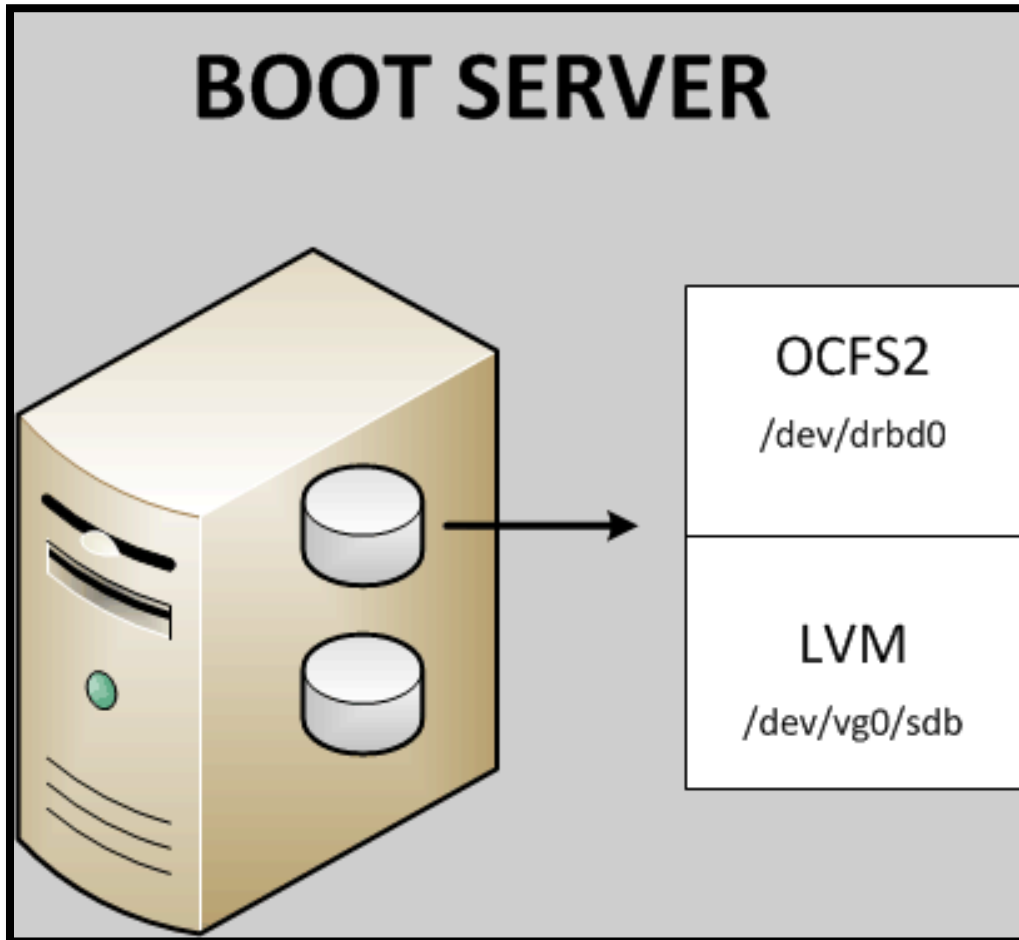


Figure 10: Oracle Cluster File System 2

### 3.2.2 Installation of OCFS2 & OCFS2 tools

```
# wget http://oss.oracle.com/projects/ocfs2/files/Redhat/RHEL5/x86_64/1.4.7-1/2.6.18-274.el5/ocfs2-2.6.18-274.el5-1.4.7-1.el5.x86_64.rpm
# rpm -Uvh ocfs2-2.6.18-274.el5-1.4.7-1.el5.x86_64.rpm
# yum install ocfs2
```

### 3.2.3 Updating of Configuration Files

1. /etc/sysconfig/o2cb [DRBD3, 2011]

2. /etc/ocfs2/cluster.conf

### 3.2.4 Configure Distributed lock manager (DLM)

Distributed lock manager comes with OCFS2. For this, give the following command on each node.

```
# /etc/init.d/o2cb configure
```

Check the status of o2cb service

```
# /etc/init.d/o2cb status
```

## 3.3 Configuring NFS

All preliminary steps are covered, now we are ready to format and mount cluster file system.

On any one of the two boot nodes, format the DRBD share as an OCFS2 partition:

```
# mkfs.ocfs2 -b 4K -T mail -N 2 -v /dev/drbd0
```

Create a directory in /mnt named nfs-root and mount the ocfs2 partition on both VMs:

```
# mkdir /mnt/nfs-root/  
# mount -t ocfs2 -o noatime /dev/drbd0 /var/lib/tftpboot/nfsroot/
```

Then edit /etc/exports and add the following line:

```
/tftpboot/nfsroot/ 192.168.0.0/24(rw,sync,no_root_squash,no_subtree_squash)
```

Next run these commands on both systems:

```
# chkconfig nfs on  
# service nfs start
```

# CHAPTER 4

## CLUSTERWARE

This chapter covers the formation of a high availability cluster of two boot nodes using oracle Clusterware. The detail of Clusterware and its benefits along with necessary technical knowledge of oracle Clusterware installation on a two nodes cluster has also been discussed. The necessary hardware and network requirements for boot server nodes to install oracle Clusterware software are also enlisted. We have covered all the pre-requisites for this installation in this chapter and step by step guide for installation and configuration is also given for better understanding of Oracle Clusterware 11gR2 in Appendix.

### 4.1 Clusterware

Clusterware is software which empowers servers to operate together as if they are one server. Every server appears like any standalone server. However, each server has its additional processes that communicate with each other so end users or applications can see them as one server.

### 4.2 Benefits of using Clusterware

Some benefits of using Clusterware solution are as under:

- Applications scalability
- Low cost hardware
- Failover support
- Ability to grow the existing capacity of servers.

Oracle Clusterware is programmed to sustain the availability of user applications. In an Oracle Real Application Clusters (Oracle RAC) environment, Oracle Clusterware manages all of the Oracle Database processes automatically. Anything that Oracle Clusterware manages is known as a cluster resource, it can be a database, an instance, a service, a listener, a virtual IP (VIP) address, an application process. Etc.

Oracle Clusterware offers the proficiency to:

- Eliminate downtime issues due to any of the hardware or software failure.
- Improve performance by allowing the applications to run on all of the nodes in the cluster which in its turn increase the throughput for applications.
- Add server nodes to the exiting cluster easily, when necessary, which increases the overall throughput for the application.
- Provide a scalable system with low cost commodity hardware to reduce the total cost of ownership for infrastructure.

### **4.3 Real Application Clusterware in our case**

In this research work Oracle Clusterware solution is set up on two boot nodes (OL5-SRV1 & OL5-SRV2). The general specification of these boot nodes is given as under:

#### **OL5-SRV1**

- Hostname: OL57-SRV1.localdomain
- Public IP Address eth0: 192.168.0.51
- Default Gateway eth0: 192.168.0.1
- Private IP Address eth1: 192.168.254.51

#### **OL5-SRV2**

- Hostname: OL57-SRV2.localdomain
- Public IP Address eth0: 192.168.0.52
- Default Gateway eth0: 192.168.0.1
- Private IP Address eth1: 192.168.254.52

## 4.4 Preparation for Installation of Clusterware

In order to have successful installation of Clusterware on boot server nodes it is equally important to prepare server nodes with mandatory hardware and software requirements.

### 4.4.1 Hardware Requirements for Boot Server Nodes

- Each Boot node must have minimum 2GB Ram available for installation of grid infrastructure of Clusterware.
- /tmp directory must have minimum 1GB free space on each Boot node.
- Each boot server nodes must have 2 network interface cards to configure public and private IP addresses.

### 4.4.2 Network Requirements for Boot Server Nodes

Each boot server node must have three IP addresses preconfigured to network interface cards.

1. Public IP address
  2. Private IP address
  3. Virtual IP address
- The Public network IP chosen in this research work is 192.168.0.51 and 192.168.0.52 for boot server 1 and boot server 2 respectively.
  - The Private network IP chosen in this research work is 192.168.254.51 and 192.168.254.52 for boot server 1 and boot server 2 respectively.
  - The Virtual IP chosen in this research work is 192.168.0.53 and 192.168.0.54 for boot server 1 and boot server 2 respectively.
  - The Single Client Access Name (SCAN) IP chosen in this research work is 192.168.0.55 for both server nodes. It is important to know that SCAN must assigned a unique name across all nodes and this name must be configured on same network as public and virtual IP's.

