

Achieving High Availability by Implementing Beowulf Cluster

A.F.A. Abidin, N.S.M. Usop

Abstract—A computer cluster is a group of tightly coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance and/or availability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or availability. This paper proposed the way to implement the Beowulf Cluster in order to achieve high performance as well as high availability.

Keywords—Beowulf Cluster, grid computing, GridMPI, MPICH.

I. INTRODUCTION

IN computers, clustering is the use of multiple computers, typically PCs or UNIX workstations, multiple storage devices, and redundant interconnections, to form what appears to users as a single highly available system. Cluster computing can be used for load balancing as well as for high availability. There are several cluster categories that are High-availability (HA) clusters [1], Load-balancing clusters and Grid computing [2].

High-availability clusters (also known as failover clusters) are implemented primarily for the purpose of improving the availability of services which the cluster provides. They operate by having redundant nodes, which are then used to provide service when system components fail. The most common size for an HA cluster is two nodes, which is the minimum requirement to provide redundancy. HA cluster implementations attempt to manage the redundancy inherent in a cluster to eliminate single points of failure.

Load-balancing clusters operate by distributing a workload evenly over multiple back end nodes. Typically the cluster will be configured with multiple redundant load balancing front ends. Grid computing or grid clusters are a technology closely related to cluster computing. The key differences (by definitions which distinguish the two at all) between grids and traditional clusters are that grids connect collections of computers which do not fully trust each other, or which are geographically dispersed. Grids are thus more like a computing utility than like a single computer. In addition, grids typically support more heterogeneous collections than are commonly supported in clusters. It is optimized for workloads which consist of many independent jobs or packets of work, which do not have to share data between the jobs during the computation process. Grid computing or grid

clusters are a technology closely related to cluster computing. The key differences (by definitions which distinguish the two at all) between grids and traditional clusters are that grids connect collections of computers which do not fully trust each other, or which are geographically dispersed. Grids are thus more like a computing utility than like a single computer. In addition, grids typically support more heterogeneous collections than are commonly supported in clusters. It is optimized for workloads which consist of many independent jobs or packets of work, which do not have to share data between the jobs during the computation process.

II. OVERVIEW

Cluster computing closely related to grid computing. Traditionally, grid computing only existed in the realms of high performance computing and technical compute farms. Increasing demand for compute the power and data sharing, combined with technological advances, particularly relating to network bandwidth increases, has extended the scope of the grid beyond its traditional bounds.

Grid computing provides an environment in which network resources are virtualized to enable a utility model of computing. Grid computing provides highly available services with a high degree of transparency to users. These services can be delivered through Quality of Service guarantees, auto-negotiated contracts, metered access, and so forth.

Grid computing can be divided into three logical levels of deployment which are Global grids, Enterprise Grids and Cluster Grids.

III. METHODOLOGY

A. Beowulf

Beowulf [3] is a HPC cluster based on commodity hardware. It uses ordinary networks (Ethernet) and LINUX and other open source programs. It started out in late 1993. The thought behind the project was to develop a cheap alternative to the big supercomputers that were used at the time. There are two types of Beowulf. The Class 1 is based on Ethernet, while Class 2 is more expensive and uses more of specialized gear.

B. GridMPI Technology

GridMPI [4] is a new open-source and free software that implement the standard Message Passing Interface library designed for Grid environment. It enables unmodified programs to run on cluster computers distributed across Grid environment. It establishes a synthesized cluster computer by binding multiple cluster computers distributed geographically.

Users are able to seamlessly deploy their application programs from a local system to the Grid environment for processing a very large data set, which is too large to run on a single system.

C. Network File System

NFS uses client/server architecture and consists of a client program and a server program. The server program makes file systems available for access by other machines via a process called exporting. NFS clients' access shared file systems by mounting them from an NFS server machine.

D. SKaMPI (Benchmarking Tool)

Benchmarking tool provide a method of comparing the performance of various subsystems across different chip/system architectures. Benchmarking is helpful in understanding how the cluster responds under varying conditions.

