東海大學

資訊工程與科學研究所

碩士論文

指導教授：楊 朝 棟 博士

在東海校園內實作一個無碟叢集計算環境

Implementation of a Diskless Cluster Computing Environment in a Computer Classroom of Tunghai Campus

研 究 生：謝文峰

中華民國九十七年六月

摘要

在本篇的論文中，將利用校園內電腦教室的電腦，配合無硬碟工作站的技術，建置高效能計算平台，期望能夠將校園中的電腦資源做最有效的發揮。首先本文使用了兩套建置叢集的軟體，一套是由國家高速網中心所開發的自由軟體教學平台「企鵝龍」(無硬碟遠端開機)所建置的叢集系統來和專為叢集系統建置所開發的「可開機式叢集光碟」比較在叢集計算效能的優劣，以證明企鵝龍有不錯的叢集效能。而企鵝龍軟體的開發本身即是做為自由軟體教學平台，因此為了希望將自由軟體帶進校園的教育環境中，本文利用企鵝龍套件軟體和使用者控制例行性命令的指令軟體使電腦教室的電腦能在原本的微軟作業系統環境、自由軟體教學平台和叢集系統中進行自動或手動切換，讓使用者使用高效能計算系統就像是隨插即用一樣的便利；在叢集系統方面，將電腦教室32台實際佈署叢集系統應用於平行計算；在自由軟體教學方面，可以發現不論在電腦開機時間和自由軟體開啟時間都有不錯的表現。另外，我們還將該叢集系統安裝生物計算的軟體，來應證該平台可以實際的實用性。另外，為了隨時掌握各節點電腦的狀態，使用監控軟體以便得到叢集節點相關的資訊如：記憶體、中央處理器、硬體輸出和輸入使用率等，並且在比較網路環境和交換檔各方面的效能，試圖找出在電腦教室中最佳的叢集環境，最後，使用高效能運算挑戰指標測試軟體來瞭解叢集系統的運算效能。

Keywords: 企鵝龍，可開機式叢集光碟，電腦教室，無硬碟工作站，個人電腦叢集，高效能運算挑戰指標測試

ABSTRACT

The objective of this paper is to implement and evaluate a High Performance Computing environment by clustering idle PCs (personal computer) with Diskless slave nodes on campuses in order to obtain the effectiveness of the largest computer potency. Two sets of Cluster platforms BCCD and DRBL are used to compare parallel computing performance. In addition, it is to prove that DRBL has better performance than BCCD in this experiment. The original purpose of DRBL is created to facilitate instruction for Free Software Teaching platform, in order to achieve the purpose DRBL is applied to the computer classroom with 32 PCs, enabling PCs to be switched manually or automatically among different OS (operating system) of Windows, Free Software Teaching and PC Cluster. The bioinformatics program, mpiBLAST, is executed smoothly in the Cluster architecture as well. From management's view, the state of each Computation Node in Cluster is monitored by "Ganglia", an existing Open Source. We can gather the relevant information of CPU; Memory; Network Load for each Computation Nodes in every network section. Through comparing various aspects of performance, including performance of Swap and different network environment, this paper attempted to find out the best Cluster environment in computer classroom at school. Finally, HPCC is used to demonstrate Cluster performance.

Keywords: DRBL, BCCD, computer classroom, Diskless, PC Cluster, HPCC

ACKNOWLEDGEMENTS

I would like to express my gratitude to all the people who have made writing this thesis a more pleasant task. Particular thanks to Professor Chao-Tung Yang, principal advisor of mine, his encouragement, guidance, and support were invaluable in my work which no matter the capability in the open discussions, or the preciseness in thesis writing. Professor Yang gave me a deep influence and inspiration. I would also like to thank Professor Hsiao-His Wan and Professor Cheng-Chung William for their valuable comments and advice given while serving on my reading committee.

There are many people whom I would like to thank; especially my group-mate Hao-Yu Tung and Jen-Hsiang Chang gave me a lot of assistance for completing my thesis. Many of my classmates have encouraged and supported me through this research. For this reason, I could complete my work without fear.

Last, I must be grateful beyond place to my parents, who at an early age instilled a love of learning and through the years gave me all of their support and encouragement to pursue it.

TABLE OF CONTENTS

LIST OF TABLES

List of Figures

Chapter 1 INTRODUCTION

1.1 Motivation

In the campus, there are many computer resources in the computer classroom whereas resources left unused most of the time in the school. It is because, if the actual operation time of school computer room can be maximize to 8 hours a day, this paper is to try to expand the computer operation time to its full capacity as 24 hours a day. What we want is to consider the cost-effective utilization is to the largest computer potency.

Signal technological improvements over the past few years in areas such as microprocessors, memory, networks, and software have made it possible to cluster groups of inexpensive personal computers and/or workstations into a cost effective system that functions in concert and possesses tremendous processing power. Cluster computing is not new, but in company with other technical capabilities, particularly in the area of networking, this class of machines is becoming a high-performance platform for parallel and distributed applications. Although history of PC Cluster is short, it is quite fast to develop. In the recent decade, PC Clustering has developed flourishingly for a number of applications. Efficiency and economy that PC Cluster demonstrated so far have attracted many researchers who need High Performance Computing. Furthermore, regarding either existing PC or newly purchased PC can be linked to promote PC Cluster to sweep across all over the world. Cluster has many applications in different fields. For example, a Linux Cluster has become significant in the film industry for rendering quality graphics. In the movie Titanic, a Linux cluster was used to render the background in the ocean scenes. This paper aimed to find out one PC Cluster [1] suitable for campus in the article and idle computer resources in the campus can reach their full capacity.

1.2 Contributions

Construction of PC Cluster in the past was that every node all needed to install relevant services and it had better performance in the past [2].But it will often meet such questions as a large number of machines need managing, erecting and hard disk equipment is damaged in the general computer classroom. Therefore, it is recommended that PC Cluster in the classroom uses Diskless technology [3] to centralize all nodes services, and can reduce the use of consumptive material of hard disk. Therefore, two sets of Cluster between BCCD (Bootable Cluster CD) [4] and DRBL (Diskless Remote Boot in Linux) [5] are compared in respects of parallel computing performance, convenience of OS switch over different systems, attempt to demonstrate that DRBL is suited to using in the campus.

Plug and Play (PnP) means that you can connect a device or insert a card into your computer and it is automatically recognized and configured to work in your system. Equally, we create switching method to switch different systems among Windows, Cluster, and Free Software Teaching, serves the similar function likely PnP.

At present, computers in most schools were installed with Microsoft Windows as OS (operating systems). By DRBL switching method, there is no need to alter the existing computer OS via Diskless technology, this can benefit, not only switching environment into parallel computing readily, but also bringing spirit of the Free Software into the education in the school. We can use DRBL to build Free Software Teaching platform. Moreover, the advantage of Free Software is that school does not need to buy a lot of copyright software (including OS) for teaching if the budget is limited. Therefore, this paper is also proposed to use the free software- such as document (OpenOffice.org), drawing (Gimp), network browser (Firefox)- to construct Free Software Teaching platform for promoting the pluralism of the Information Education. In conclusion, we hope that Free Software Teaching of DRBL not only can reduce the expenditure of budgets but also this promotion cultivates the “Freedom and Open” concept to teachers and students.

By switching method, computers can be turned into Cluster system automatically or manually and the computer resource can be got most effectiveness use.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In chapter 2, we review the Cluster Computing, introduction of Diskless PC Clusters, the two major implementations of Message Passing Interface, and related work. In chapter 3, we try to fine out the best cluster system by introducing two set of Diskless PC Cluster systems and discuss details of technologies used in the construction of cluster platform and present related performance evaluation. In chapter 4, we use diskless technology to deploy 32 PCs in computer classroom of campus. The switching method among local OS, cluster system and Free Software Teaching platform is presented. In addition, different Cluster environment performance evaluation is discussed which tried to find out the Cluster environment fit in the school. The bioinformatics software mpiBLAST is also executed to demonstrate that the PC Clusters architecture with diskless slave nodes is workable in real life. Our experimental results as mpiBLAST, HPCC are discussed in chapter 5. Finally, in chapter 6, we make some conclusions and feature works.

Chapter 2 BACKGROUND REVIEW

2.1 Beowulf Cluster

Beowulf [6] is a design for high-performance parallel computing Clusters on inexpensive personal computer hardware. The name comes from the main character in the Old English epic poem Beowulf. A Beowulf Cluster is a group of usually identical PC computers running a Free and Open Source Software (FOSS) Unix-like operating system, such as BSD, Linux or Solaris. They are networked into a small TCP/IP LAN, and have libraries and programs installed which allow processing to be shared among them.

There is no particular piece of software that defines a Cluster as a Beowulf. Commonly used parallel processing libraries include MPI (Message Passing Interface) and PVM (Parallel Virtual Machine). Both of these permit the programmer to divide a task among a group of networked computers, and recollect the results of processing. Different libraries [7] based on the MPI standards exist, and two commonly used implementations are MPICH [8] and LAM/MPI [9]. To make use of multiple processes each executed on a separate processor, we need to apply parallel computing algorithms.

There are two common types of parallelism: LAM/MPI and PVM.

- PVM: This is a master-worker approach and is the simplest and easiest to implement. It relies on being able to break computations into independent tasks. A master then coordinates completion of these independent tasks by worker processes.
- LAM/MPI: This is for use when computations cannot (or cannot easily) be broken into independent tasks. In this kind of parallelism, the computation is broken down into communicating, interdependent tasks.

We used LAM/MPI for our cluster system. LAM/MPI is a high-quality open-source implementation of the Message Passing Interface specification, including all of MPI-1.2 and much of MPI-2. Intended for production as well as research use, LAM/MPI includes a rich set of features for system administrators, parallel programmers, application users, and parallel computing researchers.

A common misconception is that arbitrary software will run faster on a Beowulf. Software must be revised to take advantage of the Cluster. Specifically, it must perform multiple independent parallel operations that can be distributed among the available processors. A Beowulf cluster uses multi-computer architecture. It features a parallel computing system that consists of one or more master nodes and available compute nodes, or cluster nodes, interconnected via widely available network interconnects. All of the nodes in a typical Beowulf cluster are commodity systems-PCs, workstations, or servers-running commodity software such as Linux. The master node acts as a server for Network File System (NFS) and as a gateway to the outside world. As an NFS server, the master PC provides user file space and other common system software to the compute nodes via NFS. As a gateway, the master node allows users to gain access through it to the compute nodes. Usually, the master node is the only machine that is also connected to the outside world using a second network interface card (NIC). The sole task of the compute nodes is to execute parallel jobs. In most cases, therefore, the compute nodes do not have keyboards, mice, video cards, or monitors. All access to the client nodes is provided via remote connections from the master node. Since compute nodes do not need to access machines outside the cluster, nor do machines outside the cluster need to access compute nodes directly, compute nodes commonly use private IP addresses, such as the 10.0.0.0/8 or 192.168.0.0/16 address ranges. From a user’s perspective, a Beowulf cluster appears as a Massively Parallel Processor (MPP) system. The most common methods of using the system are to access the master node either directly or through Telnet or remote login from personal workstations. Once on the master node, users can prepare and compile their five parallel applications, and spawn jobs on a desired number of compute nodes in the cluster. Applications must be written in parallel style and use the message-passing programming model. Jobs of a parallel application are spawned on compute nodes, which work collaboratively until finishing the application. During the execution, compute nodes use standard message-passing middleware, such as Message Passing Interface (MPI) and Parallel Virtual Machine (PVM), to exchange information.