## 4.5 Steps for Installation and Configuration of Oracle Clusterware 11gR2

### 4.5.1 Installation of Necessary Packages

Before installing oracle Clusterware, we need to install some necessary packages on both server nodes i.e. OL57-SRV1 & OL57-SRV2 [Oracle11gR2, 2011]

Use the command below:

```
# yum -y install libaio-devel unixODBC unixODBC-devel sysstat libaio DHCP  
debootstrap
```

### 4.5.2 Updating of Files

After installation of basic packages, some files are required to be updated as a preliminary step for installation of Oracle Clusterware on both boot nodes. The script which is required to update these files is given in Appendix- D. The list of files which will be updated in this context is given below:

1. /etc/fstab
2. /etc/hosts
3. /etc/sysctl.conf
4. /etc/security/limits.conf
5. /etc/pam.d/login
6. /etc/selinux/config
7. /home/oracle/.bash\_profile
8. /etc/exports

### 4.5.3 Adding of Groups and Users

Now we are required to add group and users to our nodes, here we will add a group named “Oracle” and the password for this group is set to be “oracle” as given in Appendix- D



#### 4.5.4 Creation of Some Directories

Creation of some directories is the next step to installation setup for Clusterware as shown in Appendix- D

#### 4.5.5 Setting up NFS Share and its Export and Mounting

NFS share is needed to setup on both nodes; the specific commands for this work are given in Appendix- D

#### 4.5.6 Installation of Oracle Clusterware Latest Release

At this point, all preliminary configurations are done. Now we have to download the latest version of oracle Clusterware i.e. Oracle RAC 11g Release 2 (11.2.0.2). Copy the zip file into /home/oracle directory of Boot Node 1 and unzip it. Now start both Nodes as oracle user. Install the Grid Infrastructure on Boot Node 1. Start the Oracle installer.

```
./runInstaller
```

#### 4.5.7 Guide line for Oracle Installer

1. Select “Install and configure grid infrastructure for a cluster” option.
2. Input the Following details as required in next steps:

```
Cluster Name: OL57-SR-CLUSTER
SCAN Name: OL57-SCAN
SCAN Port: 1521
Host Name: OL57-SRV1
Virtual IP Name: OL57-VIP1.localdomain
Host Name: OL57-SRV2
Virtual IP Name: OL57-VIP2.localdomain
```

3. Select the Public and Private Interfaces. We have set eth0 as public Interface and eth1 as Private Interface in our case.
4. Give the location of OCR File and voting disk file as `01/shared_config/ocr_configuration`
5. Give the location of Voting disk file as `/u01/shared_config/voting_disk`

6. Give the software locations for storing oracle software files as `u01/app/11.2.0/grid`
7. Oracle Grid Infrastructure configuration is completed as shown in Figure.11

```

oracle@OL57-SRV1:/home/oracle/grid - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
CRS-2673: Attempting to stop 'ora.ctssd' on 'ol57-srv1'
CRS-2677: Stop of 'ora.ctssd' on 'ol57-srv1' succeeded
CRS-2673: Attempting to stop 'ora.cssdmonitor' on 'ol57-srv1'
CRS-2677: Stop of 'ora.cssdmonitor' on 'ol57-srv1' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'ol57-srv1'
CRS-2677: Stop of 'ora.cssd' on 'ol57-srv1' succeeded
CRS-2673: Attempting to stop 'ora.gppnd' on 'ol57-srv1'
CRS-2677: Stop of 'ora.gppnd' on 'ol57-srv1' succeeded
CRS-2673: Attempting to stop 'ora.gipcd' on 'ol57-srv1'
CRS-2677: Stop of 'ora.gipcd' on 'ol57-srv1' succeeded
CRS-2673: Attempting to stop 'ora.mdnsd' on 'ol57-srv1'
CRS-2677: Stop of 'ora.mdnsd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.mdnsd' on 'ol57-srv1'
CRS-2676: Start of 'ora.mdnsd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.gipcd' on 'ol57-srv1'
CRS-2676: Start of 'ora.gipcd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.gppnd' on 'ol57-srv1'
CRS-2676: Start of 'ora.gppnd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'ol57-srv1'
CRS-2676: Start of 'ora.cssdmonitor' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.cssd' on 'ol57-srv1'
CRS-2676: Start of 'ora.cssd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.diskmon' on 'ol57-srv1'
CRS-2676: Start of 'ora.diskmon' on 'ol57-srv1' succeeded
CRS-2676: Start of 'ora.cssd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.ctssd' on 'ol57-srv1'
CRS-2676: Start of 'ora.ctssd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'ol57-srv1'
CRS-2676: Start of 'ora.crsd' on 'ol57-srv1' succeeded
CRS-2672: Attempting to start 'ora.evmd' on 'ol57-srv1'
CRS-2676: Start of 'ora.evmd' on 'ol57-srv1' succeeded

ol57-srv1 2012/02/29 17:29:40 /u01/app/11.2.0/grid/cdata/ol57-srv1/backup_20120229_172940.olr
Preparing packages for installation...
cvuqdisk-1.0.7-1
Configure Oracle Grid Infrastructure for a Cluster ... succeeded
Updating inventory properties for clusterware
Starting Oracle Universal Installer...

Checking swap space: must be greater than 500 MB. Actual 1439 MB Passed
The inventory pointer is located at /etc/oraInst.loc
The inventory is located at /u01/app/oraInventory
'UpdateNodeList' was successful.

```

**Figure 11: Grid Infrastructure Configuration Success**

At this point we are successfully done with our Oracle RAC installation on both Boot Server nodes.

# CHAPTER 5

## SINGLE SYSTEM IMAGE CLUSTER

This chapter deals with Single System Image Cluster and its numerous flavors. We have argued the technical details of SSI cluster types including Mosix, OpenMosix, OpenSSI and Kerrighed. We have used Kerrighed v2.4.4 in our research work and have shown various features of Kerrighed. The overall guide for compiling Kerrighed kernel alongwith necessary commands is also given in Appendix-E.

### 5.1 Introduction

Single System Image (SSI) is a software mechanism which suggests the view of one distinctive machine on top of a cluster. The aim of the SSI software is to offer a scalable, available and convenient server cluster environment. SSI operating systems are very striking as they provide simple and user friendly techniques for cluster programming and use. Some implementations of SSI are Mosix, OpenMosix, Kerrighed and Open SSI.

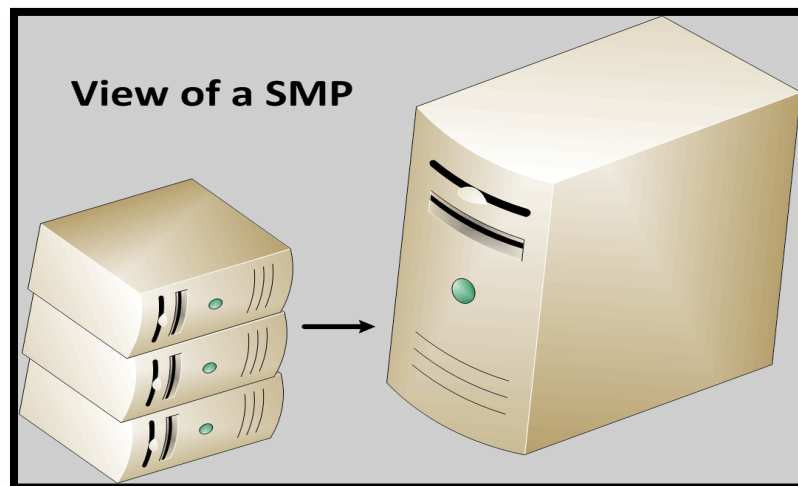


Figure 12: View of a SMP

## **5.2 Mosix**

MOSIX is a software suite that advances the Linux kernel with cluster capabilities. The improved kernel can provide support to any size cluster of X86/Pentium based boxes. MOSIX lets the clear migration of processes to other nodes in the cluster, while standard Linux process control utilities show all processes as if they are running on the node the process originated from. For example 'ps' utility.

## **5.3 OpenMosix**

OpenMosix was developed in 2001 by Linux MOSIX community after MOSIX became patented software. However OpenMosix supported only Linux Kernel version 2.4 and could not reached to the version 2.6. Ultimately the maintainer of OpenMosix had to announce the end of OpenMosix project with effect from 1<sup>st</sup> March, 2008, and claimed that "with the increase in power and availability features of low cost multi-core processors, single-system image (SSI) clustering is becoming the less of a factor in computing.". Now openMosix is just of historical interest.

## **5.4 OpenSSI**

The SSI project popped up with an aim to pool available Linux clustering related projects and to create a SSI cluster. Regrettably the project could not run. Originally the project had a strong obligation and support from Hewlett-Packard. After a few years of vigorous work only a few contributors remain devoted to the project. The last major release was made in 2006.

