

# Development of Beowulf Cluster to Perform Large Datasets Simulations in Educational Institutions

Mkhuseli Ngxande  
Computer Science Department  
Fort Hare University, South Africa

Nyalleng Moorosi  
Computer Science Department  
Fort Hare University, South Africa

## ABSTRACT

This paper presents the design and development of the Beowulf cluster that can be used by institutions to perform research that requires high performance computing. In many industries and scientific applications there is often a need to analyse large datasets using computational power of distributed and parallel systems. The High Performance Computing (HPC) components are very expensive but with the use of commercial-off-the-shelf (COTS) hardware components, the costs can be lowered down. COTS provide inexpensive computing alternative to educational institutions to perform their research that need high performance computers. In High Performance Computing jobs are divided into several small jobs that can be distributed in to all the nodes and they run concurrently on the system, this is made possible by the use of parallel computing. This paper will address the advantage that the Beowulf clusters have in the high performance computing field as well as how it can be implemented using COTS. The most important part of this paper is to investigate the factors that affect the cluster.

## General Terms

High performance computing system.

## Keywords

High Performance Computing, Beowulf Clusters, Large data sets, Parallel Computing.

## 1. INTRODUCTION

High performance computing can be defined as a computing environment which applies supercomputers and computer clusters to address complex computational requirements [1]. HPC was introduced in the early 1960's by Seymour Cray at Control Data Corporation (CDC) [2]. Since then the field has expanded rapidly. Unfortunately, while the need for this specialized equipment increased with exponential growth of data, the equipment has largely remained very expensive. Clusters technologies are easier to understand and administer and they offer unrivalled availability and are largely expandable to unlimited number. Handling large datasets can be a huge problem for most institutions and industries because of high computational and storage requirements, which creates the need for specialized computers to perform the analyses.

These specialized computers can be used in many fields such as Bioinformatics, Computational Electromagnetics and Network Simulations. Almost every company or institution that deals with large data needs fast processing power and lot of storage. With the increasing availability of inexpensive and faster computers, more users are interested in gaining the technological benefits. There is no upper boundary to the needs of computer processing power; even with the rapid

increase in power, the demand is considerably more than what is available.

## 2. PARALLEL COMPUTING

Parallel computing is a way of dividing a large job into several tasks and using more than one processor simultaneously to perform these jobs [3]. Computer scientists have noticed the importance of parallel computing this technology has been used because it has the following advantages:

- Solve larger problems;
- Faster turn-around time;
- Overcome limits to serial computing;
- Cheap components to achieve high performance.

The field of HPC has grown rapidly but it has however been not accessible to under-resourced entities due to the high cost. This has led to the introduction different architectures such as symmetric multiprocessors, vector processors and cluster computing. These architectures each have advantages but the one that will be discussed in this paper is cluster computing architecture.

The cluster computing architecture is a concept where a set of loosely connected computers work together so they can be logically viewed as one computer. It is technique came up on distributed systems which are computers connected together which share computing tasks assigned to the system [4]. These connected computers communicate using the network to pass messages between them. Clusters consist of computers, and switches which are used for the communication. There are two types of cluster nodes: master node and computing nodes which are connected over the network.

This paper proposes the use of Beowulf cluster in the educational institution to carry out high performance computing research. This type of cluster is chosen because it has a low cost and produces high performance computing. This technique firstly appeared in 1994 and it was introduced by Thomas Sterling and Donald Becker at The Centre of Excellence in Space Data and Information Sciences (CESDIS) when they were commissioned to investigate whether clustered computers could perform heavy computational tasks at a greater capability than contemporary workstations [5]. Figure 1 shows the high level of Beowulf architecture where there is a master node which is connected to a public network and also a private network which contains compute nodes.



**Figure 1: High Level Beowulf Architecture [6].**

The compute nodes are connected to a switch so as to communicate with each other and also for a master node to distribute jobs to compute nodes. There are many benefits in using this type of cluster which are as follows:

- Cost-effective: The cluster can be built from inexpensive commodity components that are widely available;
- Keeps pace with technologies: It is easy to employ the latest technologies to maintain the cluster;
- Scalability: it is easy to expand the cluster;
- Flexible configuration: the users can tailor a configuration that is suitable to them and allocate the budget that meets the performance requirements of the analyses.