Cluster computing focuses on platforms consisting of often-homogeneous interconnected nodes in a single administrative domain:

- Clusters often consist of PCs or workstations and relatively fast networks,
- Cluster components can be shared or dedicated,
- Application focus is on cycle-stealing computations, high-throughput computations, and distributed computations.

2.2 Message Passing Interface

Message Passing Interface (MPI) is a specification for an API that allows many computers to communicate with one another. It is used in computer clusters and supercomputers. There are many MPI implementations for parallel processing; most of the used libraries are MPICH and LAM.

2.2.1 MPICH

MPICH is a robust and flexible implementation of the MPI (Message Passing Interface). MPI is often used with parallel or distributed computing projects. MPICH is a multi-platform, configurable system (development, execution, libraries, etc) for MPI. It can achieve parallelism using networked machines or using multitasking on a single machine.

2.2.2 LAM

LAM is an implementation of the Message Passing Interface (MPI) parallel standard that is especially friendly to clusters. It includes a persistent runtime environment for seven parallel programs, support for all of MPI-1, and a good chunk of MPI-2, such as the dynamic functions, one-way communication, C++ bindings, and MPI-IO.

2.3 Diskless Technology

Diskless Linux Computers do not have any hard disk, floppy drives and tape drives. They offer significant savings in "Total Cost of Ownership" by eliminating the maintenance costs. You can accomplish Diskless Linux Computer by one of the following two methods:

- Boot directly a Live Linux CDROM. The Linux CDROM is live Linux system, which contains all the applications and programs and loads the software into RAM disks.
- Recent Linux kernels offer the possibility to boot a Linux box entirely from network, by loading its kernel and root file system from a server. In that case, the client may use several ways to get the first instructions it has to execute when booting home made eproms, special network cards implementing the RARP, BOOTP or DHCP protocols,

cdroms, or bootloaders loaded from a boot floppy or a local hard drive.

Diskless Linux computer will become immensely popular and will be the product of this century and in the next century. The diskless Linux computers will be very successful because of the availability of high-speed network cards at very low prices. Today 100 Megabit per second (11.92 Megabytes per sec transfer rate) network cards are common and in about 1 to 2 years 1000 Mbit (119.2 Megabytes per sec transfer rate) network cards will become very cheap and will be the standard. As Linux was first been adopted by the HPC community the installation practice of standard cluster started with an installation of Linux on every node. This means that for a 50 node cluster, with each installation taking at least 1 interactive hour, it will take an engineer more then a 40 hour work week just to install the Operating Systems for a mid sized cluster. Including the installation of the cluster tools, customizations, optimizations it is not unreasonable to spend 80-120 engineering hours to get the system built. Diskless PC cluster is defined as a computer cluster system without any local storage media in cluster slave computing nodes, i.e., no floppy or hard drive. In general, a diskless PC cluster depends solely on a LAN-based boot server node using DHCP to provide the operating system from which the diskless PC slave-computing node can boot. A slave computing node of diskless PC cluster also requires a network adapter interface and boot ROM (also known as boot PROM) to communicate with the boot server node of such cluster system.

Diskless PC clusters offer several advantages over traditional workstations in the area of manageability and security, such as:

● No moving parts, so they are less susceptible to dust, noise and vibration.

● Hard disk or floppy failure is no longer an issue.

● Less security risk since no data is stored locally.

● Easier to replace than traditional workstation - no OS and/or software re-installation is required.

2.4 DRBL

DRBL stands for Diskless Remote Boot with Linux. This solution is solely developed and implemented by people in National Center of High-performance Computing (NCHC), Taiwan. Administrators interested in a cost-efficient, powerful, and easy-to-manage thin client server should consider Diskless Remote Boot in Linux (DRBL), a server-based open source application that lets organizations deploy GNU/Linux across many clients. DRBL lets clients boot a Linux image over the network, so client hard drives can retain their own operating systems, or clients can do without hard drives altogether. DRBL uses PXE/ETHERBOOT [10], NFS [11], and NIS to provide services to client machines. Once the server is ready to be a DRBL server, then the client machines can boot via PXE/ETHERBOOT (Diskless). DRBL does NOT touch the hard drive of the clients, so other Operating Systems (for example, M$ Windows) installed on your client machines will not be affected. This may be important in a phased deployment of GNU/Linux, where users still want to have the option of booting to Windows and running Office. DRBL allows you to be flexible in your deployment of GNU/Linux. DRBL is an open source solution to managing the deployment of the GNU/Linux operating system across many clients. Imagine the time required to install GNU/Linux on 40, 30, or even 10 client machines individually! DRBL allows for the configuration all of your client computers by installing just one server machine. DRBL’s features:

● Peacefully coexists with other OS.

● Simply install DRBL on a single server and all your clients are taken care of.

● Save on hardware, budget, and maintenance fees.

2.5 BCCD

The BCCD [12] is abbreviated from full name Bootable Cluster CD, which is designed and implemented by Paul Gray in University of Northern Iowa. The Bootable Cluster CD (BCCD) is an established, well maintained, cluster toolkit used nationally and internationally within several levels of the academic system. The standard BCCD image is packaged in the 3", mini-CD format, easily fitting inside most wallets and purses. A typical Windows or Macintosh lab can be temporarily converted into a working GNU/Linux-based computational cluster without modification to original disk or operating system. Students can immediately use this computational cluster framework to run a variety of real scientific models conveniently located on the BCCD and downloadable into any running BCCD environment. It is reflected in the ability to take a completely unconfigured lab of networked workstations and, within a few minutes, create a fully functioning computational cluster complete with application profiling, MPI support with MPICH or LAM-MPI, PVFS2 file system support, and applications like

Gromacs and mpiBLAST at the ready. The advantages of the BCCD approach include:

- There is no required formatting of the hard disk
- There is no required operating system installation.
- The system can run completely in RAM and the CDROM image. No portion of the computer's disk is modified.
- Students have the same environment in and outside of class, as well as in a production environment.
- Educators can focus on curricular issues rather than cluster administration issues.
- Production servers with minimized administration and maintenance can be created, reducing cost, and allowing seamless migration of skills from an educational environment to a real world one.

The BCCD was created to facilitate instruction of parallel computing aspects and paradigms. Part of the difficulty instructor’s face is lack of dedicated resources to explore distributed computing aspects lack of time to preconfigured and test the supporting environment. The BCCD image addresses this problem by providing a non-destructive overlay way to run a full-fledged parallel computing environment on just about any workstation-class system.

2.6 Free Software

The first formal definition of Free Software [13] was published by FSF in February 1986. The definition is written by Richard Stallman; is still maintained today and states that software is Free Software if people who receive a copy of the software have the following four freedoms:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1).
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3).

General copyright software has limited these freedoms. In 1984, Richard Stallman launched the GNU project after becoming frustrated with the effects of the change in culture of the computer industry and users. Software development for the GNU [14] operating system began in January 1984, and the Free Software Foundation (FSF) was founded in October 1985. He introduced a free software definition and "copyleft", designed to ensure software freedom for all.

In 1992, the Linux kernel, started by Linus Torvalds the previous year, was released as free software. The combination of the kernel with components from GNU and elsewhere, such as a tool chain and shell, came to be known as Linux, and was the first complete Free Software operating system.

Chapter 3 PERFORMANCE EVALUATION ON DISKLESS PC CLUSTER SYSTEMS

First, we set up two kinds of Diskless PC Cluster systems in order to evaluate which system performance is better for us to deploy it in the computer classroom. In order to get better just data, each measured data in experiment is to ask average via ten times. Diskless system architecture is shown in Figure 3.1.

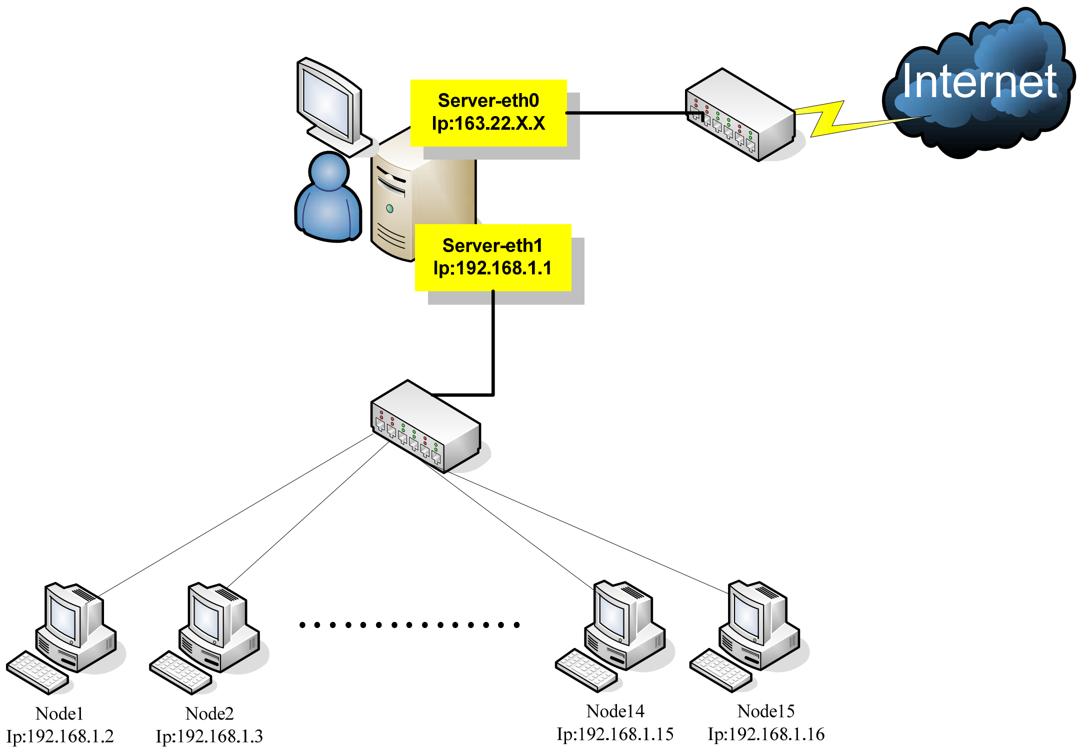


Figure 3.1: Diskless System Architecture

The Cluster consists of sixteen PC-based connected by one 24-port 100Mbps Ethernet switches with Fast Ethernet interface. There are one server node and fifteen computing nodes. The server node has one Intel Pentium 4 2.8GHz processor and 1GBytes of shared local memory. The other fifteen nodes are Intel Celeron D 3.466Ghz machines with 512MBytes of shared local memory. The hardware specification and the operating system used in our Diskless Clusters are shown as Table 3.1.

Table 3.1: Equipments and Software on Each Machine

Machines / Equipments	Diskless 1 Server	Nodes 15 Nodes
CPU	Intel P4-2.8G	Intel Celeron D 3.466Ghz
OS	BCCD: Bootable Cluster CD 2.2.1c DRBL : Fedora Core 5	
Kernel	BCCD: 2.4.25 ; DRBL: 2.6.15	
Disk	1	1
RAM	1GB	512MB
SWITCH	D-LINK DES-1026G 24-Port 10/100 M	
NIC	2 (Gigabit)	1 (Gigabit)

3.1 Performance Evaluation by Using MPI

BCCD can divide into running in CD-ROM drive (BCCD_CDROM) and RAM (BCCD_RAM) model. Matrix multiplication programs are executed by MPICH parallel software to compare performance between BCCD_CROM and BCCD_RAM. In this experiment, MPICH parallel software is defaulted by BCCD.