## **5.5 Kerrighed**

Kerrighed is a Single System Image operating system based on the Linux kernel. The Kerrighed project was launched at the French National Research Laboratory (INRIA) in 1998. The project may be regarded as mature as it is existed for over a decade. Kerrighed offers the view of a unique SMP machine on top of a cluster of standard PCs. The goals of Kerrighed are ease of use, extraordinary performance of applications, High availability of the cluster, effective resources management, and high customizability of the operating

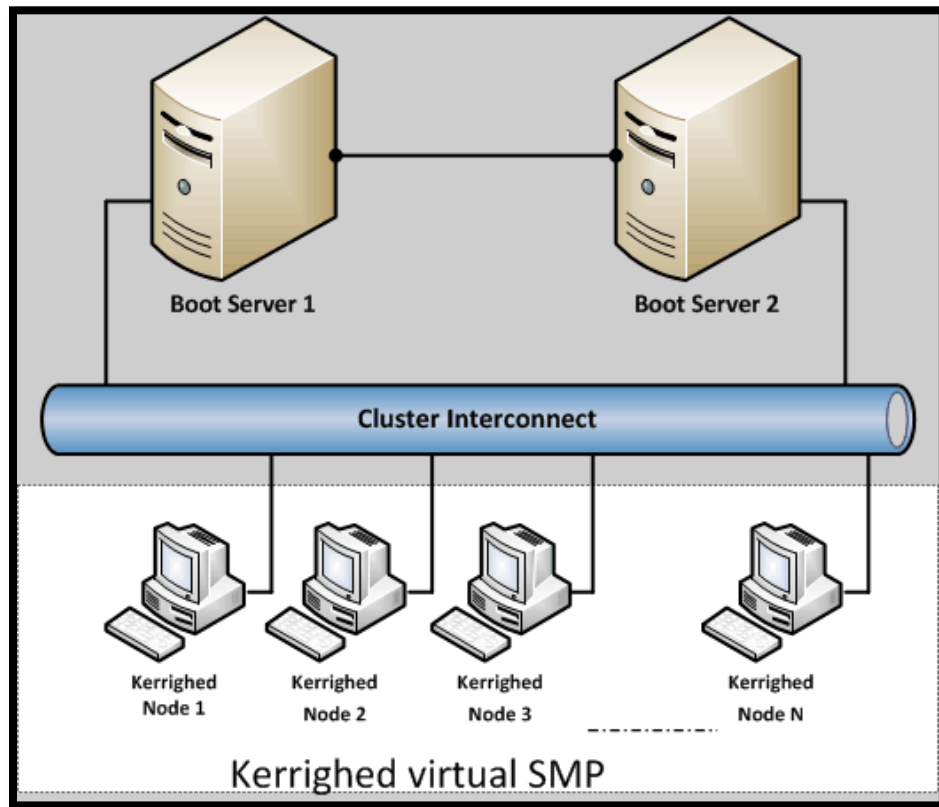
system. Kerrighed is implemented as an extension to the Linux operating system (a set of modules and a patch to the kernel).

## 5.6 Kerrighed Features

The main features supported by Kerrighed architecture are as under:

**Kerrighed Hotplug:** Kerrighed offers practical support of adding various nodes to a live cluster without affecting the running cluster. This experimental support enables us to add nodes to cluster with very simple commands

```
# krgadm nodes add
```



**Figure 13: Kerrighed**

**Migration Support:** Kerrighed supports migration of processes and tasks from overloaded nodes to under loaded nodes to perform leveling.

# CHAPTER 6

## BREAKING RSA USING MULTI QUADRATIC SEIVE ALGORITHM

This chapter presents results of our work on Multiple Polynomial Quadratic Sieve [MPQS] algorithm for integer factorization. We have factored a 100 decimal digit integer from RSA Factoring Challenge by using MPQS algorithm implementation running on our 19 node Kerrighed SSI cluster within seven hour time. We have performed a failover test while the "msieve" factorization is in progress and show that the Kerrighed cluster does not hang to boot server failure.

### 6.1 MULTIPLE POLYNOMIAL QUADRATIC SIEVE

In 1981 Carl Pomerance invented a factorization algorithm for integers known as quadratic sieve which is actually an improvement to Schroeppel's linear sieve algorithm. It is considered to be the fastest algorithm for integer's factorization under 100 decimal digits and second fastest for above 100 or above digit number. [MPQS, 2012] It is a simple algorithm and it runs solely on size of the integer that has to be factored.

### 6.2 Public-Key Cryptosystems

Some popular public-key algorithms utilize long number modular exponentiation are:

- RSA
- Diffie-Hellman Key Exchange
- Digital Signature Algorithm, DSA

Here we will only discuss RSA as other security algorithms are out of scope of this research.

## 6.3 RSA

RSA is a block cipher scheme. It is used for both encryption and authentication. It was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman [RRA78]. RSA scheme is renowned as the most widely accepted and implemented general-purpose approach to public-key encryption. RSA security relies on a hard problem known as IFP (Integer Factorization Problem). [MPQS, 2012].

### 6.3.1 Key Generation

RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated in the following way

1. Choose two large random prime numbers  $p$  and  $q$
2. Calculate  $n = p \times q$
3. Calculate  $\phi(n) = (p-1) \times (q-1)$
4. Choose  $d$  such that  $d$  is co-prime with  $\phi(n)$ , i.e.  $\gcd(d, \phi(n)) = 1$
5. Calculate  $e = d^{-1} \pmod{\phi(n)}$
6. Publish  $\{e, n\}$  as the public key and keep  $d$  secret as the private key.

### 6.3.2 Encryption

Let say a user A transmits her public key  $(n, e)$  to user B and keeps the private key secret. B then wishes to send message  $\mathbf{M}$  to A.

He first turns  $\mathbf{M}$  into an integer  $m$ , such that  $0 < m < n$  by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext  $c$  corresponding to

$$c = m^e \pmod{n}.$$

This can be done quickly using the method of exponentiation by squaring. B then transmits  $c$  to A. Note that at least nine values of  $m$  will yield a ciphertext  $c$  equal to  $m$ , But this is very unlikely to occur in practice.

### 6.3.3 Decryption

A can recover  $m$  from  $c$  by using her private key exponent  $d$  via computing

$$m = c^d \pmod{n}.$$

Given  $m$ , the original message  $\mathbf{M}$  can be recovered by reversing the padding scheme.

(In practice, there are more efficient methods of calculating  $c^d$  using the pre computed values below.)

### 6.3.4 RSA Security

The most damaging method to break RSA security is to discover the private key corresponding to a given public key. The obvious way to do this attack is to factor the public modulus  $n$ , into its two prime factors  $p$  and  $q$ . From  $p$ ,  $q$  and  $e$ , the public exponent, the attacker can easily get  $d$ , the private exponent. The hard part is factoring  $n$ ; the security of RSA depends on difficulty of factorizing  $n$ .

## 6.4 RSA Numbers

RSA numbers are difficult to-factor composite numbers having exactly two prime factors (i.e., so-called semiprimes) that were listed in the Factoring Challenge of RSA Security®--a challenge that is now withdrawn and no longer active.

Number	Digits	Factored (references)
RSA-100	100	Apr. 1991
RSA-110	110	Apr. 1992
RSA-120	120	Jun. 1993
RSA-129	129	Apr. 1994 (Leutwyler 1994, Cipra 1995)
RSA-130	130	Apr. 10, 1996
RSA-140	140	Feb. 2, 1999 (te Riele 1999a)
RSA-150	150	Apr. 6, 2004 (Aoki 2004)
RSA-155	155	Aug. 22, 1999 (te Riele 1999b, Peterson 1999)
RSA-160	160	Apr. 1, 2003 (Bahr et al. 2003)
RSA-200	200	May 9, 2005 (see Weisstein 2005a)
RSA-576	174	Dec. 3, 2003 (Franke 2003; see Weisstein 2003)
RSA-640	193	Nov. 4, 2005 (see Weisstein 2005b)
RSA-704	212	
RSA-768	232	Dec. 12, 2009 (Kleinjung 2010, Kleinjung et al. 2010)

**Table 1: RSA**

**Adaptation: [MPQS, 2012]**



## 6.5 Factorization Results

We have factored the following 100 decimal digit integer from Factoring Challenge of RSA Security® by using msieve algorithm implementation.

RSA-100 =                   15226050279225333605356183781326374297180681149613  
80688657908494580122963258952897654000350692006139

Prime factors generated with msieve using Quadratic Sieve algorithm are:

RSA-100 =                   37975227936943673922808872755445627854565536638199  
× 40094690950920881030683735292761468389214899724061

We have used the processing power of different CPU's (Dual core machine was used with a 2.0 GHz Pentium D processors and 512 megabytes of memory each). The number is break up in as many processes as required and allowed to run on a cluster. Each process is responsible to generate the Relations as defined by the user; in our case 5000 relations were assigned to each process. Once all process completes generating the relations, all relations are combined to form a single file. Next the process is run again, this time it will not calculate computationally intensive part to find the relations and only applies the post processing part to find the prime factor of that number.

The time required to factor with and without the cluster is given in Table 2 below.

Number of Nodes	Number of Cores	Time (hours)	Speedup (Approx)
1	1	64	1X
4	4	22	3X
8	8	12	5X
12	12	10.5	6X
16	16	8.5	7.5X
19	19	7	9X

**Table 2: Performance of MPQS Algorithm on 19 Nodes Cluster**

**Adaptation: [MPQS, 2012]**

# CHAPTER 7

## TEST SCENARIOS AND RESULTS

This chapter deals with different test cases we have adopted for checking the High Availability and stability of cluster based solution. These tests include tests for Real time data synchronization on Boot server nodes, High availability tests of boot servers and Booting tests of Kerrighed cluster nodes. We also have tested High Availability scheme on a Kerrighed cluster solving factorization problem using Quadratic Sieve algorithm. The research for this quadratic Sieve algorithm was done by a student of MS-EE-2008. We have adopted his work and tested the high availability solution which worked successfully and kept running in case of failure of a single boot node. During this seven hour test, the network card of one boot server node was plugged off and once the boot node was turned off however it caused no effect on running algorithm. The Basic Tests and their results are gathered in the following Table 3.