There are a number of types of cluster such as:

- Fail-over clusters: This type of cluster consists of a group of independent computers which are connected via a local network and are linked together by cluster software. They operate by moving resources between the nodes to provide service if system components fails [7].
- Load-balancing clusters: This type of cluster is usually used in web sites which have a high traffic, whereby several nodes host the same site. Then a request for a web page it is dynamically routed to nodes that have a lower load. This is a critical issue that is faced in parallel computing to ensure fast processing and efficient utilization [8].
- High performance clusters: This type of cluster is used to run parallel programs for time intensive computations and it uses computer clusters to address complex computational rudiments.

### 3. RELATED WORK

There is much work that has been done in past decades to develop high performance clusters that at a low cost and get a high performance.

Fan et al. proposed to use a cluster of Graphics Processing Unit (GPU) for high performance computing where they developed a parallel flow simulation using the Lattice

Boltzmann model (LBM) [9]. The Lattice Boltzmann model (LBM) was developed on a GPU cluster and have simulated the dispersion of airborne contaminants in the Times Square area in New York City [9]. The GPU cluster was developed for two purposes: first one was a GPU cluster for graphics and computation, and secondly one for visualization for interpretation large capacity data sets [9].

Hu and Evans developed a Power and Environment Awareness Module (PEAM) for Beowulf clusters [10]. The development of the PEAM addresses the issues that the Beowulf clusters encounter such as heat-induced hardware failure which makes large scale commodity clusters fail frequently and the cost effectiveness of Beowulf clusters is challenged by lack of means of adapting its power state according to varying work load [10]. The PEAM module was developed on a Beowulf cluster at Purdue University, which aimed at reducing the operational cost and increase the reliability of the cluster by reducing heat generation and optimizing workload distribution in an environment aware manner [10].

Hsu and Feng conducted a feasibility analysis of power awareness in commodity based high performance clusters [11]. Hsu and Feng study revolves around a 16 processors Opteron-based Beowulf cluster. The Beowulf cluster was configured as four nodes of quad processors. The study showed that a 5 % performance slowdown can be traded off for average of 19% system energy which is saving and 24% system power reduction [11]. Hsu and Feng further stated that power efficiency is the most critical part for developing cost-effective, small-footprint clusters [11]. Hsu and Feng performed various tests such as benchmark tests of different software's to compare power consumption between them. The HPL and NAS MPI benchmarks were compared so as to see which can save power.

Al-shaikh et al. proposed to build High-Availability (HA) clusters based model for high performance computing [12]. Al-shaikh et al. also proposed to investigate the hardware and management layers of the HA-HPC cluster design together with the parallel applications layer [12]. Al-shaikh et al. stated that one of the challenges in a clustered environment is to keep system failure to minimum levels and achieve the highest possible level of system availability [12]. Al-shaikh et al. analyzed small scale HA-HPC functionality and performance, for the evaluation and eight- nodes cluster were used.

### 4. IMPLEMENTATION

The Beowulf implementation that is proposed was built in a computer with the following specifications: Pentium (R) Dual-core E5200 @ 2.50 GHz, 2.50 GHz processor, 2 Gigabytes of RAM and 250 Gigabytes of hard drive. The connection medium was a Cisco system catalyst 2960 series switch. The cluster runs on Linux Ubuntu 13.10 for master node and a mini version of Linux Ubuntu 13.10 for the compute nodes. The cluster consists of one master node and five compute nodes.

There are standard software that must be installed for a cluster to work properly, these software are Message Passing Interface (MPI) which is used for distribution of the jobs, the Network File System (NFS) which is for sharing files remotely and the Secure Shell (SSH) which is responsible for remote login to the compute nodes. These are most important software in the cluster to be complete and work properly.