The matrix operation derives a resultant matrix by multiplying two input matrices, *a* and *b*, where matrix *a* is a matrix o f N rows by P columns and matrix *b* is of P rows by M columns. The resultant matrix *c* is of N rows by M columns. The serial realization of this operation is quite straightforward as listed in the following:

```
for(k=0; k<M; k++)
    for(i=0; i<N; i++){
        c[i][k]=0.0;
        for(j=0; j<P; j++)
            c[i][k]+=a[i][j]*b[j][k];
}
```

Its algorithm requires N^3 multiplications and N^3 additions, leading to a sequential time complexity of $\boldsymbol{O}(N^3)$. For matrix multiplication, the smallest sensible unit of work is the computation of one element in the result matrix. It is possible to divide the work into even smaller chunks, but any finer division would not be beneficial because of the number of processor is not enough to process, i.e., N^2 processors are needed. The matrix multiplication was ran with forking of different numbers of tasks to demonstrate the speedup.

The problem sizes of matrix multiplication were 256×256, 512×512, 1024×1024, 2048×2048 and 4096×4096 in this experiment and tested by different CPU numbers. The experimental relevant results are shown in Figures 3.2, Figure 3.3, and Figure 3.4.

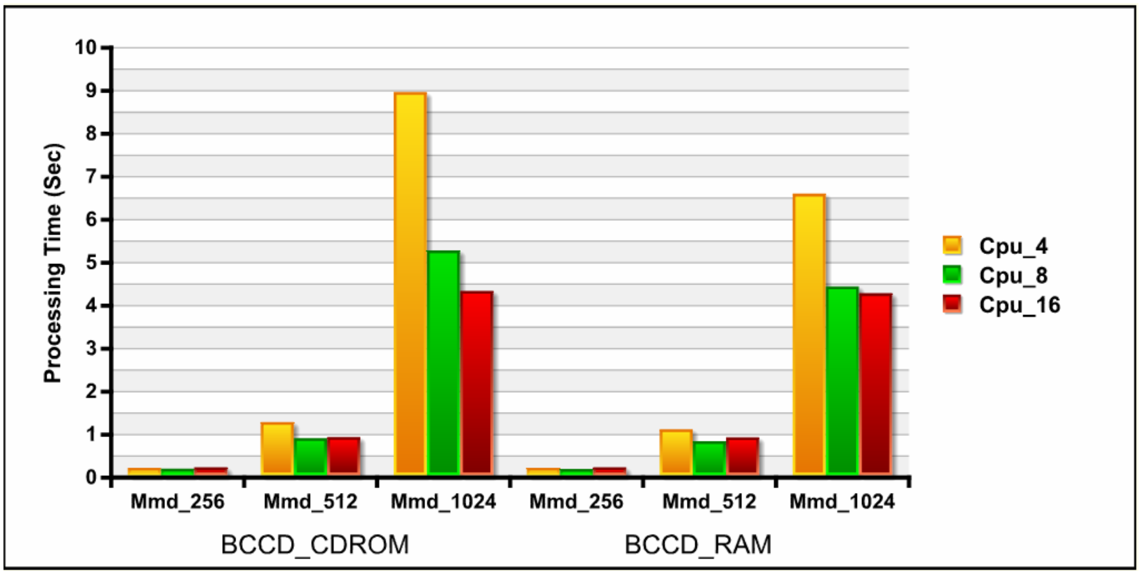


Figure 3.2: System Performance about Different Number of CPUs by Using Problem Size from 256~1024

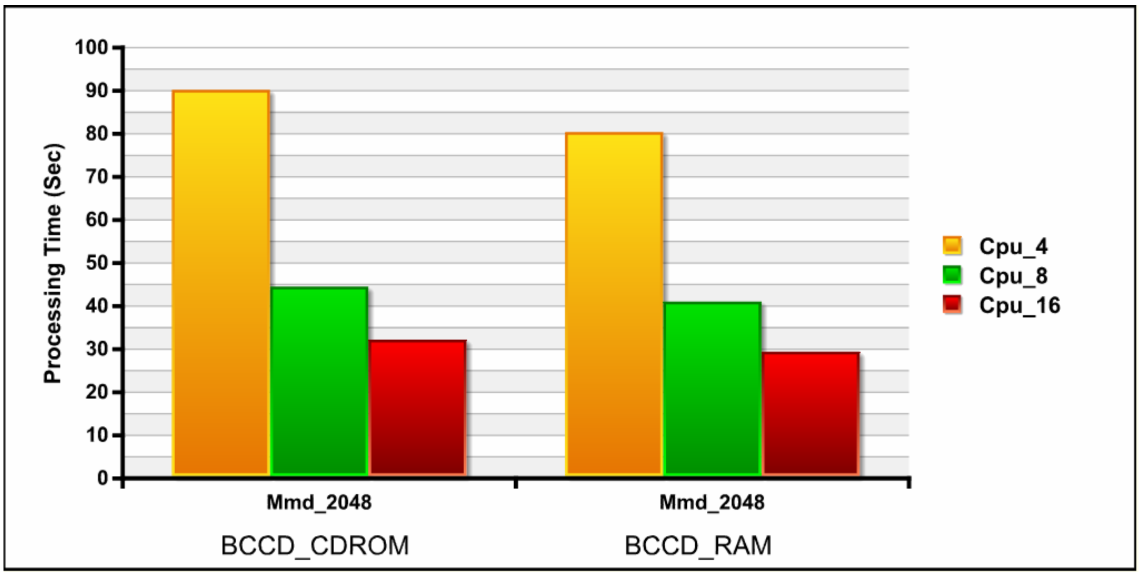


Figure 3.3: System Performance about Different Number of CPUs by Using Problem Size 2048

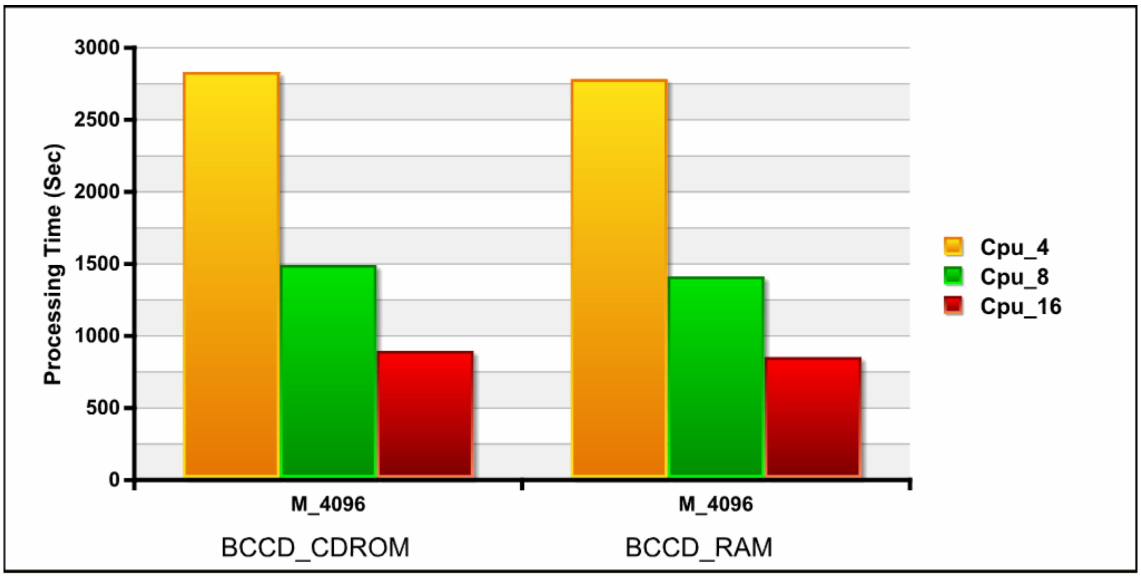


Figure 3.4: System Performance about Different Number of CPUs by Using Problem Size 4096

Above, when CPU number increases with bigger problem size, it can be found that BCCD_RAM has better performance than BCCD_CDROM. The possible reason is that BCCD_CDROM needs to use resources of I/O to do parallel computing, results in taking more time on the communication of the hardware, and it causes performance to be decrease. Because of BCCD_RAM has good performance, it is compared parallel computing performance with DRBL in computer classroom. The problem sizes of matrix multiplication were 256×256, 512×512, 1024×1024, 2048×2048 and 4096×4096 and tested by different CPU numbers.

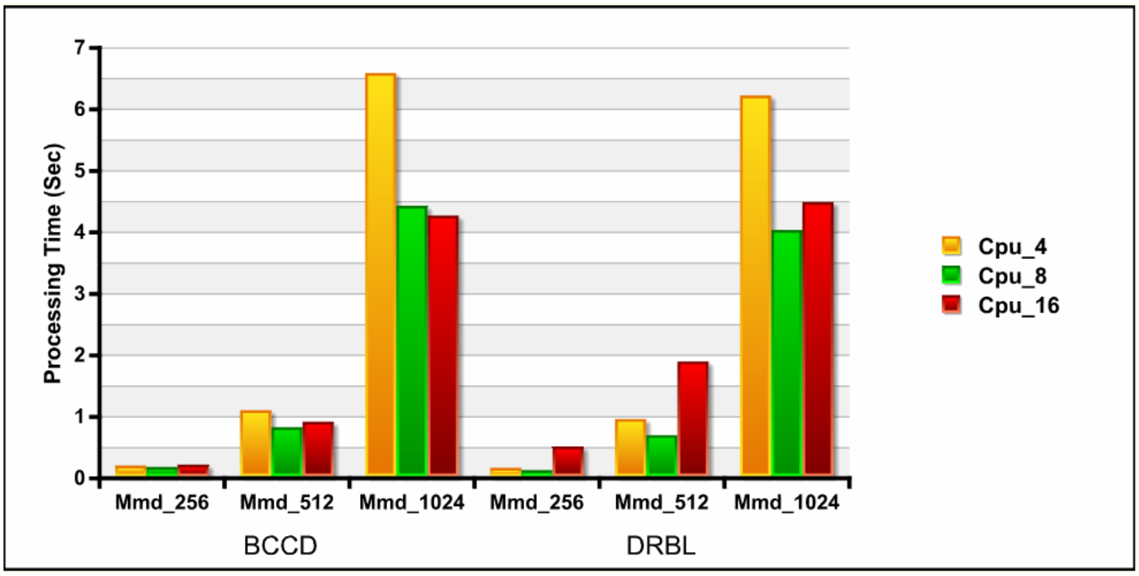


Figure 3.5: System Performance about Different Number of CPUs by Using Problem Size from 256~1024