<b>Sr.No.</b>	<b>Test Name</b>	<b>Test Result</b>
<b>1</b>	High Availability of Boot Server Nodes	Successful
<b>2</b>	Booting of Nodes (Kerrighed Cluster Nodes)	Successful
<b>3</b>	Storage Cluster Testing (Data Replication on Boot Server Nodes)	Successful
<b>4</b>	High Availability Test on Kerrighed Cluster solving a factorization problem using Quadratic Sieve Algorithm	Successful

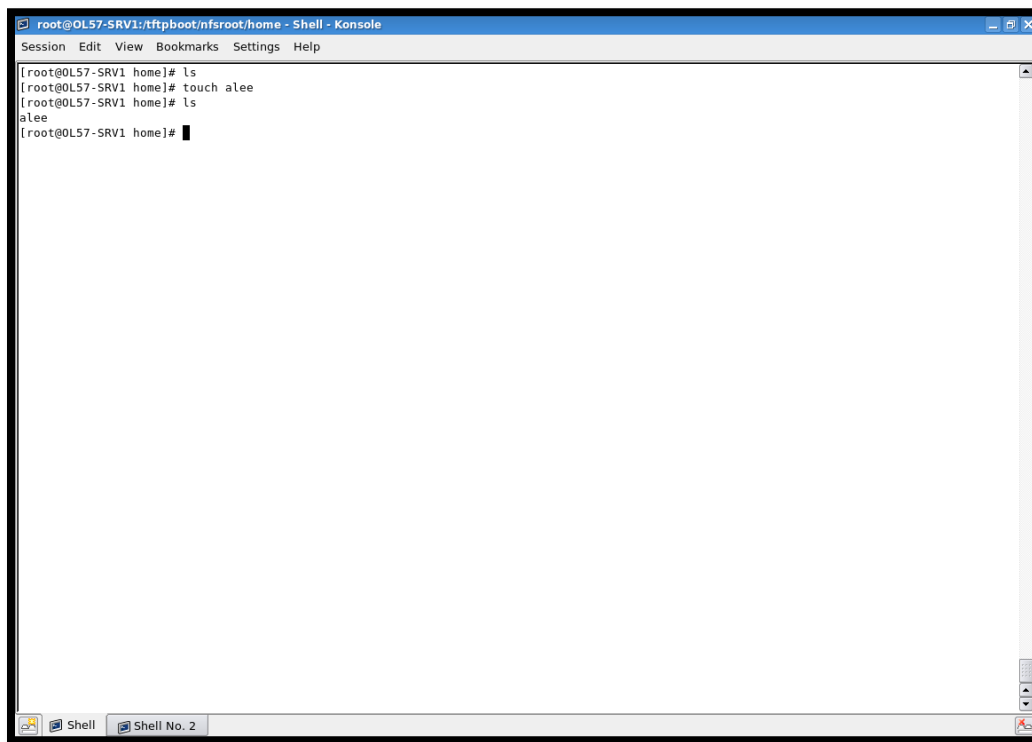
**Table 3: Test Scenarios**

## 7.1 Storage Cluster Testing

**Objective:** To Test if storage cluster is working and replicating or mirroring on Boot Server nodes.

**Test Procedure:** Create or Copy a file to the shared partition on Boot Server1 and check if it is automatically replicated to Boot server 2. We have created a file named “alee” in ‘/home’ on Boot Node 1 as shown in Figure-14 and it was automatically replicated to Boot Node 2 as shown in Figure-15

**Test Result:** Successful Data Replication on Boot Nodes



```
root@OL57-SRV1:/tftpboot/nfsroot/home - Shell - Konsole
Session Edit View Bookmarks Settings Help
[root@OL57-SRV1 home]# ls
[root@OL57-SRV1 home]# touch alee
[root@OL57-SRV1 home]# ls
alee
[root@OL57-SRV1 home]#
```

The image shows a terminal window titled "root@OL57-SRV1:/tftpboot/nfsroot/home - Shell - Konsole". The terminal output shows the following sequence of commands and results: a directory listing showing no files, the creation of a file named "alee" using the "touch" command, and a subsequent directory listing that shows the file "alee" has been created. The terminal window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". At the bottom, there are two tabs labeled "Shell" and "Shell No. 2".

**Figure 14: Memory Replication (1)**

```

root@OL57-SRV2:/tftpboot/nfsroot/home - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help
[root@OL57-SRV2 home]# ls
[root@OL57-SRV2 home]# ls
aLee
[root@OL57-SRV2 home]#

```

**Figure 15: Memory Replication (2)**

- A better understanding of Storage cluster test can be achieved with following table:

Sr.No.	Test Name	Size of File to be Replicated	Time Taken
1	Storage Cluster Test	10Kb	<1 Sec
2	Storage Cluster Test	100Kb	<1 Sec
3	Storage Cluster Test	1MB	<1 Sec

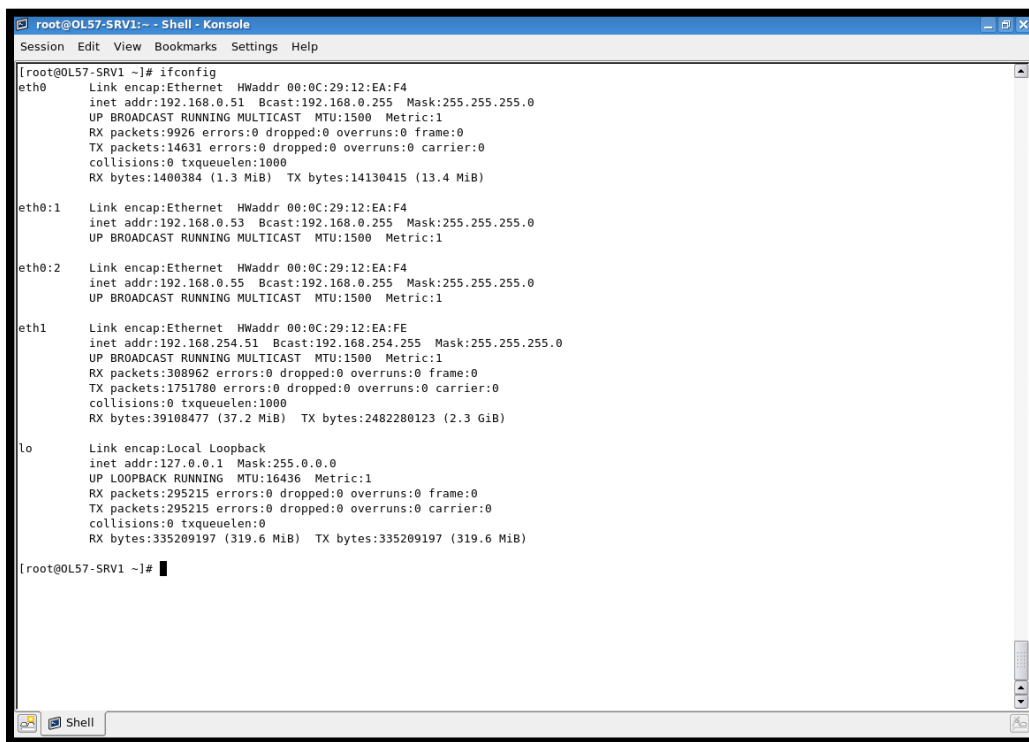
**Table 4: Storage Cluster Test Analysis**

## 7.2 High Availability Test on Boot Servers

### Test 7.2.1

**Objective:** To test if both Boot Servers are Highly Available on the network.

**Test Procedure:** Apply # ipconfig command to check available network interfaces on both Boot Nodes. SCAN IP 192.168.0.55 will be appearing on Boot Server 1: Now disable the network interface card on Boot Server-1 where the SCAN IP exists and check if the SCAN IP shifts on Server-2.



```

root@OL57-SRV1:~ - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@OL57-SRV1 ~]# ipconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:12:EA:F4
          inet addr:192.168.0.51  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:9926 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14631 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1400384 (1.3 MiB)  TX bytes:14130415 (13.4 MiB)

eth0:1    Link encap:Ethernet  HWaddr 00:0C:29:12:EA:F4
          inet addr:192.168.0.53  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

eth0:2    Link encap:Ethernet  HWaddr 00:0C:29:12:EA:F4
          inet addr:192.168.0.55  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

eth1      Link encap:Ethernet  HWaddr 00:0C:29:12:EA:FE
          inet addr:192.168.254.51  Bcast:192.168.254.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:308962 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1751780 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39108477 (37.2 MiB)  TX bytes:2482280123 (2.3 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:295215 errors:0 dropped:0 overruns:0 frame:0
          TX packets:295215 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:335209197 (319.6 MiB)  TX bytes:335209197 (319.6 MiB)

[root@OL57-SRV1 ~]#