## 4.1 Message-passing Libraries

This is a standardized system which is portable message-passing system that is used in clusters, parallel computers and also in heterogeneous networks [13]. It is not a programming language but it is used to define the syntax of the core library for the use of portable message-passing languages such as C++, C and FORTRAN. These message-passing libraries are needed when one needs to perform intensive calculations, it is use to divide and distribute independent jobs to different computers. For this project MPICH2 was used. Message-passing interface because it is the most used library for numerical analysis. The goal of message passing interface is to provide a widely used standard for writing message passing programs, also the interface attempts to more portable, efficient, flexible and practical. The MPI is the only message passing library that is considered a standard because it is supported on virtually all high performance computing platforms.

The MPICH2 is built so that a number of communication infrastructures can be used, these are called devices which are most relevant for the Beowulf environment [6]. MPICH2 is freely available online software that is portable implementation of MPI. It is a standard for message passing for distributed memory applications that are used in parallel computing. The runtime for MPICH2 consist of asset of daemons such as mpd and hydra process managers. The MPICH2 will be installed on all the nodes. The MPICH2 needs to be installed in the shared file which will have the entire executable that is needed to run MPI programs. The following steps are taken in installing MPICH2-1.4 version. After the software version was downloaded, the tar file was extracted and copied to the shared directory.

## 4.2 Network File System (NFS) and Secure Shell (SSH)

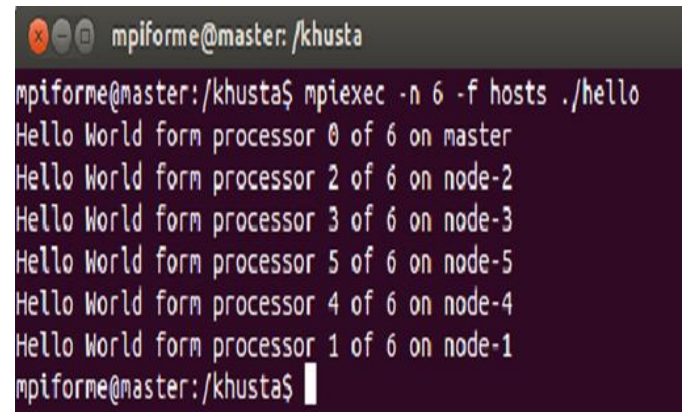
The files and programs used for MPI jobs need to be available to all the nodes in the cluster. These files will be accessed on the master node by the compute nodes. The NFS enables the mounting of the files remotely so they can be accessed as if they are in a local directory. The node where the files will be saved acts as a NFS server and the NFS kernel-server portmap is installed. For the other nodes that will access the file remotely, NFS kernel-common portmap must installed on the clients.

On the master node, the directories that will be shared needs to be created and its owner change, so it can be accessed by compute nodes. Once the directory is created it needs to be exported to other nodes, the export file is edited with the details of all the compute nodes. In this file the nodes that will share this directory are listed and the permission on how to use it. Once the conditions are stated the NFS server must be restarted so as to update the server. On the compute nodes the same directory is created and it is mounted with the master IP address and the directory that will be shared. To prevent the mounting of the directory every time the compute nodes are booted, the fstab file is edited by adding the IP address of the node that contains the directory and also the name of the directory. When this is done the directory can be mounted successfully.

For the cluster to work, the master node needs to communicate with compute nodes and vice versa. This communication is made possible by installing SSH server at all nodes. Once the SSH is installed in all the nodes, the SSH must be tested by logging into another node, this will create a file in the SSH directory which is called unknown\_hosts.

The next step is to login to the user that was created and generate the keys that will be distributed to all the nodes. This is done by “ssh-keygen” command and, which creates two files and the id\_dsa.pub file is transferred to other nodes. Once the transfer is done the contents of that file is copied to another file which is called authorized\_keys and then the communication is successful.

Once all the above applications are installed and working correctly, a simple program was compiled to test if all nodes are communicating to one another and the jobs are divided to all of them. Figure 2 below show a hello program that was used to test the communication between the nodes.



```
mpiforme@master: /khusta
mpiforme@master: /khusta$ mpiexec -n 6 -f hosts ./hello
Hello World form processor 0 of 6 on master
Hello World form processor 2 of 6 on node-2
Hello World form processor 3 of 6 on node-3
Hello World form processor 5 of 6 on node-5
Hello World form processor 4 of 6 on node-4
Hello World form processor 1 of 6 on node-1
mpiforme@master: /khusta$
```

Figure 2: Hello World output results.