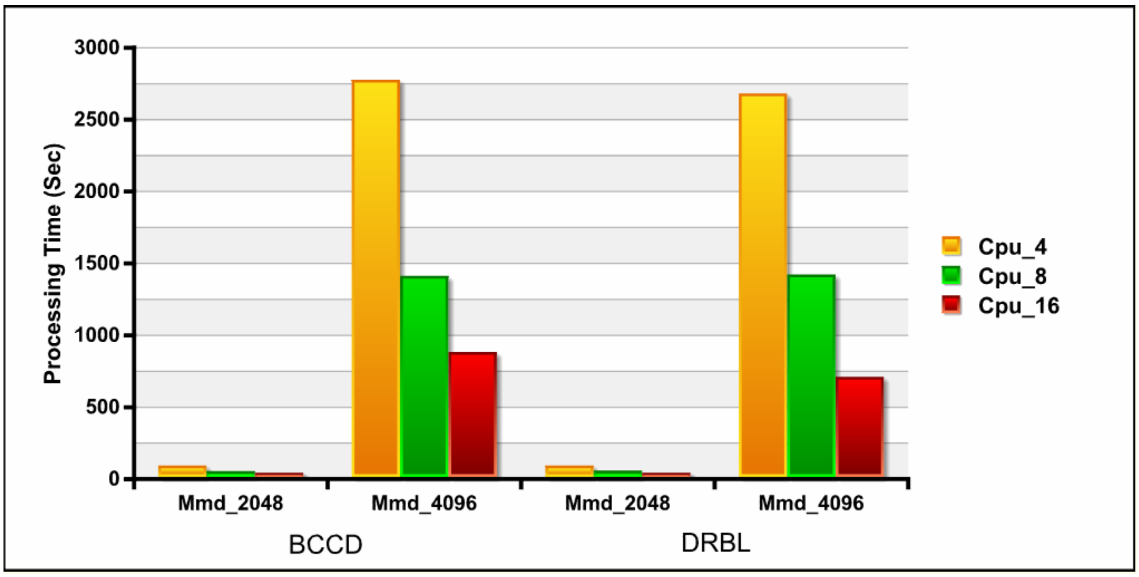


Figure 3.6: System Performance about Different Number of CPUs by Using Problem Size from 2048~4096

In Figure 3.5, it can be discovered that no matter what BCCD_RAM or DRBL performance is not good for result when uses smaller than matrix size 1024 by 16 CPUs. Especially for DRBL system, we inferred that performance result is poor because of taking too much time in mutual communication among Cluster nodes when the matrix size is smaller. In Figure 3.6, it shows that two sets of Cluster system have better performance when using more CPU numbers to compute with greater matrix size, like 2048, 4096 and DRBL is better than BCCD_RAM. Therefore, DRBL is best choice for us to deploy Cluster in computer classroom of campus.

3.2 Convenience of Cluster System Deploying

In Cluster constructing aspects, the paper compares deploying time for Cluster. Installation of DRBL needs to link to National Center of High-performance Computing through network, so the first time of DRBL takes thirty minutes generally, but installation time of BCCD takes about twenty minutes. BCCD server node can be set up in five steps, and other nodes can finish just by commanding “automode”. Hence, installation of BCCD is much more easily. However, BCCD is CD-ROM-Based Cluster, it needs to reinstall each time when computers reboot. Although DRBL takes more time to install at first, but it can be convenient for any time at next. The administration of DRBL does not need to reinstall and DRBL server is on line any time whenever you need Cluster or Free

Software Teaching. According to the above-mentioned, DRBL is more free than BCCD because DRBL can be easy switched over local OS and Cluster, besides DRBL has higher extra worth for building Free Software platform. DRBL includes startup options for clients. A DRBL system administrator can set clients to log on using the Cluster system or Free Software Teaching platform, or allow clients to boot their systems by using their local hard drives, if the local drives have an existing operating system. For administration, DRBL is very convenient and accords with the spirit of PnP whatever platform you need.

Chapter 4 USING DISKLESS SYSTEM IN THE COMPUTER CLASSROOM

4.1 Switch Method

After comparing two sets of Cluster system differences in parallel computing performance, this paper regards DRBL as the construction platform in the computer classroom. In order have not to influence the originally environment, it was essential that Windows and Cluster platform are switched over freely and it works well [15]. A corresponding sketch diagram is shown as Figure 4.1. An automatic or manual switching technology between two different platforms can be reached by Crontab software built in Linux and DRBL-WinRoll tool. Crontab is a UNIX command that creates a table or list of commands, each of which is to be executed by the operating system at a specified time. DRBL-WinRoll is a tool that enables MS-Windows client to accept commands from DRBL server. Related installation steps can be found out in DRBL website. As Figure 4.2 shows, we can obtain the most potency of idle computer resources; nodes in the computer classroom can PXE-Boot when they are available for computation in the night even on holidays, and go back to Windows when entirely necessary in the morning a school day. We can maximize computing power while minimizing costs, optimize the use of the resources that are already available, maximize resource availability and permit peaceful coexistence with previously existing OS.

As Figure 4.3 shows, the corresponding action plans are necessary, we can go into Free Software Teaching platform by commanding “remote-linux-gra” argument and go into PC Cluster platform by commanding “remote-linux-txt” argument or go into Windows platform by commanding “local”. After setting up whatever platform we choice, we just command “reboot” argument to reboot all nodes and then we will go into the platform what we choice. As Figure 4.4 shows, we combine switching procedure from Figure 4.3 to configure in Crontab file and all nodes in computer classroom will switch into different system day and night automatically.