```

**Figure 16: HA of Boot Servers (1)**

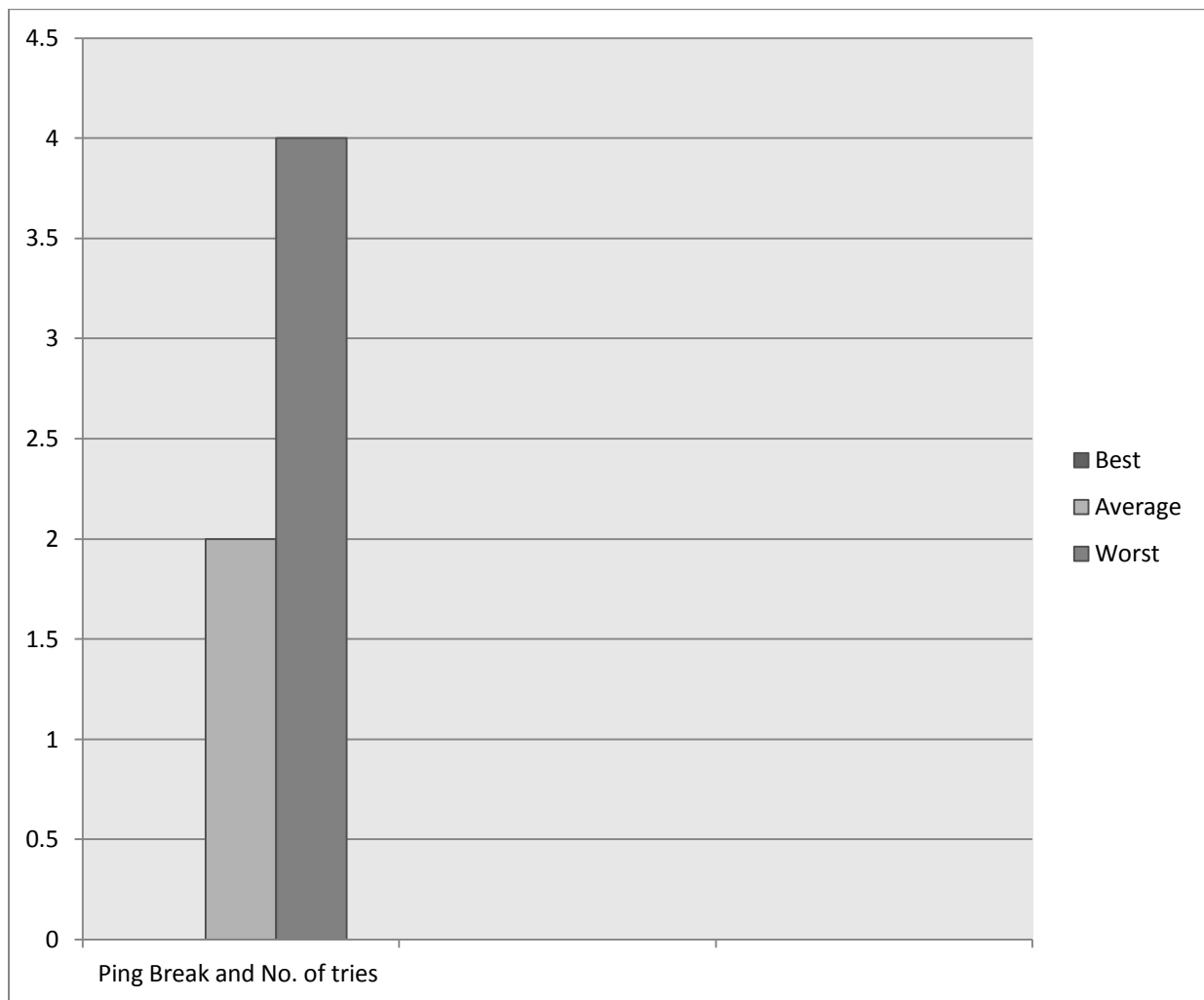
**Test Result:** Successful Shift of SCAN IP to Boot Server 2.

### Test 7.2.2

**Objective:** To test if both Boot Servers are Highly Available on the network

**Test Procedure:** Start a ping on SCAN IP from any one Kerrighed Node and then disable the network interface card of Boot Server 1, The ping will be discontinued for a

while and will automatically be restored since the Boot Server 2 takes over the charge as the Boot Server 1 fails, satisfying the High Availability Test of Boot Nodes. Graph 1 shows a statistical analysis of high availability case. Network SCAN IP was ping for a long period of time and boot nodes were set to off after few breaks and it was noted that Ping breaks after how much time. It was observed that the best case has no ping break, the other boot node automatically take over in less than a mill second of time. In average case ping breaks and tries two times. In worst case of our scenario the node has to try 4 times to find the boot node on network.



**Graph 1: Statistics of Ping Break & Try**

## 7.3 Diskless Boot System Testing

**Objective:** To test if Diskless boot system setup is working.

**Test Procedure:** Boot the cluster nodes over network and monitor the parameters like PXE, TFTP and DHCP activity, Messages related to Kernel loading, Messages related hardware detection and initialization and NFS mounting of Root file system.

**Test Result:** Booting of Cluster nodes through PXE boot over the network card and Successful mounting of Root File system as shown in Figure 17, 18 and 19



**Figure 17: Diskless Booting Test (1)**

```

CLIENT MAC ADDR: 00 0C 29 9E E3 3B  GUID: 564D9D88-9AB1-2A09-0206-68EEDB9EE33B
CLIENT IP: 192.168.0.242  MASK: 255.255.255.0  DHCP IP: 192.168.0.51
GATEWAY IP: 192.168.0.1

PXELINUX 3.10 2005-08-24  Copyright (C) 1994-2005 H. Peter Anvin
UNDI data segment at:  0009C7F0
UNDI data segment size: 24D0
UNDI code segment at:  0009ECC0
UNDI code segment size: 0A0D
PXE entry point found (we hope) at 9ECC:0106
My IP address seems to be C0A800F2 192.168.0.242
ip=192.168.0.242:192.168.0.55:192.168.0.1:255.255.255.0
TFTP prefix: /
Trying to load: pxelinux.cfg/01-00-0c-29-9e-e3-3b
Trying to load: pxelinux.cfg/C0A800F2
Trying to load: pxelinux.cfg/C0A800F
Trying to load: pxelinux.cfg/C0A800
Trying to load: pxelinux.cfg/C0A80
Trying to load: pxelinux.cfg/C0A8
Trying to load: pxelinux.cfg/C0A
Trying to load: pxelinux.cfg/C0
Trying to load: pxelinux.cfg/C
Trying to load: pxelinux.cfg/default
Loading boot/vmlinuz-2.6.20-krg...._

```

Figure 18: Diskless Booting Test (2)

```

DUFS initialisation done
KerIPC initialisation : start
KerIPC initialisation done
Proc initialisation: start
Proc initialisation: done
EPM initialisation: start
EPM initialisation: done
Init Kerrighed distributed services: done
scheduler initialization succeeded!
Kerrighed... loaded!
Try to enable bearer on eth0:<5>TIPC: Enabled bearer <eth:eth0>, discovery domain <1.1.0>, priority 10
ok
Try to enable bearer on lo:<5>TIPC: Enabled bearer <eth:lo>, discovery domain <1.1.0>, priority 10
ok
TIPC: Established link <1.1.243:eth0-1.1.240:eth0> on network plane A
TIPC: Established link <1.1.243:eth0-1.1.241:eth0> on network plane A
TIPC: Established link <1.1.243:eth0-1.1.242:eth0> on network plane A
.
Starting periodic command scheduler: crond.

Debian GNU/Linux 5.0 192.168.0.242 tty1

192.168.0.242 login: _

```

Figure 19: Diskless Booting Test (3)



Sr.No	Test Name	Parameters	Status
1	Diskless Booting Test	<ul style="list-style-type: none"> <li>✓ PXE Boot</li> <li>✓ TFTP and DHCP activity.</li> <li>✓ Messages related to Kernel loading,</li> <li>✓ Messages related to hardware detection and initialization.</li> <li>✓ NFS mounting of Root file system.</li> </ul>	Passed Passed Passed Passed Passed

**Table 5: Diskless Booting**

## 7.4 Kerrighed Testing

**Objective:** To Test if Kerrighed is successfully configured and working. Following are the tests which will be performed:

1. To check if the nodes are up
2. To check if the cluster is working
3. To list the process capabilities
4. Process migration check between cluster nodes