## 4.3 Htop System Monitoring

The htop system monitoring is a tool that is used in Linux to allow the users to monitor resources and the processes that are running in real-time. This software can be used to check the usage per Central Processing Unit (CPU) by the programs that are running. The htop software was installed so as to check the usage of the programs that are running on the cluster.

An excellent collection of benchmarks that can be used on Beowulf clusters is HPL (High performance Linpack). This benchmark suite allows making accurate predictions on which type of computational problems that can be suitable to be solved on the particular Beowulf cluster. This measures the performance of the system, this can be bone theoretical or practical. For the theoretical performance one can submit the hardware specifications on the HPL website to find the theoretical system performance.

## 5. EXPERIMENTS

The following experiments were performed using the Linux cluster, these experiments ranged from small to large jobs. A theoretical and practical performance of the cluster was also performed. The theoretical performance of the cluster is not based on the actual performance that is obtained from a benchmark tests but it depends the cluster specification to determine the peak rate of execution of floating point operation for the computer.

To calculate the theoretical peak performance of the cluster, first it is required to calculate the peak of a master node in GFlops and then multiply the node performance by the number of the nodes that the cluster have. The following is the standard formula used for node theoretical peak performance:

Node performance in GFlops = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node).

For cluster:

CPUs based on Intel Premium(R) E5200 (2.50GHz 2-cores):

$2.50 * 2 * 4 = 20$  GFlops

Five PC Clusters Theoretical Peak Performance:

$20 \text{ GFlops} * 5 = 100 \text{ GFlops}$

The High Performance Linpack package was used to test the actual performance of the Beowulf cluster. Figure 3 show the actual performance of the Beowulf cluster.

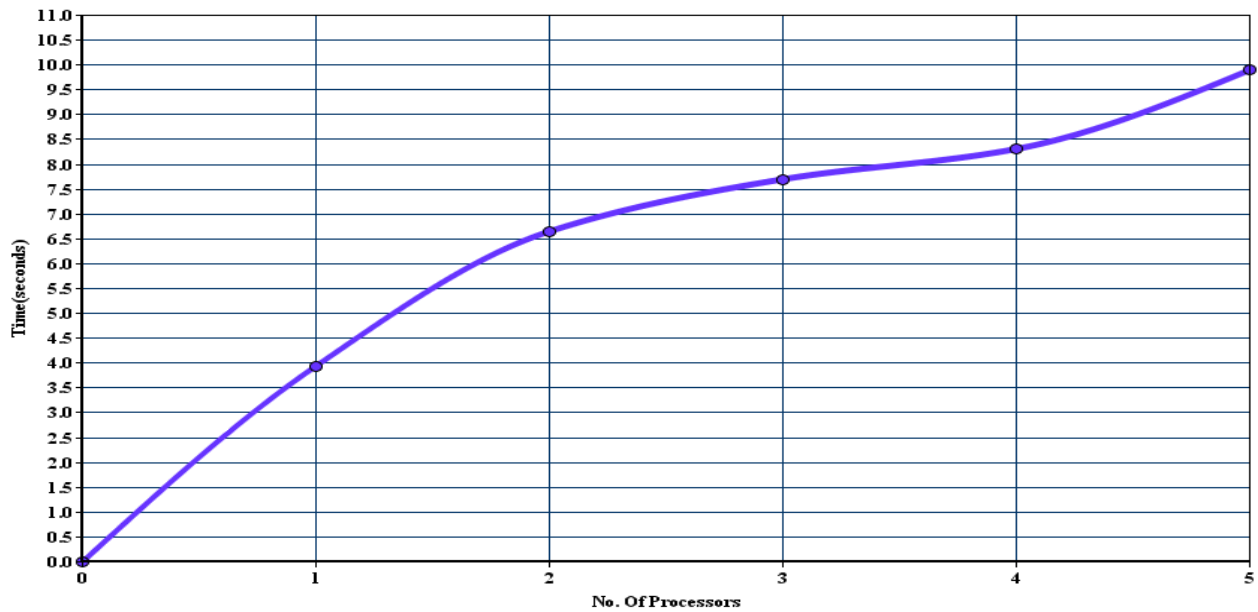


Fig 3: HPL results. Where the y-axis has is the speed (GFlops) and x-axis is the number of processors.