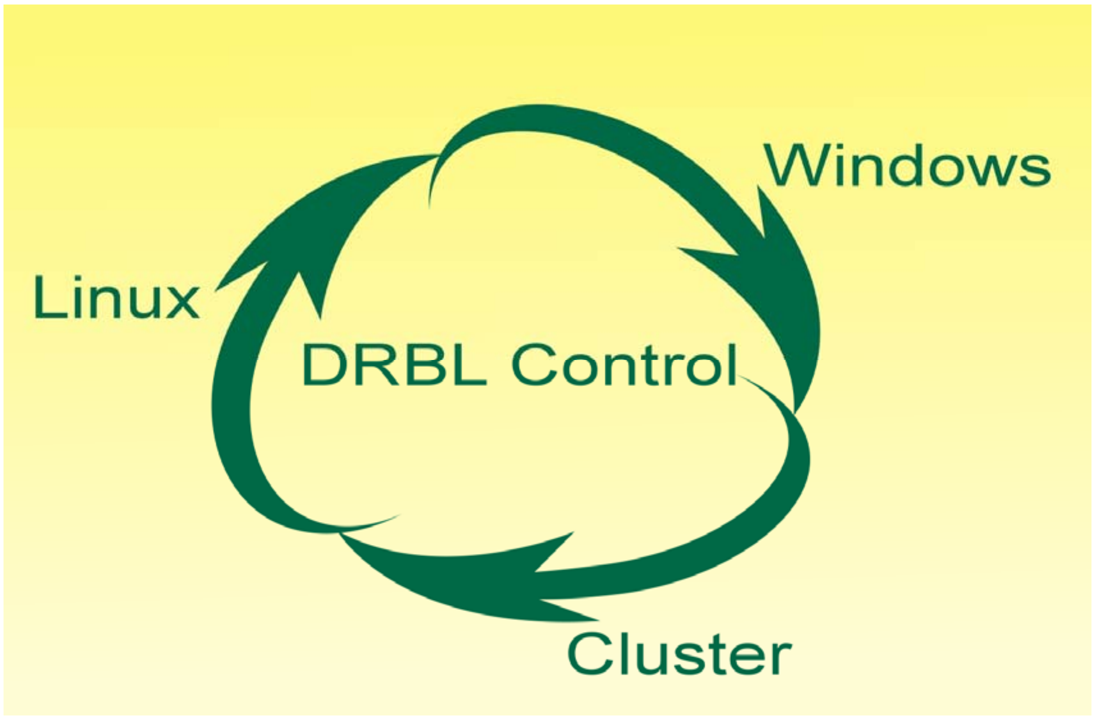


Figure 4.1: The Switching Diagram

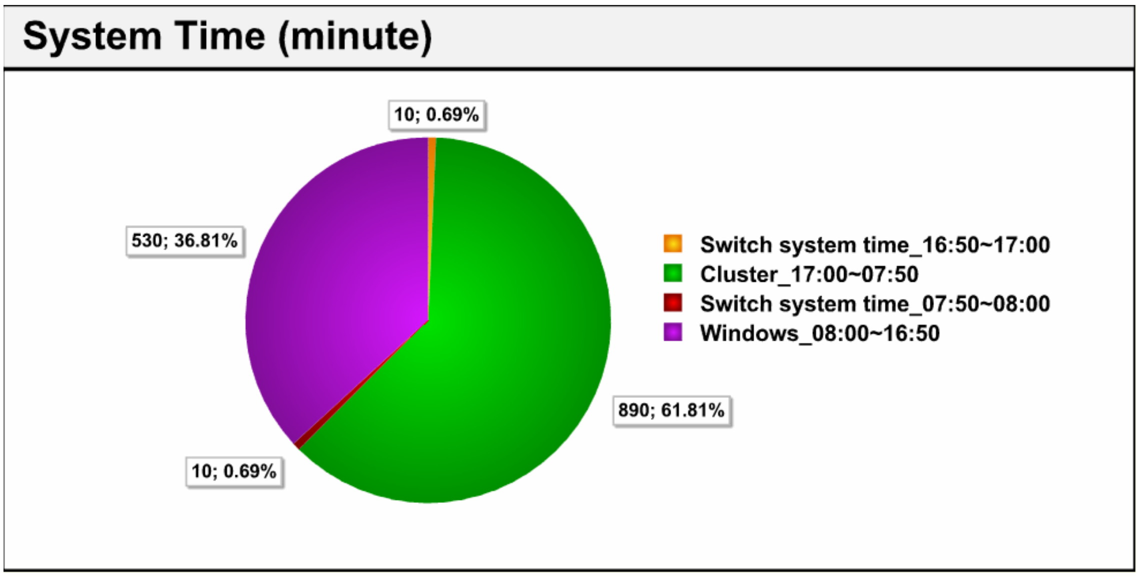


Figure 4.2: Using Time of Different System for a Whole Day

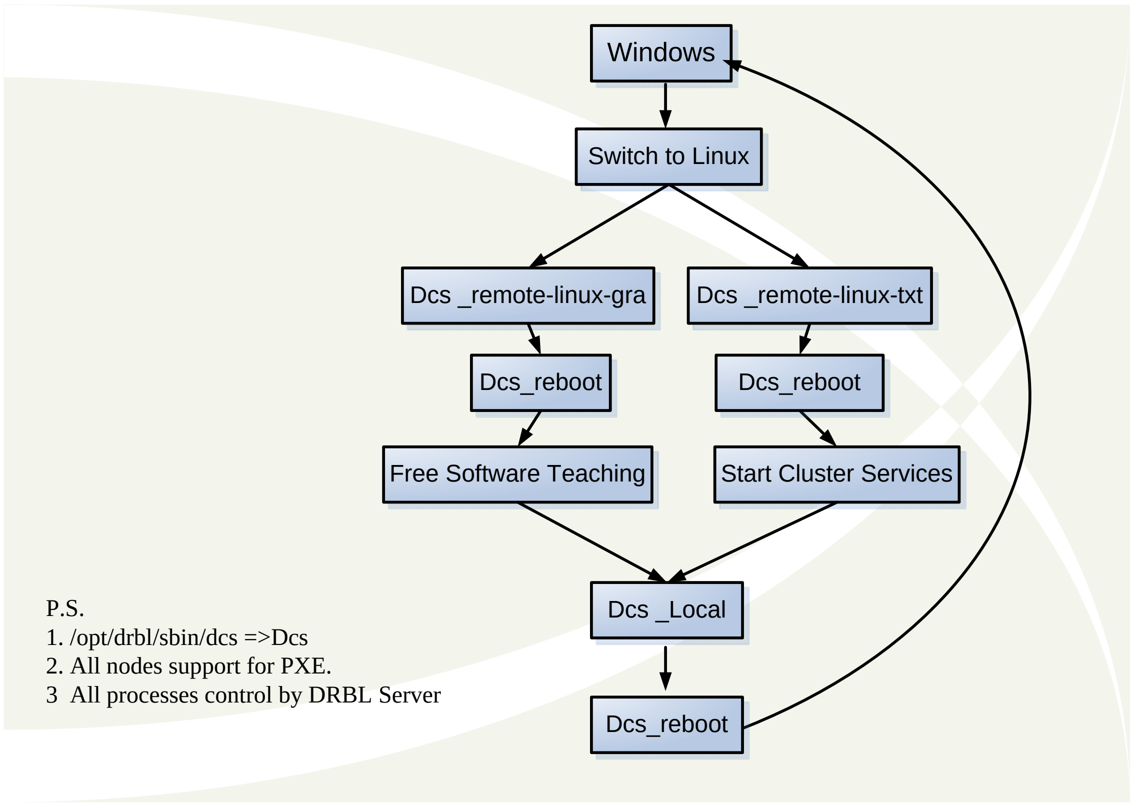
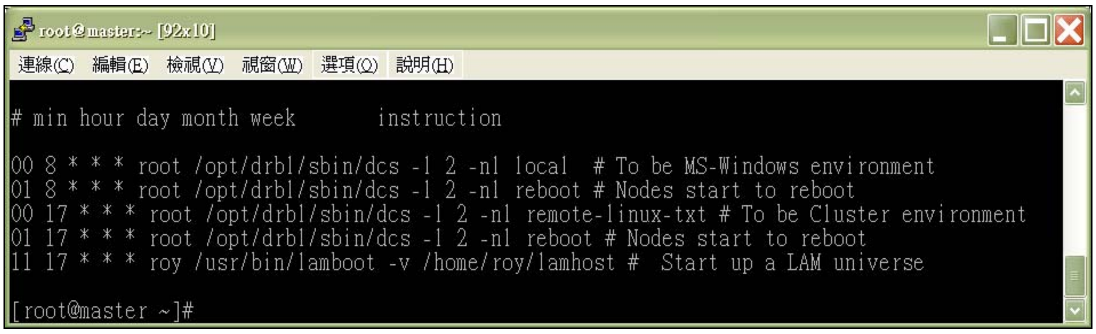


Figure 4.3: The Switching Procedure for Linux PC Cluster



```
# min hour day month week        instruction

00 8 * * * root /opt/drbl/sbin/dcs -l 2 -nl local  # To be MS-Windows environment
01 8 * * * root /opt/drbl/sbin/dcs -l 2 -nl reboot # Nodes start to reboot
00 17 * * * root /opt/drbl/sbin/dcs -l 2 -nl remote-linux-txt # To be Cluster environment
01 17 * * * root /opt/drbl/sbin/dcs -l 2 -nl reboot # Nodes start to reboot
11 17 * * * roy /usr/bin/lamboot -v /home/roy/lamhost #  Start up a LAM universe

[root@master ~]#
```

Figure 4.4: Related Configuration in Crontab File

By switch method, nodes in the computer classroom can PXE-Boot when they are available for computation (evening / holidays /vacations) and go back to Windows when strictly necessary (morning) a school day. We can maximize computing power while minimizing costs, optimize the use of the resources that are already available, maximize resource availability and permit peaceful coexistence with previously existing OS.

4.2 Matrix Multiplication

First, a Gigabit Network Interface Card is added to DRBL server in order to combine 32 nodes to compute, its relevant environmental structure chart is as Figure 4.5.

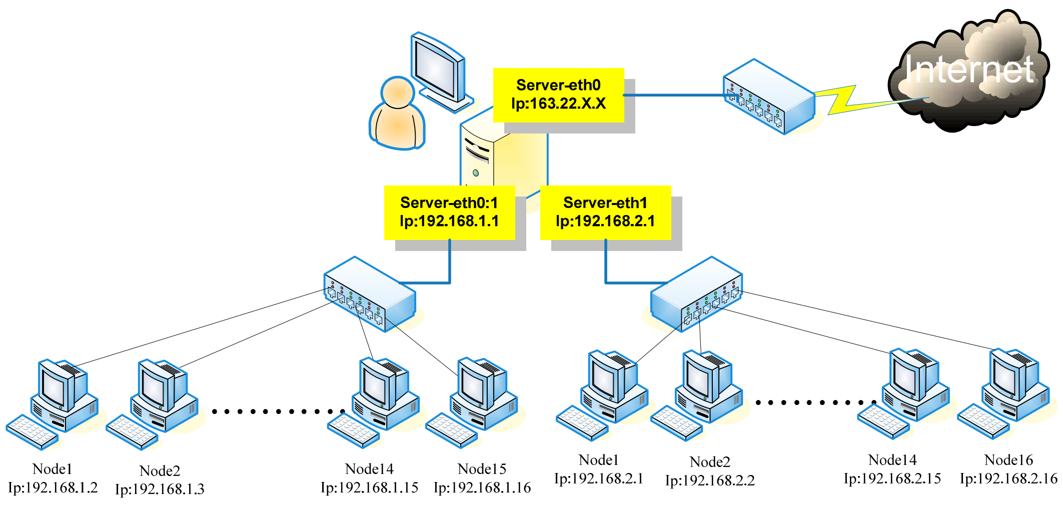


Figure 4.5: Expand the Network Bandwidth by Two NICs

DRBL server uses two NICs and two switches to expand the network bandwidth (meaning that two NICs with two private subnets such as 192.168.1.x, 192.168.2.x,). The network is divided into two sections, NetA_16 and NetB_16, and each section contains 16 desktops. Each multiple of the matrix of 256, 512, 1024, 2048 and 4096 is performed by LAM-MPI parallel computing respectively in 4, 8, 16, 32 nodes to demonstrate that the PC Clusters architecture with Diskless slave nodes is workable as well as their performance is stated.

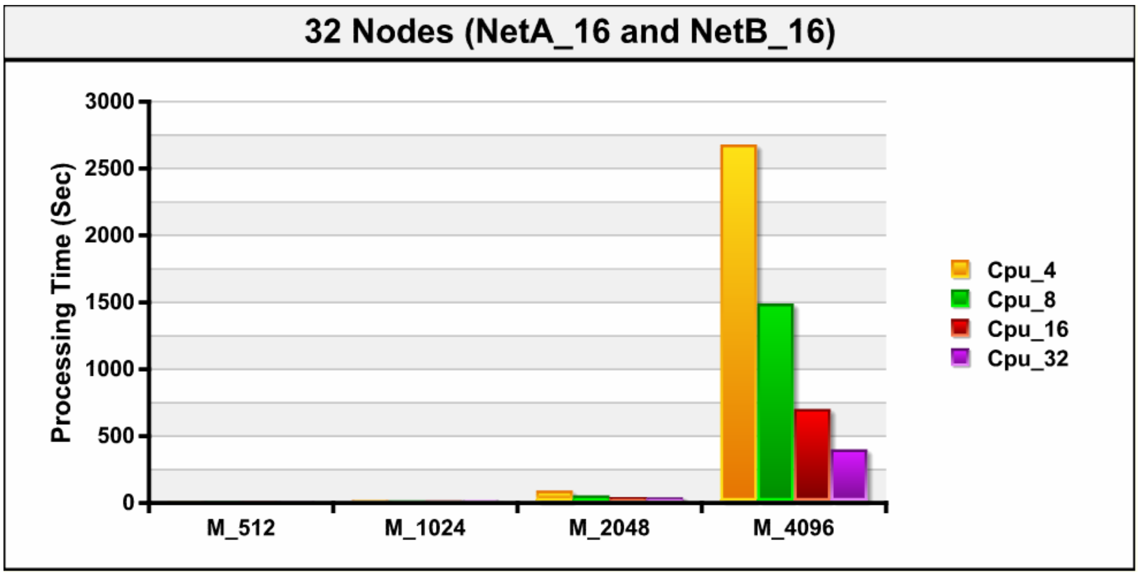


Figure 4.6: The Performance Result of Matrix Size 256~4096 in two Expanded Network by 32 nodes

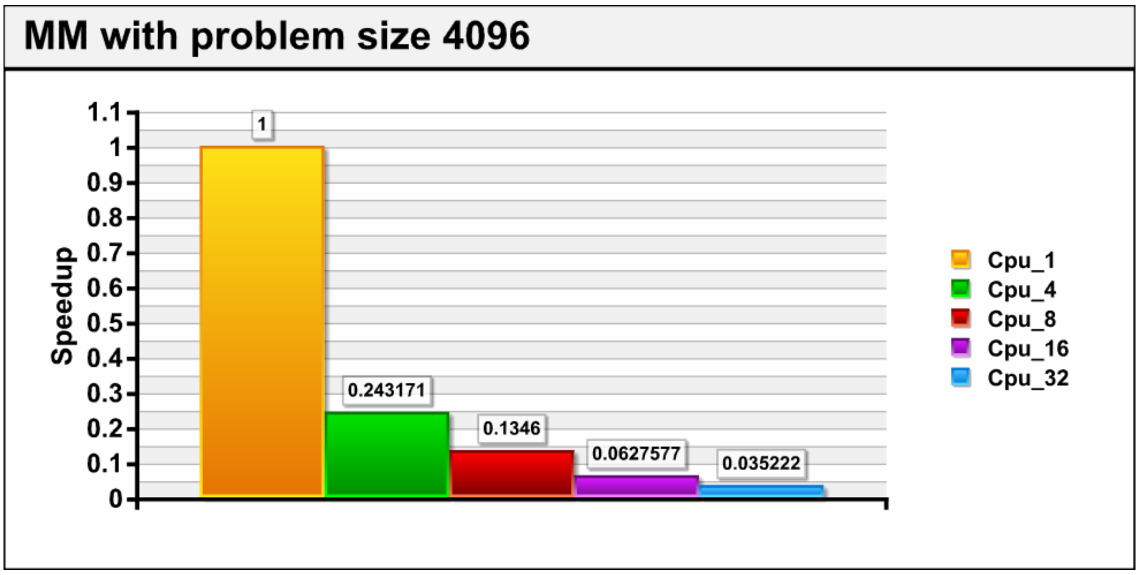


Figure 4.7: The Performance Speedup of Matrix Size 4096 in Two Expanded Network by 32 nodes

In Figure 4.6, it seems that using two NICs to expand the network bandwidth is a good idea. Especially, it can obtain close to 4 times performance when computing matrix size 4096 by 16 nodes in Figure 4.7.

4.3 Free Software Teaching

In addition, DRBL is an open source solution to managing the deployment of the GNU/Linux operating system. To understand performance of Free Software Teaching in realistic environment, evaluating booting time and software opening time are brief for test. We choice three PCs by random, and every PC is tested two times. System Ready Time includes Boot and Login Time, and Login is the way by auto because of avoiding different key in time.

TABLE 4.1: SYSTEM READY TIME FOR FREE SOFTWARE TEACHING

System Ready Time (Boot Time+ Login)				
Network Section	Node	First Time (s)	First Time (s)	First Time (s)
NetA	PC1	96	85	87
	PC2	87	84	85
	PC3	91	87	86
NetB	PC1	91	95	94
	PC2	94	92	94
	PC3	92	92	85
NetB+NetA	PC_32	128	124	124

In Table 4.1, it is discovered that booting time of every node is about less 100 seconds, if 32 nodes in the environment all start the machine; it is about less 130 seconds to finish all nodes.