**Test Procedure:** Boot the Kerrighed Nodes over the network through PXE.

1. Check the status of Kerrighed nodes  
# krgadm nodes
2. Start the Kerrighed cluster  
# krgadm cluster start
3. Check the CPU usage of all nodes  
# top
4. To list the process capabilities  
# krgcapset -s
5. To allow migration of process take place between cluster nodes  
krgcapset -d +CAN\_MIGRATE

```

student@kerrighednode19: ~
File Edit View Terminal Help
top - 10:20:20 up 1:58, 2 users, load average: 19.87, 9.33, 4.28
Tasks: 336 total, 19 running, 317 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.2%us, 0.1%sy, 0.0%ni, 97.8%id, 0.0%wa, 0.2%hi, 0.7%si, 0.0%st
Mem: 9601496k total, 1359436k used, 8242060k free, 0k buffers
Swap: 0k total, 0k used, 0k free, 279460k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 929041169 student  25   0 47956 25m  236 R   102   0.3    2:05.87 msieve
 929041176 student  25   0 47844 20m  208 R   102   0.2    1:52.59 msieve
 929041168 student  25   0 47940 30m  512 R   101   0.3    2:14.35 msieve
 929041170 student  25   0 47800 29m  520 R   101   0.3    2:11.08 msieve
 929041171 student  25   0 47616 29m  552 R   101   0.3    2:27.32 msieve
 929041172 student  25   0 48004 29m  512 R   101   0.3    2:10.89 msieve
 929041173 student  25   0 48036 28m  512 R   101   0.3    2:03.84 msieve
 929041177 student  25   0 47852 28m  492 R   101   0.3    2:03.36 msieve
 929041178 student  25   0 47788 28m  512 R   101   0.3    2:05.27 msieve
 929041186 student  25   0 47612 25m  340 R   101   0.3    1:51.04 msieve
 929041175 student  25   0 47536 29m  512 R   100   0.3    2:22.72 msieve
 929041179 student  25   0 47488 28m  512 R   100   0.3    2:15.71 msieve
 929041182 student  25   0 47988 28m  552 R   100   0.3    2:04.21 msieve
 929041183 student  25   0 47540 29m  512 R   100   0.3    2:25.56 msieve
 929041184 student  25   0 47784 25m  340 R   100   0.3    1:58.60 msieve
 929041180 student  25   0 47736 28m  552 R   99   0.3    1:59.87 msieve
 929041167 student  25   0 48048 19m  208 R   98   0.2    1:38.62 msieve
1004538456 root      10  -5    0    0    0  S    0  0.0    1:45.35 krgcom/0
1004538788 student  16   0 19108 1360  864 R    4  0.0    0:00.04 top
 929041185 student  18   0 47848 13m  208 R    2  0.1    1:15.94 msieve
   1 root      18   0 4012  340  64  S    0  0.0    0:00.57 init
   2 root      RT    0    0    0    0  S    0  0.0    0:00.00 migration/0
   3 root      34  19    0    0    0  S    0  0.0    0:00.00 ksoftirqd/0
   4 root      RT    0    0    0    0  S    0  0.0    0:00.00 watchdog/0
   5 root      RT    0    0    0    0  S    0  0.0    0:00.00 migration/1
   6 root      34  19    0    0    0  S    0  0.0    0:00.00 ksoftirqd/1
   7 root      RT    0    0    0    0  S    0  0.0    0:00.00 watchdog/1
   8 root      10  -5    0    0    0  S    0  0.0    0:00.01 events/0
   9 root      10  -5    0    0    0  S    0  0.0    0:00.00 events/1
  10 root      10  -5    0    0    0  S    0  0.0    0:00.00 khelper
  11 root      10  -5    0    0    0  S    0  0.0    0:00.00 kthread
 101 root      10  -5    0    0    0  S    0  0.0    0:00.00 kblockd/0
 102 root      15  -5    0    0    0  S    0  0.0    0:00.00 kblockd/1

```

**Figure 20: Multiple msieve processes performing sieving on the Kerrighed cluster**

## Summary of High Availability Tests

<b>S/No.</b>	<b>Test Name</b>	<b>Test Procedure</b>	<b>Time Taken in Best Case</b>	<b>Test Result</b>
<b>1</b>	<b>High Availability of Storage Cluster</b>	Copy a file to the shared partition on Boot Server1 and check if it is automatically replicated to Boot server2	< 1sec Time in Best case of Replication	Successful Working of Storage Cluster
<b>2</b>	<b>High Performance Compute Cluster of Kerrighed</b>	Boot the Kerrighed nodes over the network through PXE and check if kerrighed is properly working, Kerrighed cluster and nodes are up and is migration of process takes place	=7 hrs to solve factorization problem	Successful working of High Performance Compute Cluster
<b>3</b>	<b>High Availability Cluster of two Boot Server Nodes</b>	Start a ping on SCAN IP from any Kerrighed Node and then disable/enable the network interface card of Boot Server 1 multiple times to check the behavior of boot nodes. While factorization is in process, switch off Boot Server 1 and check if it causes a hang to process failure.	<1 sec time to restore of ping. No ping failures in Best case of test observation	Successful High Availability of Two boot server nodes cluster

**Table 6: High Availability Test Summary**

# RESULTS

We have been able to achieve prime objective of this research, the removal of single point of failure of any cluster based solution carrying only one boot server. We have successfully setup three different types of clusters. (1) A Storage Cluster based on DRBD, OCFS2 and NFS (2) A High Availability Cluster of two Boot Servers using Oracle Clusterware and (3) High Performance Computing Cluster of Kerrighed based on SSI. We have performed several test scenarios on our clusters and have recorded the results. We have also given the basic procedures for preparing the given three types of clusters and have added the necessary scripts and commands in Appendices.

Our results show that the boot server and its services are now highly available to outside world. Any High performance computing cluster of kerrighed based on SSI can continue its processes working while the main boot server node fails because the 2<sup>nd</sup> boot server automatically takes over the charge and allows the processes to work without any hang over the network. We have setup Oracle Clusterware software 11gR2 for two nodes boot servers which makes them highly available to kerrighed cluster based on their SCAN IP address. A storage cluster is configured for data synchronization and replication between the boot servers for which we have used the Distributed Replicated Block Device (DRBD). DRBD is configured on both boot server nodes. The replicated device is formatted with Oracle Cluster File System 2. We have compiled the Kerrighed Kernel version 2.4.4 which creates a Single Server Image compute cluster based on configured policy. We have performed the required test scenarios to check the High Availability of Clusterware, Kerrighed performance, High Availability of shared disks and boot infrastructure of nodes through PXE boot.

# CONCLUSION

This thesis was motivated by the desire of learning how compute clusters work to solve different mathematical problems efficiently and how we can be able to remove the bottle neck i.e. the single point of failure of such cluster solution.

We designed three types of clusters: Storage Cluster based on DRBD, OCFS2 and NFS, High Availability Cluster of Boot Servers using Oracle Clusterware, High Performance Computing Cluster of Kerrighed based on SSI and concluded after performing tests on clustered environment, that high availability clusters of two boot nodes outperform the single point of failure. We also learned and presented the basic requirements to build aforementioned types of cluster. Real time replication of data on boot server nodes is also achieved and tested in this research work. In addition to it we have tested a Kerrighed cluster solving factorization problem using Quadratic Sieve algorithm. To improve and maintain the performance of given solution we will recommend that Cluster nodes available in RND lab must be upgraded. We would strongly suggest that our proposed dual boot node solution must be used in PNEC, Karachi in future. We are now working on a research paper to be published internationally to make this research available to the people working on cluster based solutions.

# Glossary

## Abbreviations

DRBD	Distributed Replicated Block Device
DHCP	Dynamic Host Configuration Protocol
TFTP	Trivial File Transfer Protocol
DNS	Domain Name System
NTP	Network Time Protocol
PXE	Pre-execution Environment
OCFS 2	Oracle Cluster File System 2

# APPENDIX-A

Appendix-A covers the necessary scripts regarding DHCP and TFTP configuration files.

Some parameters require brief explanation are given as under:

1. **next-server:** In this parameter we are required to give the SCAN IP or the IP which is common for both boot server nodes i.e. 192.168.0.55.
2. **range:** This is the range of IP addresses which we have reserved for Kerrighed cluster nodes to get assigned. 192.168.0.100 to 192.168.0.254.