### 5.1 Ring MPI Experimentation.

This program estimates the time taken to send a vector of F double precision values through each process in a ring. Process 0 sends F double precision values to process 1, which passes them to process 2 and so on until to the last process sends back the value to process 0. The time for transmission is

recorded and the process is repeated in different array sizes F. the F double precision values are random values. The different times are recorded which are minimum, average and maximum time taken. The computations are done in parallel using MPI. Figure 4 shows the performance of the ring\_mpi using all the five compute nodes.

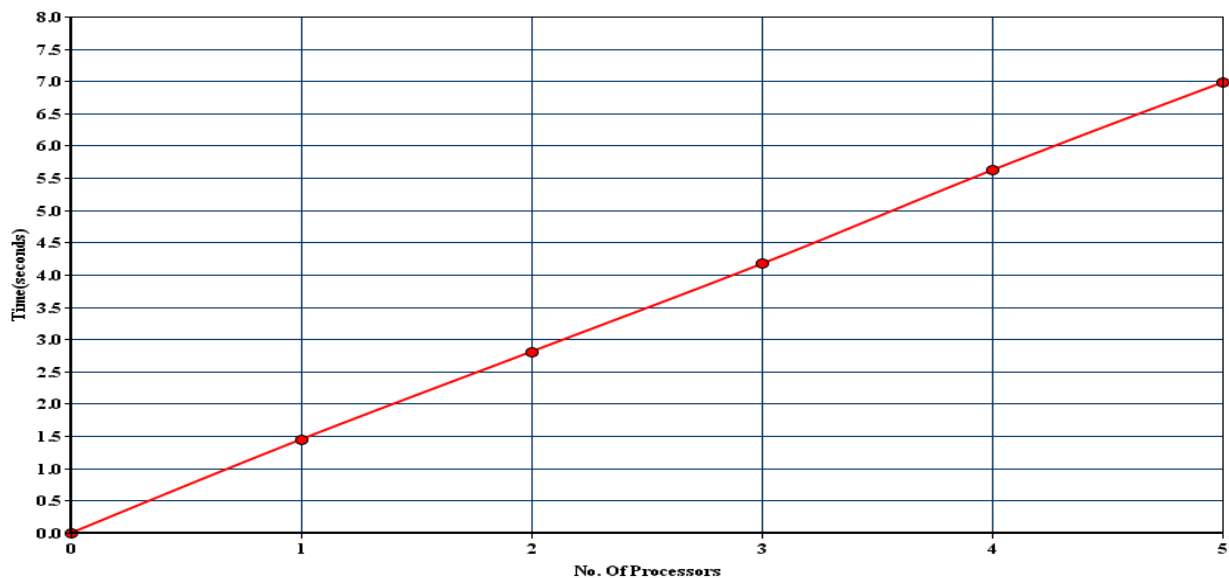


Figure 4 : Ring MPI Results. Where y-axis is the time (seconds) and x-axis is the number of processors.

## 5.2 Quad\_MPI Experimentation

This program approximates an integral using a quadrature rule. It uses MPI to compute the results in parallel. The estimation of an integral of  $f(x)$  from A to B is  $50/(\pi*(2500*x*x+1))$ .