TABLE 4.2: APPLICATION PROGRAM OPENING TIME

Program	Gimp(Window)			Firefox (index.html)			OOo1.2 (mouse point)		
Time(s)	First time	Second time	Third time	First time	Second time	Third time	First time	Second time	Third time
PC1	5.32	3.65	3.67	3.77	3.18	2.99	6.32	3.54	3.26
PC2	4.36	3.69	3.27	5.77	2.73	2.72	4.17	3.13	2.75
PC3	4.21	3.24	3.27	3.32	2.66	2.82	5.29	3.14	3.04

In loading program time aspect, the paper tests loading time of Free Software such as OpenOffice, Gimp, and Mozilla. The second and third time is less than first time; it means that cache in the server works well. In Tables 4.2, it can be found that three kinds of software are all being finished in 6 seconds, and it can be accepted for most people. The cache for opening application program is only helpful on opening same program of selfsame machine, and opening application program of the other PCs are not influenced by previous cache. For example, the first opening OOo1.2 program time of PC3 does not reduce because of PC2.

4.4 Performance Evaluation from Different Cluster Environment

4.4.1 Expand the Network Bandwidth

In order to know how cluster environment influences. At first we test network performance with and without expanded the network. Because this paper can’t obtain one 32-port 100Mbps Ethernet switch to compare performance with 32 nodes which use two 24-port Fast Ethernet switches to expand into two networks. Therefore, we compare 16 nodes performance in using sixteen processors with and without expanding the network. Therefore, 16 nodes are expanded the network bandwidth. The network is divided into two sections, NetA_8 and NetB_8, and each section contains eight desktops.

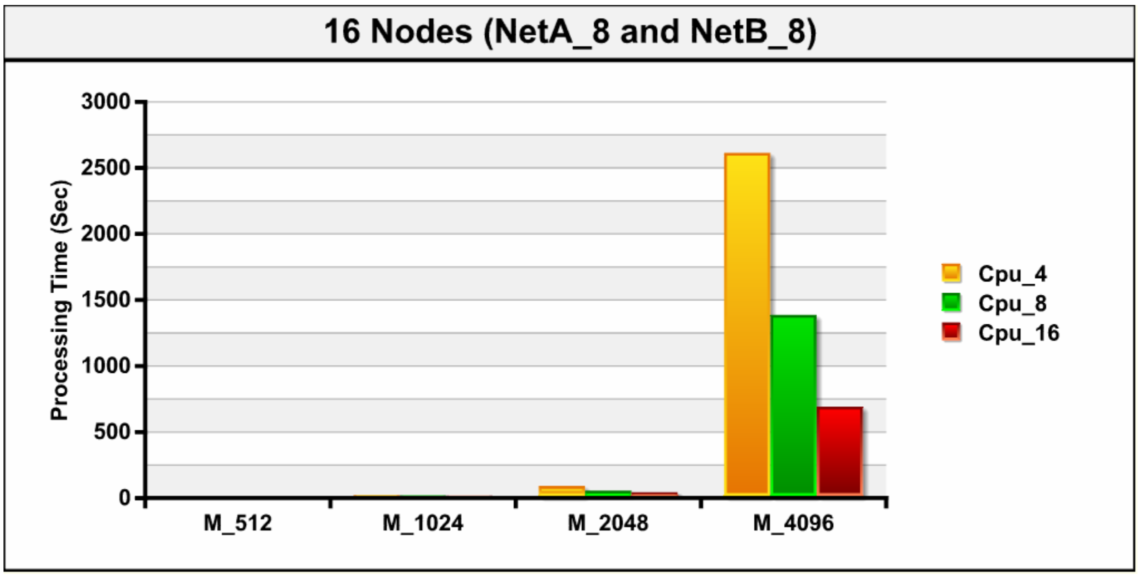


Figure 4.8: The Performance Result of Matrix Size 256~4096 in two Expanded Network by 16 Nodes

Next, we compare 16 nodes performance with and without expanding the network bandwidth by using multiple of matrix of 2048, 4096.

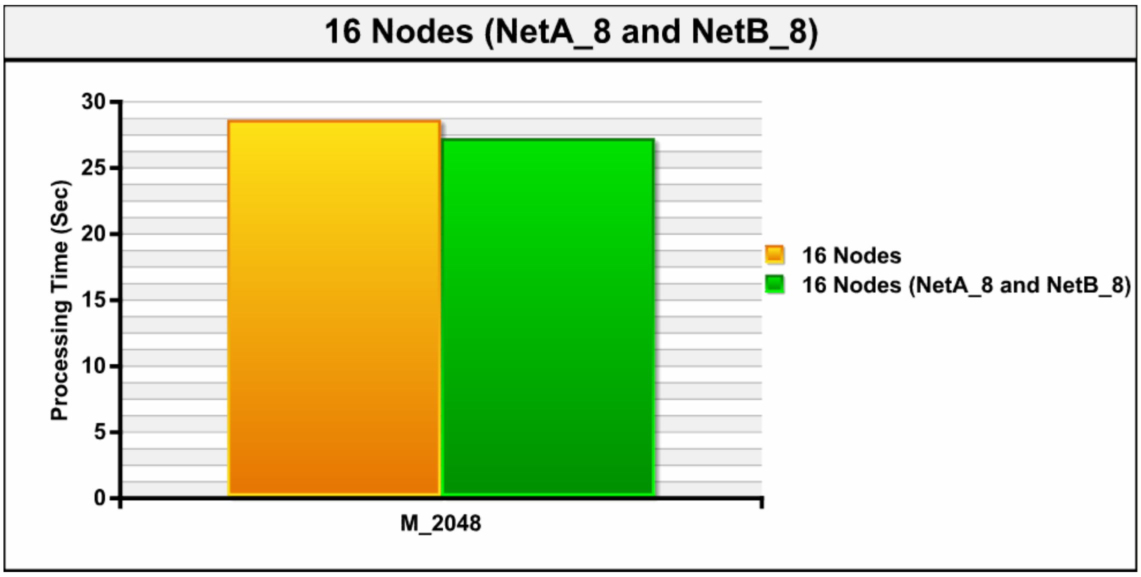


Figure 4.9: Performance of Expanding the Network Bandwidth by Computing Matrix Size 2048

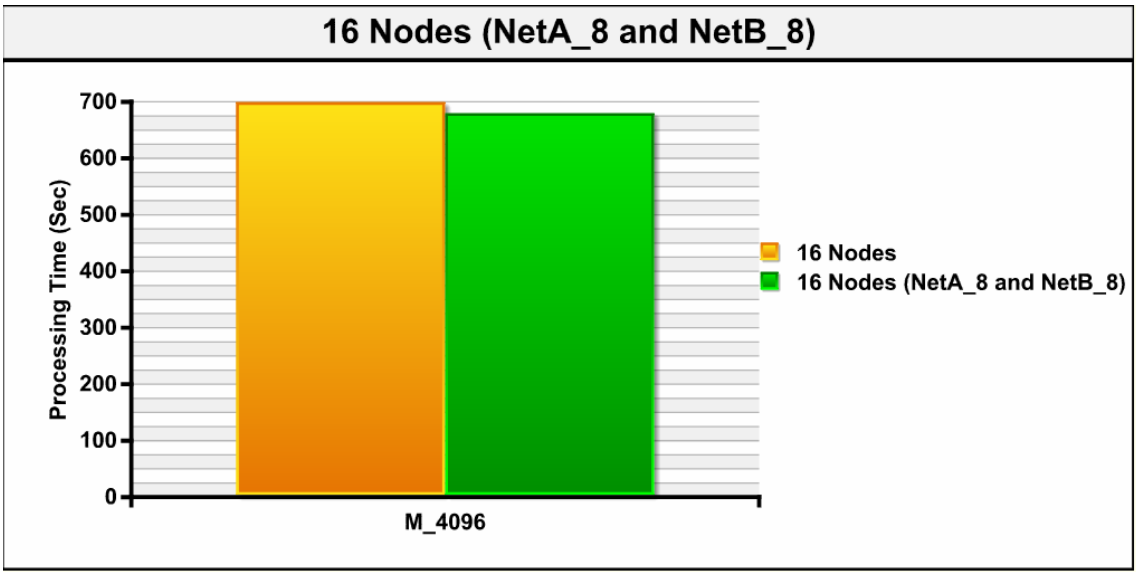


Figure 4.10: Performance of Expanding the Network Bandwidth by Computing Matrix Size 4096

As Figure 4.9 and Figure 4.10 show, the network performance with expanded in two sections is better without. The network with two 24-port Fast Ethernet switches are expanded into two independent sections in order to reduce the collision of the package as well as increase the efficiency of the network.

4.4.2 Switch Performance

A network switch is a computer-networking device that connects network segments. The network switch, packet switch (or just switch) plays an integral part in most Ethernet local area networks or LANs. The experimental environment is turned into Gigabit Ethernet, because of adding two Gigabit switches by the permission of budget. We compare performance between Gigabit and original 100Mbps Ethernet switch. The relevant performance result with 16 nodes is shown in Figure 4.11.

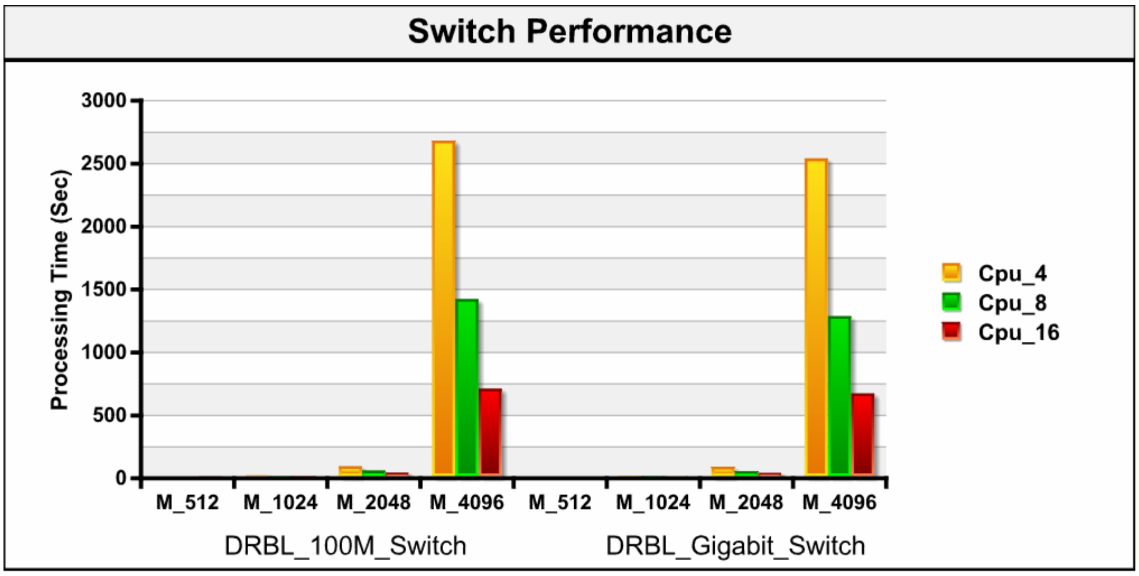


Figure 4.11: Switch Performance

As Figure 4.11, we can find out that more system performance can be obtained when Gigabit switch [16, 17] is used. All different Cluster environments performance is shown in Figure 4.12.