Update /etc/dhcpd.conf and add the lines below:

```
# DHCP Server Configuration file.
# see /usr/share/doc/dhcp*/dhcpd.conf.sample
authoritative;
ddns-update-style none;
allow booting;
allow bootp;
option option-128 code 128 = string;
option option-129 code 129 = text;
next-server 192.168.0.55;
filename "/pxelinux.0";
#failover peer "DHCP-Failover" {
#     primary;
#     address 192.168.0.51;
#     port 647;
#     peer address 192.168.0.52;
#     peer port 647;
#     max-response-delay 60;
#     max-unacked-updates 10;
```

```

# mclt 3600;
# split 128;
# load balance max seconds 3;
#}
subnet 192.168.0.0 netmask 255.255.255.0 {
    interface eth0;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.0.255;
    option routers 192.168.0.1;
    option domain-name-servers 192.168.0.1;
    max-lease-time 1800;
    range 192.168.0.100 192.168.0.254;
# pool {
#     failover peer "DHCP-Failover";
#     deny dynamic bootp clients;
#     max-lease-time 1800;
#     range 192.168.0.100 192.168.0.254;
# } host kerrighednode1 {
    fixed-address 192.168.0.101;
    hardware ethernet 00:13:72:1D:42:37;
}}
subnet 192.168.254.0 netmask 255.255.255.0 {
    interface eth1;
}

```

Create a file /tftpboot/pxelinux.cfg/default as follows:

```
touch /tftpboot/pxelinux.cfg/default
```



# APPENDIX-B

Appendix-B deals with scripts for configuring DRBD and OCFS2 on our master boot server nodes. It gives the commands for creation of physical and logical volume groups, Steps for installation of DRBD and its basic kernel modules, DRBD tools installation, setting up DRBD in dual primary mode and its configuration files

Following steps have been followed for compilation of Storage cluster based on DRBD, OCFS2 and NFS:

## 1. Creating physical volume

```
pvccreate /dev/sdb
```

## 2. Creating volume group **vg0** on this physical volume

```
vgcreate vg0 /dev/sdb
```

## 3. Creating logical volume **shared** on volume group **vg0**

```
lvcreate -l100% VG -nshared vg0
```

## 4. Install DRBD tools and kernel modules

```
cd /root
```

```
wget http://oss.linbit.com/drbd/8.4/drbd-8.4.1.tar.gz
```

```
tar zxvf drbd-8.4.1.tar.gz
```

```
cd drbd-8.4.1
```

```
./configure && make rpm && make km-rpm
```

```
cd /usr/src/redhat/RPMS/x86_64/
```

```
rpm -Uvh drbd-km-2.6.18_274.el5-8.4.1-1.x86_64.rpm drbd-utils-8.4.1-1.x86_64.rpm
```

## 5. Updating /etc/hosts file since both server nodes should be reachable to each other via network by hostnames.

```
127.0.0.1 localhost.localdomain localhost
```

```
192.168.254.51 OL57-SRV1
```

192.168.254.52	OL57-SRV2
----------------	-----------

6. Create a new file in /etc/drbd.d/ folder named shared.res and the following lines to it:

resource shared {
meta-disk internal;
device /dev/drbd0;
syncer { rate 1000M; }
net {
allow-two-primaries;
after-sb-0pri discard-zero-changes;
after-sb-1pri discard-secondary;
after-sb-2pri disconnect;
}
startup { become-primary-on both; }
on OL57-SRV1 {
disk /dev/vg0/shared;
address 192.168.254.51:7789;
}
on OL57-SRV2 {
disk /dev/vg0/shared;
address 192.168.254.52:7789;
}
}

7. Update the file /etc/drbd.d/global\_common.conf

global {
usage-count yes;
# minor-count dialog-refresh disable-ip-verification
}
common {
protocol C;
handlers {

<pre> pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-degr.sh; /usr/lib/drbd/notify- emergency-reboot.sh; echo b &gt; /proc/sysrq-trigger ; reboot -f"; </pre>
<pre> pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-sb.sh; /usr/lib/drbd/notify- emergency-reboot.sh; echo b &gt; /proc/sysrq-trigger ; reboot -f"; </pre>
<pre> local-io-error "/usr/lib/drbd/notify-io-error.sh; /usr/lib/drbd/notify-emergency- shutdown.sh; echo o &gt; /proc/sysrq-trigger ; halt -f"; </pre>
<pre> # fence-peer "/usr/lib/drbd/crm-fence-peer.sh"; </pre>
<pre> # split-brain "/usr/lib/drbd/notify-split-brain.sh root"; </pre>
<pre> # out-of-sync "/usr/lib/drbd/notify-out-of-sync.sh root"; </pre>
<pre> # before-resync-target "/usr/lib/drbd/snapshot-resync-target-lvm.sh -p 15 -- -c 16k"; </pre>
<pre> # after-resync-target /usr/lib/drbd/unsnapshot-resync-target-lvm.sh; </pre>
<pre> } </pre>
<pre> startup { </pre>
<pre> # wfc-timeout degr-wfc-timeout outdated-wfc-timeout wait-after-sb; </pre>
<pre> } </pre>
<pre> disk { </pre>
<pre> # on-io-error fencing use-bmbv no-disk-barrier no-disk-flushes </pre>
<pre> # no-disk-drain no-md-flushes max-bio-bvecs </pre>
<pre> } </pre>
<pre> net { </pre>
<pre> # snd-buf-size rcvbuf-size timeout connect-int ping-int ping-timeout max-buffers </pre>
<pre> # max-epoch-size ko-count allow-two-primaries cram-hmac-alg shared-secret </pre>
<pre> max-epoch-size 8000; </pre>
<pre> # after-sb-0pri </pre>
<pre> # after-sb-1pri </pre>
<pre> # after-sb-2pri </pre>
<pre> # data-integrity-alg no-tcp-cork </pre>
<pre> } </pre>
<pre> syncer { </pre>
<pre> # rate after al-extents use-rlc cpu-mask verify-alg csums-alg </pre>
<pre> rate 128M; </pre>
<pre> } </pre>
<pre> } </pre>

8. Append the following lines to /etc/drbd.conf file

```
#include "drbd.d/global_common.conf";  
#include "drbd.d/shared.res";
```

9. In /etc/sysconfig/o2cb file:

change O2CB\_ENABLED=false to O2CB\_ENABLED=true

10. Run the following command on Boot Node 1:

```
drbdadm create-md shared
```

11. Run the following command on Boot Node 2:

```
drbdadm create-md shared
```

```
drbdadm up shared
```

12. Append the following lines to /etc/ocfs2/cluster.conf file:

```
node:  
ip_port=7700  
ip_address = 192.168.254.51  
number=0  
name=OL57-SRV1  
cluster=ocfs2  
node:  
ip_port=7700  
ip_address = 192.168.254.52  
number=1  
name=OL57-SRV2  
cluster=ocfs2  
cluster:  
node_count = 2  
name=ocfs2
```

# APPENDIX-C

1. Add the following lines to the "/etc/sysctl.conf" file.

```
fs.aio-max-nr = 1048576
fs.file-max = 6815744
kernel.shmall = 2097152
kernel.shmmax = 536870912
kernel.shmmni = 4096
# semaphores: semmsl, semmns, semopm, semmni
kernel.sem = 250 32000 100 128
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default=262144
net.core.rmem_max=4194304
net.core.wmem_default=262144
net.core.wmem_max=1048586
```

2. Run the given command to change the current kernel parameters.

```
/sbin/sysctl -p
```

3. Following lines will be added to the "/etc/security/limits.conf" file.

```
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

4. Following lines will be added to "/etc/pam.d/login" file.

```
session required pam_limits.so
```

5. Disable secure linux by editing the "/etc/selinux/config" file as given below.

```
SELINUX=disabled
```

6. Start the Name Service Cache Daemon (nscd).

```
chkconfig --level 35 nscd on
```

```
service nscd start
```

7. Login as the "oracle" user and add the following lines at the end of the ".bash\_profile" file.

### **Node: OL57-SRV1**

```
# Oracle Settings
TMP=/tmp; export TMP
TMPDIR=$TMP; export TMPDIR
ORACLE_HOSTNAME=OL57-SRV1.localdomain; export ORACLE_HOSTNAME
ORACLE_UNQNAME=OL57-SRV; export ORACLE_UNQNAME
ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
ORACLE_HOME=$ORACLE_BASE/product/11.2.0/db_1; export ORACLE_HOME
ORACLE_SID=OL57-SRV1; export ORACLE_SID
ORACLE_TERM=xterm; export ORACLE_TERM
PATH=/usr/sbin:$PATH; export PATH
PATH=$ORACLE_HOME/bin:$PATH; export PATH
LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib; export
LD_LIBRARY_PATH
CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdb
ms/jlib; export CLASSPATH
if [ $USER = "oracle" ]; then
  if [ $SHELL = "/bin/ksh" ]; then
    ulimit -p 16384
    ulimit -n 65536
  else
    ulimit -u 16384 -n 65536
  fi
fi
```

### **Node: OL57-SRV2**

```
# Oracle Settings
TMP=/tmp; export TMP
TMPDIR=$TMP; export TMPDIR
ORACLE_HOSTNAME=OL57-SRV2.localdomain; export ORACLE_HOSTNAME
ORACLE_UNQNAME=OL57-SRV; export ORACLE_UNQNAME
ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
ORACLE_HOME=$ORACLE_BASE/product/11.2.0/db_1; export ORACLE_HOME
ORACLE_SID=OL57-SRV2; export ORACLE_SID
```

```

ORACLE_TERM=xterm; export ORACLE_TERM
PATH=/usr/sbin:$PATH; export PATH
PATH=$ORACLE_HOME/bin:$PATH; export PATH
LD_LIBRARY_PATH=$ORACLE_HOME/lib:/lib:/usr/lib; export
LD_LIBRARY_PATH
CLASSPATH=$ORACLE_HOME/JRE:$ORACLE_HOME/jlib:$ORACLE_HOME/rdb
ms/jlib; export CLASSPATH
if [ $USER = "oracle" ]; then
  if [ $SHELL = "/bin/ksh" ]; then
    ulimit -p 16384
    ulimit -n 65536
  else
    ulimit -u 16384 -n 65536
  fi fi

```

### **Create Shared Disks**

8. Set up NFS shares. Here we will do this on the OL57-SRV1 node.