- A = 0.000000
- B = 10.000000
- N = 9999999
- Exact = 0.4993633

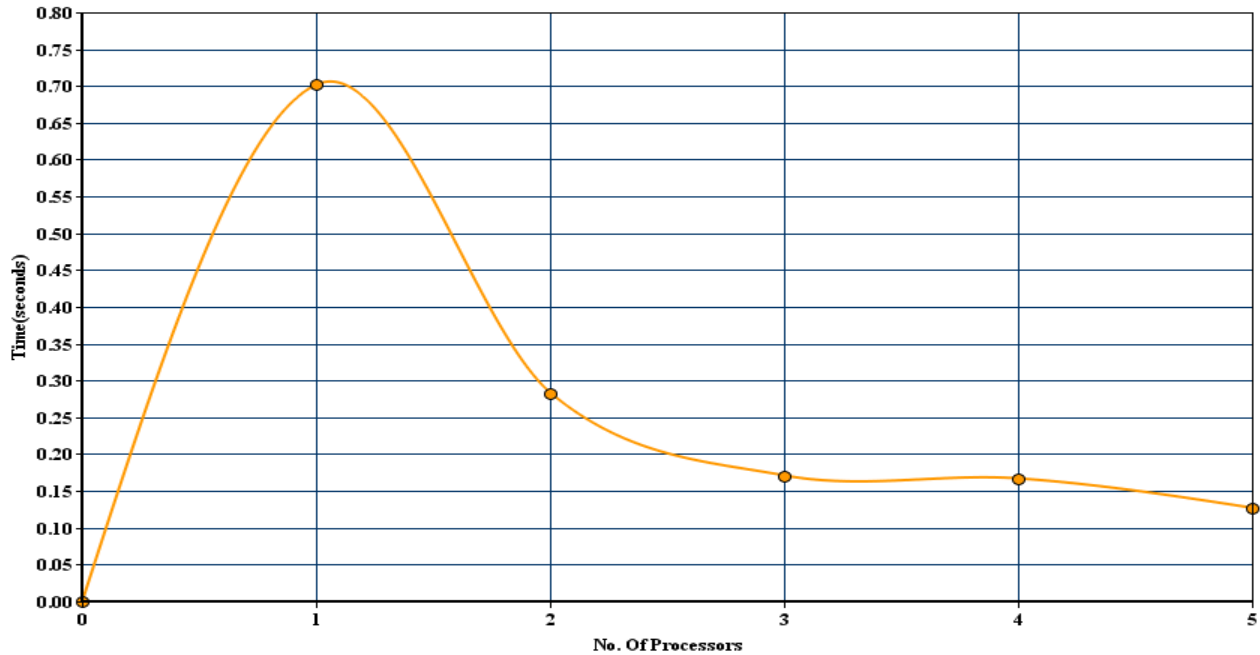


Figure 5: Where y-axis is the time (seconds) and x-axis is the number of processors.

## 5.3 Circuit Satisfiability Experiment

This program demonstrates for a particular circuit an exhaustive search for solutions of the circuit satisfy problem. The program uses MPI to carry out the solution in parallel. This problem assumes that a logical circuit of AND, OR and NOT gates are given, with N binary inputs and a single output. It determines all inputs which produce a 1 as the result. The general problem is NP complete, so there is no known polynomial-time algorithm to resolve the general case.

The natural way to search for solutions then is exhaustive search by the program. In an interesting way, this is a very risky and separate version of the problem of maximizing a scalar function of multiple variables. The difference is that here both the input and results only have the values 0 and 1, rather than a constant range of real values. This problem was a natural contender for parallel computation, since the separate evaluations of the circuit are completely independent. Figure 6 demonstrates the circuit satisfy problem in parallel computing.

## 6. DISCUSSION

Clusters can reduce the overall computational time, this part will discuss the observed results on the experiments that were performed, the small tasks and the big task will be discussed and their effect on the performance of the cluster.

The MPI divides the computation among the processors that the cluster has and each contribution of a processor is determined. The error of the computation is also calculated. The overall time taken by the program is calculated and displayed. Figure 5 shows the performance of the Quad\_MPI program running on the Beowulf cluster.

## 6.1 Small Tasks Observations and Performance

The small tasks that were performed of the experiments showed that jobs with small numbers such as Ring\_mpi and Prime Numbers experiments when there number of processors are increased the performance time takes longer than in single processor. Their communication time is bounded and because the sequential runtime is so small, the time to send and receive from the master node makes the programs take longer with more nodes.