IV. IMPLEMENTATION OF CLUSTER

A. Installing and Starting FreeBSD 6.3

The clean installation of FreeBSD is performed and a few important steps must be implementing. We need to choose to install the x window as well as Gnome or KDE as the GUI interface in order to simplify the configuration. The terminal login needs to default as root. After that type "startx" to test the GUI and press enter. X window will launch. Press ctrl+alt+spacebar to get back to the command prompt. Type Echo "startkde" > ~/.xinitrc and press enter. Type "startx" again and this time the system will start the kde GUI.

B. MPICH Configurations

The MPI [5] that used in this research is MPICH as well as the LAM-MPI [6]. It's better to get the newest tarball of lam-mpi with lesser error code. Unzip the file by using command tar -zxvf lam-mpi_7.1.4.tar.gz in /root/Desktop and the folder of lam-mpi_7.1.4 will appear. At terminal, type cd /root/Desktop/lam-mpi_7.1.4 and follow by ./configure to configure the lam-mpi and hence type make after configuration done and finally make install to install the lam-mpi.

C. NFS Configurations

The nfs-server and client configuration are stated in /etc/rc.conf file hence we need to edit the /etc/exports to enable mount on directory. Edit the /etc/exports as shown as figure1.

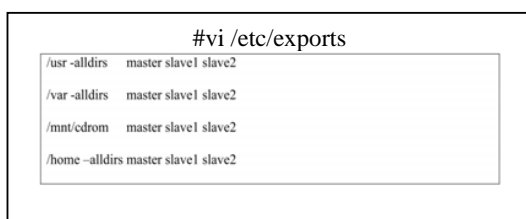


Fig. 1: Enable Mount on Directory

After editing the /etc/exports file on each node, we need to start the nfs-server before the mounting take part. Type this command at terminal in master nodes cd /etc/rc.d then preceed with. /nfs restart. This will the start the nfs-server and enable mount on each nfs-client to it.

D. SSH Configurations

Hence, we come to the configuration of SSH [8], secure shell. SSH is quite simple to configure if we know the way how it work. Many operating system come with ssh enable but for freebsd need to edit a simple command sshd_enable="YES" in /etc/rc.conf file.

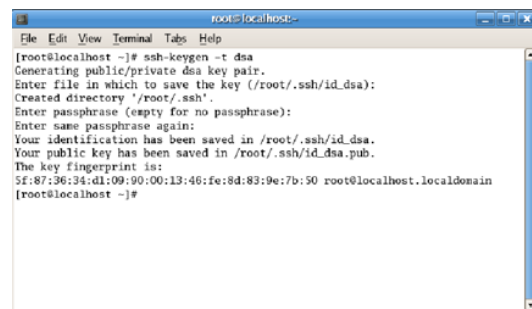


Fig. 2: Generate the key

Then, locate the directory where id_dsa and id_dsa.pub is stored. Type the following command in terminal cd /root/.ssh. Next, copy the id_dsa.pub file to new file and rename it as authorized_keys. This is done by typing the command cp id_dsa.pub authorized_keys in terminal.

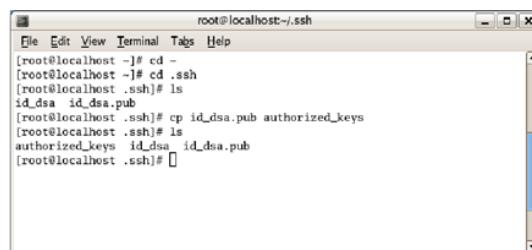


Fig. 3: Copy the key

E. Installing and Configuring Keychain

The execution command needs to be prompt everytime when the computers restart. Hence, using a keychain program to automatically execute the command everytime when login into the terminal. Download the keychain tarball from the <http://dev.gentoo.org> website. Next step, extracting the keychain tarball, keychain-2.4.3.tar.bz2 in /root/Desktop directory and a file name keychain-2.4.3 will appear after the extraction complete. After that, type tar -zxvf keychain2.4.3.tar.bz2 as shown as below to extract.

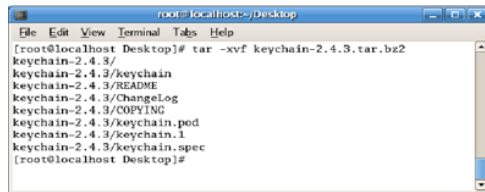


Fig. 4: Installing keychain

Next step, move the keychain-2.4.3 to /home directory and rename as keychain to shorten the name. This is done by typing `mv keychain-2.4.3 /home/keychain` command in terminal. Next, set the path of the keychain in the /etc/bashrc file to execute the keychain when u startup. Type `vi /etc/bashrc` and edit the following line to the bashrc file.