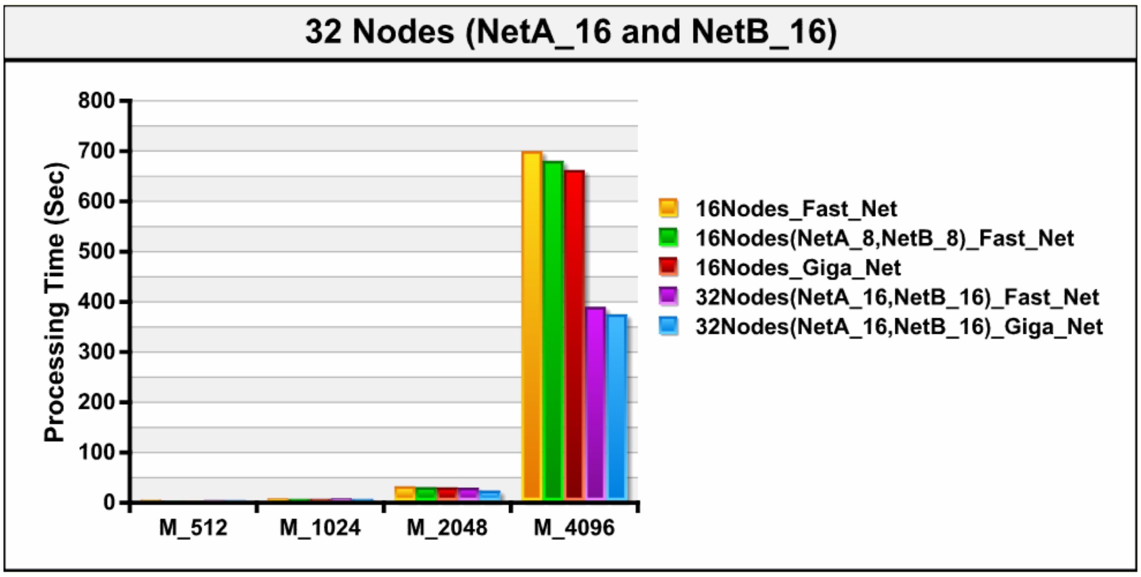


Figure 4.12: Performance of the Comprehensive Cluster Environment

In Figure 4.12, 32 nodes with Gigabit network which are expanded into two networks have highest advantage and it can be found out that Cluster performance is influenced if the network is expanded or not.

4.4.3 Swap Performance

In computer operating systems that have their main memory divided into pages, paging (sometimes called swapping) is a transfer of pages between main memory and an auxiliary store, such as hard disk drive. Paging is an important part of virtual memory implementation in most contemporary general-purpose operating systems, allowing them to use easily disk storage for data that does not fit into physical RAM. Paging is usually implemented as a task built into the kernel of the operating system. Since the swap space can be regarded as an important temporary storage to Linux system, some space of the local disk is allotted to each of the computing nodes as the swap space. Then each multiple of the matrix of 512; 1024; 2048 is performed by LAM/MPI parallel computing respectively in 4; 8; 16 computation nodes with different swap space (0 MB ; 128 MB ; 256 MB) to demonstrate that the PC Clusters architecture with Diskless slave nodes in Gigabit network environment. All different swap environments performance is shown in Figure 4.13 and Figure 4.14.

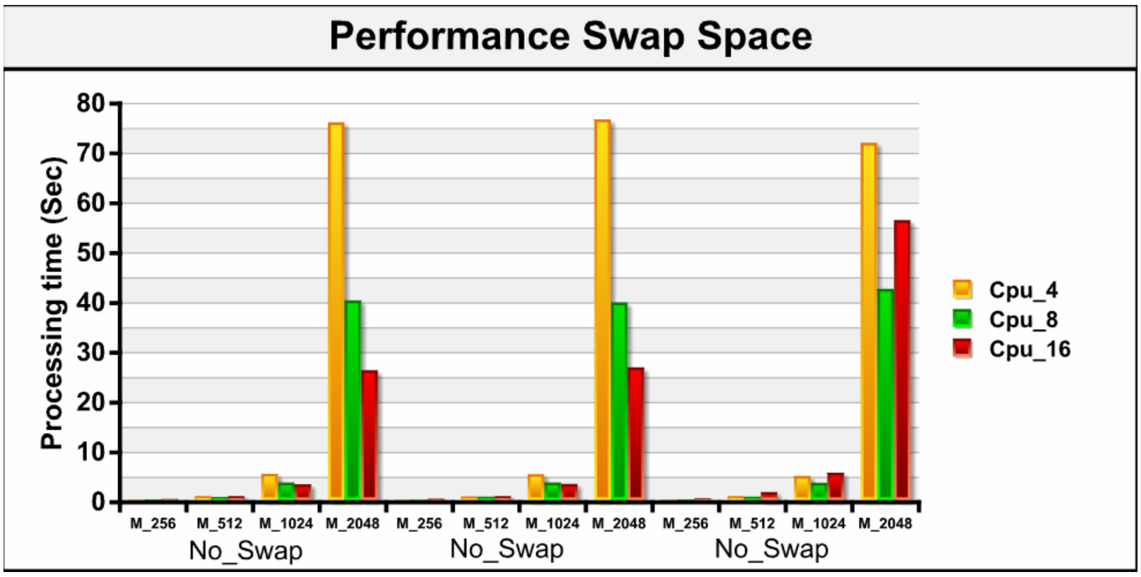


Figure 4.13: The Performance Analysis of LAM/MPI Parallel Computing in the Diskless Cluster Architecture with Different Swap Spaces

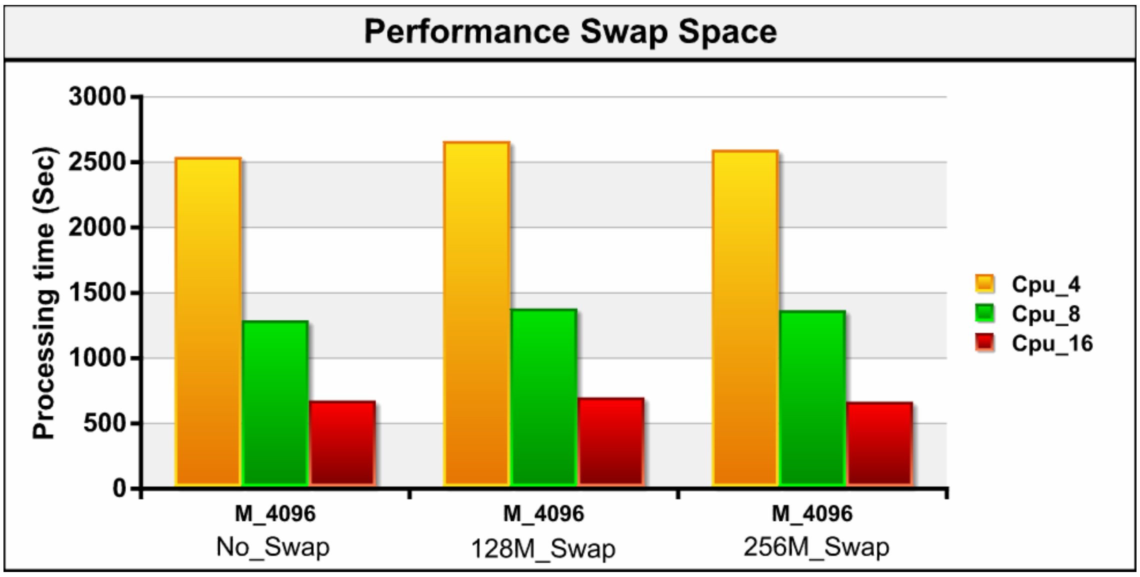


Figure 4.14: The Performance Analysis of LAM/MPI Parallel Computing with Different Swap Spaces

As Figure 4.13 and Figure 4.14 shows, it makes no difference to the system performance either with or without the local disk as the Swap space.

Chapter 5 EXPERIMENTAL RESULTS

5.1 mpiBLAST

The most popular tool for searching sequence databases is a program called BLAST (Basic Local Alignment Search Tool). BLAST compares two sequences by trying to align them, and is used to search sequences in a database. The algorithm starts by looking for exact matches, and then expands the aligned regions by allowing for mismatches. It performs pair wise comparisons of sequences, seeking regions of local similarity, rather than optimal global alignments between whole sequences. Actually, the NCBI-BLAST [18] implementation is the de facto standard for biological sequence comparison and search in computational biology.

Here are the four main executable programs in the BLAST distribution:

● [*blastall*]

Performs BLAST searches using one of five BLAST programs: *blastn*, *blastp*, *blastx*, *tblastn*, or *tblastx*

The following table summarizes the query, database sequence, and alignment types for the various BLAST commands in Table 5.1.

TABLE 5.1: BLAST COMMANDS

Program	Query sequence type	Database sequence type	Alignment sequence type
blastn	nucleotide	Nucleotide	nucleotide
blastp	protein	Protein	protein
blastx	nucleotide	Protein	protein
tblastn	protein	nucleotide	protein
tblastx	nucleotide	nucleotide	protein

● [*blastpgp*]

Performs searches in PSI-BLAST or PHI-BLAST mode. *blastpgp* performs gapped blastp searches and can be used to perform iterative searches in PSI-BLAST and PHI-BLAST mode.

● [*bl2seq*]

Performs a local alignment of two sequences. *bl2seq* allows the comparison of two known sequences using *blastp* or *blastn* programs. Most of the command-line options for *bl2seq* are similar to those for *blastall*.

● [*formatdb*]

formatdb is used to format protein or nucleotide source database. It converts a FASTA-format flat file sequence database into a BLAST database.

The bioinformatics software mpiBLAST [19] is executed to demonstrate that the PC Clusters architecture with Diskless slave is workable in real life. BLAST is a set of programs to find similarity between a query protein or DNA sequence and a sequence database. mpiBLAST is a freely available, open-source, parallel implementation of NCBI BLAST. mpiBLAST takes advantage of distributed computational resources, i.e., a cluster, through explicit MPI communication and thereby utilizes all available resources unlike standard NCBI BLAST which can only take advantage of shared-memory multi-processors (SMPs).The primary advantage to using mpiBLAST versus traditional NCBI BLAST is performance. Specifically, with the use of database fragmentation and query segmentation mpiBLAST performs a BLAST search in parallel. Database fragmentation partitions a database into multiple fragments and by distributing the fragments across many computational-resources (e.g. cluster-nodes, CPU-cores, clusters, etc.); each fragment can be searched simultaneously. In short, database fragmentation reduces execution latency because each node's database fragment resides in main memory, yielding a significant speedup due to the elimination of disk I/O. Furthermore, query segmentation increases execution throughput as a single multi-sequence query is now split into separate, independent sub queries and each sub query is executed in parallel. Here, the four bioinformatics were databases Yeast.aa, Sgt.fasta, Uniprot_sprot.fasta and Month.est_others. These databases size are about 3.24MB, 48.4MB, 151MB and 534MB. The relevant performance results by using nodes from eight to 32 are shown as Figure 19. In addition, it is found when the number of Computation Nodes increases, the execution time will reduce.