This delay is caused by the time to distribute the work and the time for the master node to collect the results. Such programs can also be performed in a single computer but it is done so as to show that how the cluster handles small jobs. The significance of these experiments was to find out which jobs are suitable for the cluster because with small jobs can also be executed but the problem is that with clusters there is that communication between the nodes that takes some time.



## 6.2 Large Tasks Observations and Performance

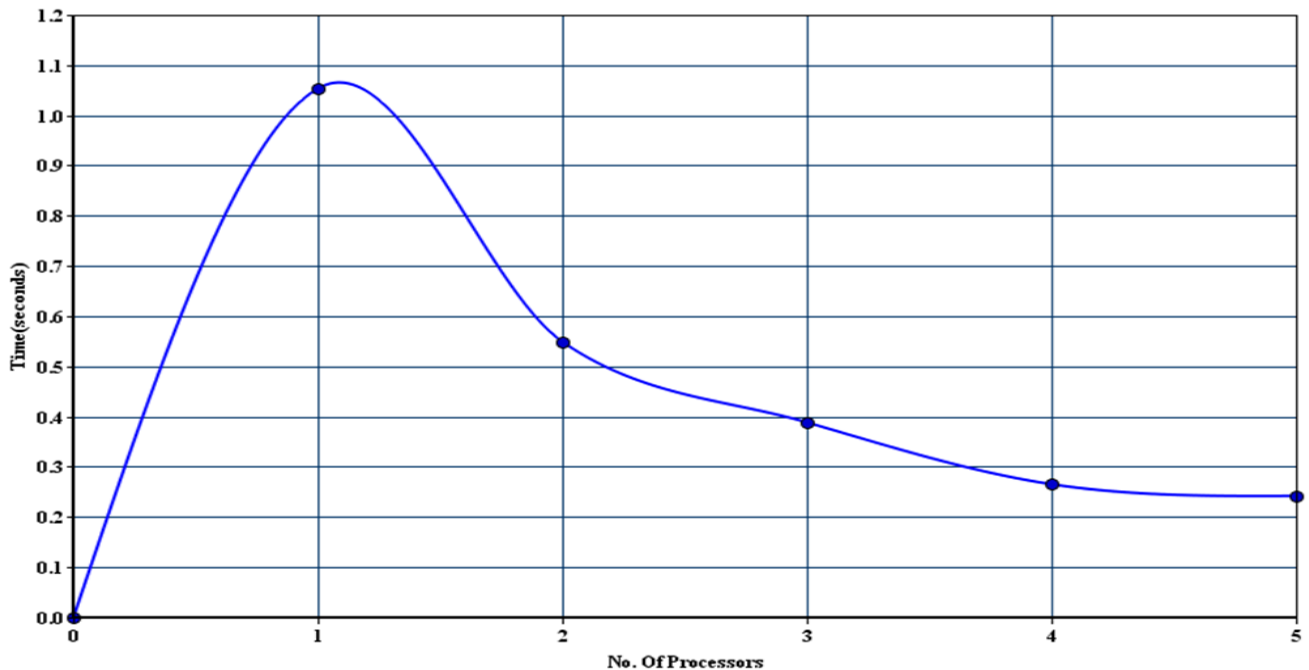


Figure 6: Circuit satisfiability Results. Where y-axis is the time (seconds) and x-axis is the number of processors.

The large jobs on the cluster are bounded by sequential computation time just like in small jobs but with more processors the communication factor also takes over. Because the sequential time for larger jobs is also large this is the advantage on the cluster to scale better than the small jobs.

With larger jobs requiring more memory and space so this part is handled well by the cluster because their memory and space are distributed. When the same jobs are executed in single PC there was an error message saying that it is out of memory and it abort the process.

### 6.3 Efficiency of the Cluster

The efficiency of the cluster, which was 19.8%, was unexpected because the predicted efficiency was 83.3%, which was found in the cluster calculator. This huge difference was because of the condition of the computers that were used. The processors that were installed on the computers were very small and also the memory which it's self-had the effect on the cluster. Some of the effects that were stated in chapter three also played a role in the efficiency of the cluster. The factor that was taken note was the network part which was limited and some of the applications were not installed because of the security measures. In the case of cluster the dataset may be required to reside in memory alongside with the messaging traffic meant for the other processors. The memory plays a huge part in the efficiency and in the development of the cluster the memory was too low and it was not taken into consideration that it will influence the percent of efficiency.