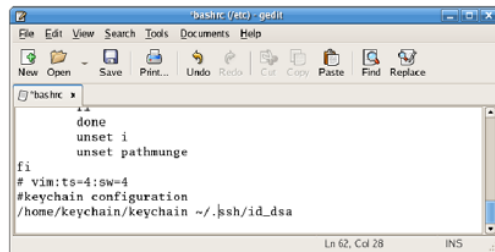


Fig. 5: Bashrc

A. Installation of GridMPI

Now, move to the configuration of GridMPI. The installation of GridMPI can be done by download the tarball of gridmpi from internet. Before extract the tar file, we need to start the path which we install the grid to, for the execution of it. First, we edit the /etc/bashrc file by adding the following line of command:

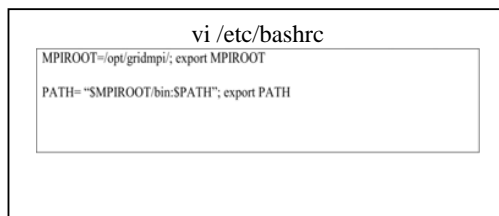


Fig. 6: Bashrc

B. Installing SKaMPI (Benchmarking tool)

For benchmarking tool, we choose SKaMPI . The installation of SKaMPI is similar to gridmpi but we do not need to set the path because it has the path set by default. Extract SKaMPI in /root/Desktop directory with `tar -zxvf skampi-5.0.4.tar.bz2` command in terminal and make install command to complete the installation.

V. TESTING AND RUNNING CLUSTER

A. Testing and Running the Cluster (GridMPI)

With all the installation complete, we have come to the testing part. We will first examine the gridmpi to make sure it works well under CentOS linux base. There are three nodes

which are running in linux with hostname apple, orange and grape. The configuration of hostname is by editing the /etc/hostname as shown in the 1st stage of BSD configuration. These nodes are bind to IP address in 192.168.10.0 network. The IP are as shown as below.

#vi /etc/hostnames		
192.168.10.1	apple.udm.edu	apple
192.168.10.2	orange.udm.edu	orange
192.168.10.2	grape.udm.edu	grape

Fig. 7: Nodes are bind to IP address

After hostname is being test ping the existing node and all should be fine if every node have their hostname file as configure as above.

In order to execute the gridmpi test, we need to change the directory to the extracted gridmpi.2.1.1 file to get the test program. Type the following command `cd /gridmpi.2.1.1/src/test/basic/`. Here, we have a default pi program that use to test the cluster nodes. We just type `mpicc pi.c` to compile the pi.c program to make sure it has no error. But before we can run the program, we need to edit the mpi_conf file which is just similar to lam-bhost.def file in lam-mpi which store the nodes that exist in the network. Edit the mpi_conf with `vi mpi_conf` command in terminal and add the apple, orange and grape nodes as follow.

```
apple
orange
grape
```

Fig. 8: Name of Nodes

Save the changes and proceed with the testing. Type the command `mpirun -np 2 ./a.out` in terminal. This command shows it run the pi program on the apple and orange nodes. We proceed with 3 nodes by prompting `mpirun -np 3 ./a.out` command in terminal as shown below.

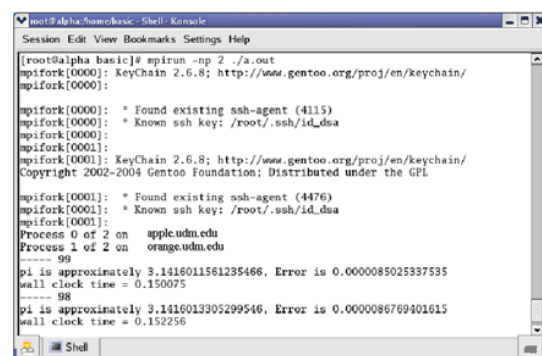
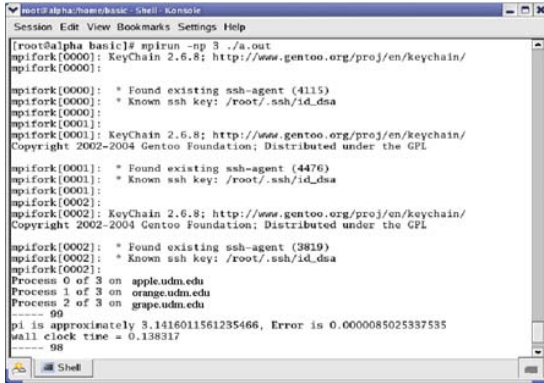