```

mkdir /shared_config
mkdir /shared_grid
mkdir /shared_home
mkdir /shared_data

```

9. Add the following lines to "/etc/exports" file.

```

/shared_config *(rw,sync,no_wdelay,insecure_locks,no_root_squash)
/shared_grid *(rw,sync,no_wdelay,insecure_locks,no_root_squash)
/shared_home *(rw,sync,no_wdelay,insecure_locks,no_root_squash)
/shared_data *(rw,sync,no_wdelay,insecure_locks,no_root_squash)

```

10. Export the NFS shares with given commands.

```

chkconfig nfs on
service nfs restart

```

11. Add the following lines to the "/etc/fstab" file.

SERVER1:

localhost:/shared_config	/u01/shared_config	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		
localhost:/shared_grid	/u01/app/11.2.0/grid	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		
localhost:/shared_home	/u01/app/oracle/product/11.2.0/db_1	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0		
01:/shared_data	/u01/oradata	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		

SERVER2:

OL57-SRV1:/shared_config	/u01/shared_config	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		
OL57-SRV1:/shared_grid	/u01/app/11.2.0/grid	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		
OL57-SRV1:/shared_home	/u01/app/oracle/product/11.2.0/db_1	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0		
01:/shared_data	/u01/oradata	nfs
rw,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=32768,wsiz=32768,actimeo=0 0 0		



# APPENDIX-D

1. Create the group and users: for user 'oracle': password is setup as 'oracle'.

```
groupadd -g 1000 oinstall
groupadd -g 1200 dba
useradd -u 1100 -g oinstall -G dba oracle
passwd oracle
```

2. Set up NFS shares by creating shared disks. Here we will do this on the OL57-SRV1 node.

```
mkdir /shared_config
mkdir /shared_grid
mkdir /shared_home
mkdir /shared_data
```

3. On both Nodes create the directories in which the Oracle software will be installed.

```
mkdir -p /u01/app/11.2.0/grid
mkdir -p /u01/app/oracle/product/11.2.0/db_1
mkdir -p /u01/oradata
mkdir -p /u01/shared_config
chown -R oracle:oinstall /u01/app /u01/app/oracle /u01/oradata /u01/shared_config
chmod -R 775 /u01/app /u01/app/oracle /u01/oradata /u01/shared_config
```

4. Mount the NFS shares on both nodes using following commands.

```
mount /u01/shared_config
mount /u01/app/11.2.0/grid
mount /u01/app/oracle/product/11.2.0/db_1
mount /u01/oradata
```

# APPENDIX-E

## Compiling Kerrighed Kernel

Following are the steps for compiling Kerrighed Kernel:

1. Root File System Setup
2. Installation of necessary packages
3. Adding all cluster nodes and server to host file.
4. Downloading Kerrighed and kernel sources from web.
5. Configure, Build and install Kerrighed

Before compiling kerrighed kernel we need to setup a new root file system in /tftpboot/nfsroot as follows:

```
debootstrap lenny /tftpboot/nfsroot/
```

1. Change the current root of the file system to the bootable filesystem directory:

```
chroot /tftpboot/nfsroot/
```

```
change password:
```

```
passwd
```

2. Mount the /proc directory of the current machine:

```
mount -t proc none /proc
```

3. To avoid errors from the Perl when installing packages in the image, type this command:

```
export LC_ALL=C
```

4. Edit /etc/apt/sources.list in order to download the necessary packages:

```
deb http://ftp.debian.org/debian/ lenny main contrib non-free
```

```
deb-src http://ftp.debian.org/debian/ lenny main contrib non-free
```

```
deb http://security.debian.org/ lenny/updates main contrib non-free
```

```
deb http://ftp.debian.org/debian/ lenny-proposed-updates main contrib non-free
```

```
deb-src http://security.debian.org/ lenny/updates main contrib non-free
```

```
deb-src http://ftp.debian.org/debian/ lenny-proposed-updates main contrib non-free
```

5. Install following packages:

```
apt-get install dhcp3-common nfs-common nfsbooted openssh-server automake autoconf
libtool pkg-config gawk rsync bzip2 gcc libncurses5 libncurses5-dev wget lsb-release
xmlto patchutils xutils-dev build-essential sudo ntp
```

6. Edit /etc/fstab of the bootable filesystem to have entries for proc and NFS mounted root file system:

```
proc /proc proc defaults 0 0
/dev/nfs / nfs defaults 0 0
configfs /config configfs defaults 0 0
```

7. Edit /etc/hosts and add all cluster nodes and server to it.

```
127.0.0.1 localhost
192.168.0.101 kerrighednode1
192.168.0.102 kerrighednode2
192.168.0.103 kerrighednode3
192.168.0.104 kerrighednode4
```

8. Automount the NFS shared filesystem at startup:

```
ln -sf /etc/network/if-up.d/mountnfs /etc/rcs.d/S34mountnfs
```

9. Edit /etc/network/interfaces and disable the network manager from managing the nodes ethernet cards. Change the network configuration to be manual as shown below:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
```

10. Add an entry for the root user in the /etc/sudoers file. This will allow the root to perform administrative commands in the bootable file system.

```
root ALL=(ALL) ALL
```

11. Create or edit /etc/kerrighed\_nodes and add the lines below:

```
session=1 #Value can be 1 - 254
nbmin=4 #Number of nodes which load before kerrighed autostarts.
Edit /etc/default/kerrighed and set ENABLE=true.
```

12. Download Kerrighed and kernel sources:

```
13. wget http://gforge.inria.fr/frs/download.php/26017/kerrighed-2.4.4.tar.gz
14. wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.20.tar.bz2
```

15. Configure, build and install Kerrighed as follows:

```
./configure --sysconfdir=/etc
make kernel
make
make kernel-install
make install
```

16. Issue the following command so that dynamic libraries installed by Kerrighed are registered with the dynamic library loader within the cluster node root filesystem:

```
ldconfig
```

17. Copy /tftpboot/nfsroot/boot/vmlinux-2.6.30-krp to /tftpboot/boot/ folder:

```
cp /tftpboot/nfsroot/boot/vmlinux-2.6.30-krp /tftpboot/boot/vmlinuz
```

18. Edit /tftpboot/pxelinux.cfg/default and add the lines below:

```
DEFAULT kerrighed
LABEL kerrighed
kernel boot/vmlinuz
append root=/dev/nfs nfsroot=<SERVER IP>:/tftpboot/nfsroot session_id=1
autonodeid=1 ip=dhcp rw
```

# REFERENCES

[Mahboob, 2009] Mahboob.A , Zubairi.J and Ikram.N , Book Chapter “High Performance Linux Clusters For Breaking RSA” in Bantham USA, eBook on Applications of Modern High Performance Networks, eISBN: 978-1-60805-077-2, 2009.

[Kerrighed, 2004] C.Morin, R. Lottiaux, G. Valle, P. Gallard, D. Margery, J. Berthou, and I. Scherson.” Kerrighed and data parallelism: Cluster computing on single system image operating systems” Proc. of Cluster 2004. IEEE, September 2004.

[Top500, 2011] Top500 Supercomputing Sites, Operating Family Share over Time, Available at <http://i.top500.org/stats>

[Kerrighed, 2011]Kerrighed on NFSROOT, Available at [http://kerrighed.org/wiki/index.php/Kerrighed\\_on\\_NFSROOT](http://kerrighed.org/wiki/index.php/Kerrighed_on_NFSROOT)

[Cluster, 2012] Cluster definition Available at <http://searchexchange.techtarget.com/definition/cluster>

[Linux clustering] Morris Law, IT Coordinator, Science Faculty, Hong Kong Baptist University

[Bader 2001] D.A. Bader and R. Pennington, “Cluster Computing: Applications,” The International Journal of High Performance Computing, 15(2):181-185, May 2001.

[DRBD1, 2011] OCFS2 cluster FS on dual primary DRBD: part 1 – setup dual primary DRBD Available at <http://server-support.co/blog/sysadmin/centos5-ocfs2-cluster-fs-on-dual-primary-drbd-part1-lvm-on-raid1/>

[DRBD2, 2011] OCFS2 cluster FS on dual primary DRBD: part 2 – setup dual primary DRBD Available at <http://server-support.co/blog/sysadmin/centos5-ocfs2-cluster-fs-on-dual-primary-drbd-part-2-setup-dual-primary-drbd/>

[DRBD3, 2011] OCFS2 cluster FS on dual primary DRBD: part 3 – setup OCFS2 cluster file system Available at <http://server-support.co/blog/sysadmin/centos5-ocfs2-cluster-fs-on-dual-primary-drbd-part-3-setup-ocfs2/>

[Oracle11gR2, 2011] Available at <http://www.oraclebase.com/articles/11g/OracleDB11gR2RACInstallationOnLinuxUsingNFS.php>

[MPQS, 2012] Breaking RSA Using Multiple Polynomial Quadratic Sieve (MPQS) using high performance Clusters, Report by Sami Khan, PNEC -2012