### 6.4 Factors that affected the Cluster

One of the factors that are affecting the cluster was the network that was used because of the overhead and its speed, the messages were delivered slowly and that cost the overall performance. The memory of the distribution which was so low contributed in the performance of the cluster. Looking at the message passing interface there were number of factors that were encountered such as the message size, process number and running processes.

For the message size which played a very important part to MPI application performance, this effect can be influenced by latency and number of processors. For large messages the application yield a better performance because sending smaller size messages reduces the performance of the application because the latency affect mostly affect short messages.

Adding another node in the cluster can reduce the computation time but increases the communication time. This was observed when increasing the number of nodes for the experimentation. When there were many processing running in the cluster there was a delay in the production of results.

## 7. CONCLUSION

This paper addresses the advantages of developing a Beowulf cluster for educational institution to undertake their research that need more computing power and space. The Beowulf cluster can be built using COTS hardware to cut down the costs of buying expensive equipment.

The Beowulf cluster have great advantages and are widely used to undertake research, Beowulf clusters are easily configured and also easy to expand the performance by adding another nodes to the cluster. Large datasets to be analyzed can be easily performed in the clusters.

As the network technologies advances with time and its getting cheaper to the users and faster, a new computing model called Grid computing, has evolved. Grid computing which is a collection of computer resources forms a many locations to reach a common goal. It provides to users a single point of access which is just a website interface to these distributed resources. The users of the grid can submit their jobs as many as they can without being concerned about where their jobs will run. Grid may range from single systems to supercomputers farm that uses thousands of processors.

## **8. ACKNOWLEDGEMENTS**

This work is based on the research undertaken within the Telkom CoE in ICTD supported in part by Telkom SA, Tellabs, Saab Grintek Technologies, Easttel, Khula Holdings, THRIP and National Research Foundation of South Africa (UID : 84006). The Centre of High Performance Computing made this work possible by providing workshops in the field of High performance Computing. The opinions, findings and conclusions or recommendations expressed here are those of the authors and none of the above sponsors accepts no liability whatsoever in this regard.

## **9. REFERENCES**

- [1] A. M. Middleton, "White Paper HPC Systems : Introduction to HPC (High-Performance Computing Cluster)," 2011.
- [2] C. P. Sosa, "HPC: Past, Present, and Future," 2011.
- [3] J. Nakano, "Parallel computing techniques," 2004.
- [4] M. O. Cortada, "High performance computing on biological sequence alignment Miquel Orobítg Cortada High performance computing on biological sequence alignment," 2013.
- [5] K. Anderson, D. Aaronson, and P. Karlsson, "An evaluation of the system performance of a Beowulf cluster by," 2001.
- [6] S. CHAVAN, "Design and Implementation of High Performance Computing Cluster for Educational Purpose," no. June, 2012.
- [7] W. Highleyman, "Windows Server Failover Clustering," no. April 2009, pp. 1–7, 2010.
- [8] J. Guo and L. N. Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications," vol. 17, no. 11, pp. 1–14, 2006.
- [9] Z. Fan, F. Qiu, A. Kaufman, and S. Yoakum-stover, "GPU Cluster for High Performance Computing," vol. 00, no. 1, 2004.
- [10] F. Hu and J. J. Evans, "Power and environment aware control of Beowulf clusters," no. May 2008, pp. 299–308, 2009.
- [11] C. Hsu and W. Feng, "A Feasibility Analysis of Power Awareness in Commodity-Based," no. Cluster, pp. 1–10, 2005.
- [12] A. B. R. Al-shaikh, M. Sechi, and M. Notare, "Towards Building a Highly-Available Cluster Based Model for High Performance Computing 3. The Beowulf Cluster Architecture," pp. 1–8, 2006.
- [13] K. Yelick, "Message Passing Programming (MPI) Slides adopted from class notes by what is MPI ?" 2001.