Fig. 9: Running GridMPI with 2 nodes

Note that, the time for the clock is 0.15~ with execution on 2 nodes, which is apple and orange node. With the execution of 3 nodes, we are expected to get a lesser time since the time to execute the result is decreased with the increase of the processor/CPU/nodes.



```

root@alpha:~/hadoop$ mpirun -np 3 ./a.out
mpifork[0000]: KeyChain 2.6.8; http://www.gentoo.org/proj/en/keychain/
mpifork[0000]:
mpifork[0000]: * Found existing ssh-agent (4115)
mpifork[0000]: * Known ssh key: /root/.ssh/id_dsa
mpifork[0001]:
mpifork[0001]: KeyChain 2.6.8; http://www.gentoo.org/proj/en/keychain/
Copyright 2002-2004 Gentoo Foundation; Distributed under the GPL
mpifork[0001]: * Found existing ssh-agent (4476)
mpifork[0001]: * Known ssh key: /root/.ssh/id_dsa
mpifork[0002]:
mpifork[0002]: KeyChain 2.6.8; http://www.gentoo.org/proj/en/keychain/
Copyright 2002-2004 Gentoo Foundation; Distributed under the GPL
mpifork[0002]: * Found existing ssh-agent (3819)
mpifork[0002]: * Known ssh key: /root/.ssh/id_dsa
mpifork[0002]:
Process 0 of 3 on apple.udm.edu
Process 1 of 3 on orange.udm.edu
Process 2 of 3 on grape.udm.edu
----- 99
pi is approximately 3.1416011561235466, Error is 0.0000085025337535
wall clock time = 0.138317
----- 98

```

Fig. 10: Running GridMPI with 3 nodes

With the result shown above, we know that the time is decrease with the number of increase nodes. Hence, the testing is a success and it was proven that, by develop the cluster could enhance the availability of the computers.

VI. CONCLUSION

Building Beowulf cluster based machine is one of the enhancements to the computer workload management technology. Cluster which bring a low cost alternative and improve the cluster-wide performance which create a convenient multiuser, time sharing environment for the execution of both sequential and parallel applications. Clusters are vital in order to solve tasks such data mining, financial market modeling, and DNA analysis.

REFERENCES

- [1] M.A. Baker and R. Buyya, Clusters serve up a challenge, *The Age* – newspaper, *Fairfax IT*, August 31st 1999.
- [2] Ian Foster. "Internet Computing and the emerging Grid". *Nature*, vol 408 issue 6815, 2000.
- [3] Becker, D. and Merkey, P., "The Beowulf Project", Online at: <http://www.beowulf.org>
- [4] B. Jacob, L. Ferreira, N. Bieberstein, C. Gilzean, J.Girard, R. Strachowski, S. Yu, Enabling Applications for Grid Computing with Globus, IBM, 2003.
- [5] <http://www.mcs.anl.gov/research/projects/mpi/>
- [6] <http://www.lam-mpi.org/>
- [7] "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." I. Foster, C. Kesselman, J.Nick, S. Tuecke; January, 2002.
- [8] Sterling, T., Salmon, J., Becker D. and Savarese, D. (1998), *How to Build a Beowulf*, MIT Press, Cambridge, Massachusetts USA.



Ahmad Faisal Amri Abidin obtained his Master of Computer Science from Faculty of Computer Science and Information Technology, Universiti Putra Malaysia in 2008. Currently, he is a Lecturer at Department of Computer Sciences, Faculty of Informatics, Universiti Darul Iman Malaysia.



Nor Surayati Mohamad Usop obtained her Master of Computer Science from Faculty of Computer Science and Information Technology, Universiti Putra Malaysia in 2009. Currently, she is a Lecturer at Department of Information Technology, Faculty of Informatics, Universiti Darul Iman Malaysia.