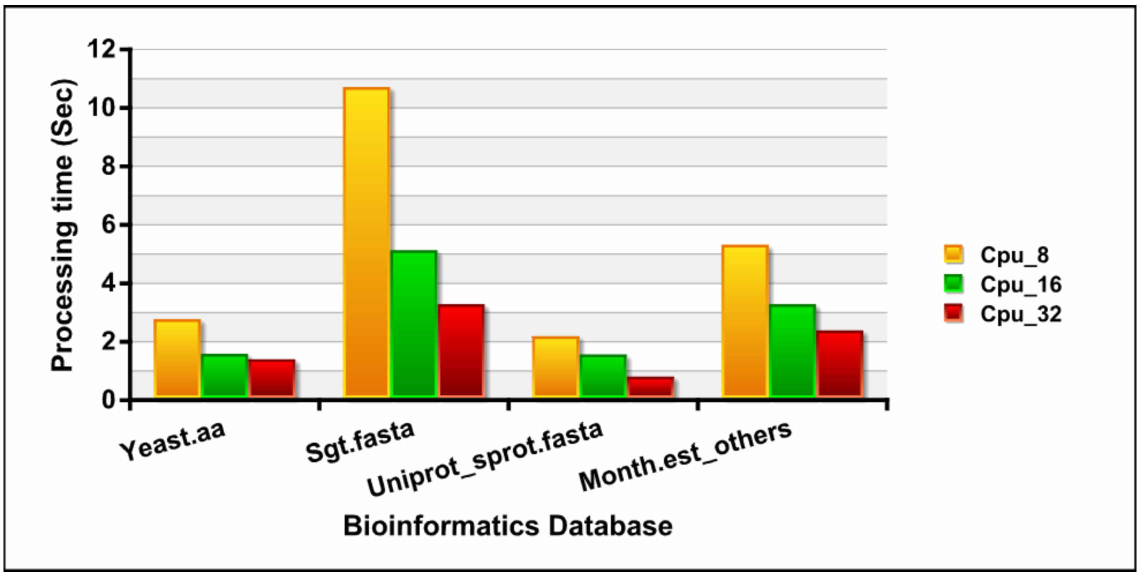


Figure 5.1: The Performance Analysis of mpiBLAST in the Diskless Cluster Architecture with Different Number of Computation Node

5.2 HPCC

The HPC Challenge (HPCC) benchmark consists at this time of seven benchmarks: HPL, STREAM, RandomAccess, PTRANS, FFTE, DGEMM and b_eff Latency/Bandwidth. HPL is the Linpack TPP benchmark. The test stresses the floating-point performance of a system. It can thus be regarded as a portable as well as freely available implementation of the High Performance Computing Linpack Benchmark.

The HPL software package requires the availability on your system of an implementation of the Message Passing Interface MPI. An implementation of either the Basic Linear Algebra Subprograms BLAS is also needed.

The best performance achievable by this software on your system depends on a large variety of factors. Nonetheless, with some restrictive assumptions on the interconnection network, the algorithm described here and its attached implementation are scalable in the sense that their parallel efficiency is maintained constant with respect to the per processor memory usage.

In order to find out the best performance of Cluster system, the largest problem size fitting in memory is what you should aim for. The amount of memory used by HPL is essentially the size of the coefficient matrix. For example, we have 32 nodes with 512 MB of memory on each; this corresponds to 16 GB total, i.e., 1000MB double precision (8 Bytes) elements. The square root of that number is about 44721. One definitely needs to leave some memory for the OS as well as for other things, so a problem size of 40000 is likely to fit. As a rule of thumb, 90% of the total amount of memory is a good guess. If the problem size you pick is too large, swapping will occur, and the performance will drop. The best values depend on the computation/communication performance ratio of your system.

This depends on the physical interconnection network you have. In other words, *P* and *Q* should be approximately equal, with *Q* slightly larger than *P*. If HPL is executed on a simple Ethernet network, there is only one wire through that all the messages are exchanged. On such a network, the performance and scalability of HPL is strongly limited and very flat process grids are likely to be the best choices: 4×8.

HPL uses the block size *NB* for the data distribution as well as for the computational granularity. From a data distribution point of view, the smallest *NB*, the better the load balance. You definitely want to stay away from very large values of *NB*. From a computation point of view, a too small value of *NB* may limit the computational performance by a large factor because almost no data reuse will occur in the highest level of the memory hierarchy.

For the TOP500 [20], we used that version of the benchmark that allows the user to scale the size of the problem and to optimize the software in order to achieve the best performance for a given machine. This performance does not reflect the overall performance of a given system, as no single number ever can. However, it reflects the performance of a dedicated system for solving a dense system of linear equations. The experimental results for HPL are shown in Table 5.2.

TABLE 5.2: EXPERIMENTAL RESULTS FOR HPCC

HPCC				GOTO	ATLAS
Ns	NBs	Ps	Qs	Gflops (Max)	
40000	72	4	8	7.06E+01	1.73E+01
40000	108	4	8	7.15E+01	5.84E+01
40000	112	4	8	**7.25E+01**	5.95E+01
40000	128	4	8	7.21E+01	5.84E+01
40000	148	4	8	7.00E+01	**6.90E+01**
40000	196	4	8	7.23E+01	6.27E+01

The benchmark, HPL is used to demonstrate the performance of our parallel test bed by using LAM/MPI. The experimental results show that our Cluster system can obtain 72.5 Gflops for HPL and the GOTO of BLAS library is better than ATLAS.

The runtime measurements described above only give an absolute measurement of the computation time used by the program. It is also interesting to know how efficient these computations actually are. The efficiency is the ratio between the actual performance and the theoretical performance of a system.

The floating-point performance of a computer is expressed in FLOPS - floating-point operations per second. The theoretical performance of a superscalar computer is calculated as follows [21]:

$$R_{peak} = n_{cores} \cdot n_{FPU} \cdot f \, ,$$

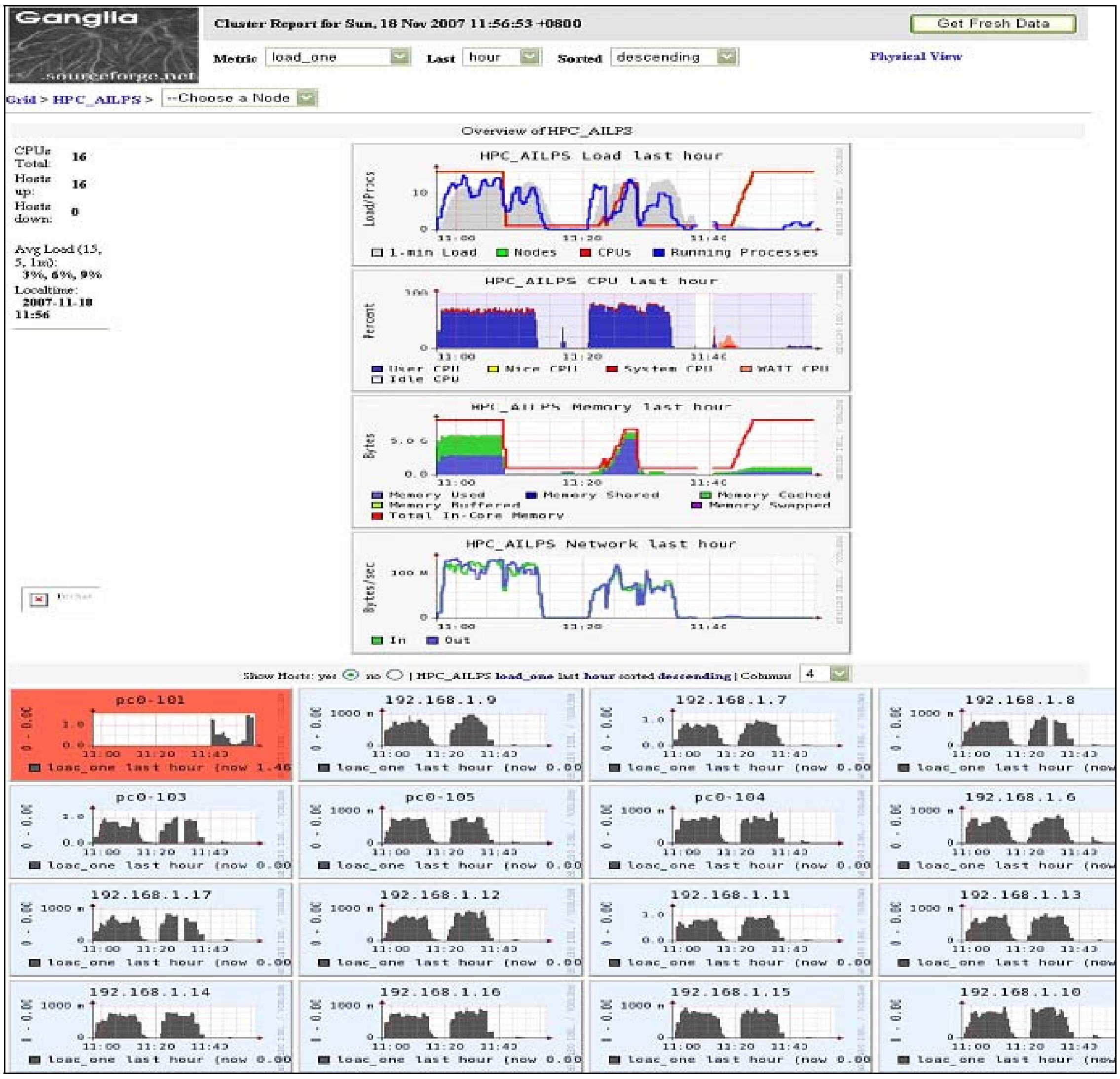
Where n_{cores} is the number of computing cores of the computer, n_{FPU} is the number of floating-point units per core, and f is the clock frequency. For a Celeron D 360 (one cores, one FPUs per core, clock frequency 3.466 GHz), and now we have 32 Nodes, R_{peak} results to $32 \cdot 1 \cdot 3.466 \cdot 10^9$ FLOPS = 110.912 GFLOPS (Gflops). We compare with NCHC Formosa Cluster and other two Cluster systems. It is discovered that HPL performance efficiency is 65% in our experiment, and it performs not bad.

TABLE 5.3: HPL PERFORMANCE ANALYSIS

HPL Performance	YCES Cluster	Formosa Cluster	HP Superdome	IBM P690
R_{max} (Gflops)	72.5	1166	642.9	736.6
R_{peak} (Gflops)	110.912	1920	768	1331.2
Efficiency (%)	65	61	84	55
CPU Number	32	340	128	256
Clock frequency (Gflops/CPU)	3.466	3.43	5	2.88
Choice of solving dense linear system and BLAS libs	HPL & GOTO lib	HPL & GOTO lib	CXML	ESSL lib

5.3 Ganglia

As the PC cluster becomes a popular low-cost high-performance computing platform, monitoring the status of a Beowulf-style cluster platform can be a daunting task for any system administrator, especially if the cluster system consists of more than a dozen computing nodes [22, 23, 24]. From management's view, the state of each Computation Node in Clusters is monitored by "Ganglia" [25], an Open Source software. "Ganglia" is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. It allows the user to view remotely live or historical statistics (such as CPU load averages or network utilization) for all machines that are being monitored. It is based on a hierarchical design targeted at federations of clusters. It relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on over 500 clusters around the world. It has been used to link clusters across university campuses and around the world. The ganglia system comprises two unique daemons, a PHP-based web front-end, Ganglia Monitoring Daemon (gmond) and Ganglia Meta Daemon (gmetad). All nodes state is shown in Figure 5.2.



Cluster Report for Sun, 18 Nov 2007 11:56:53 +0800

Metric load_one Last hour Sorted descending

Grid > HPC_AILPS > --Choose a Node

Overview of HPC_AILPS

CPUs Total: 16
Hosts up: 16
Hosts down: 0

Avg Load (15, 5, 1m): 3%, 6%, 9%
Localtime: 2007-11-18 11:56

Show Hosts: yes [✓] no | HPC_AILPS load_one last hour sorted descending | Columns 4

Figure 5.2: The 16 Nodes State by Ganglia

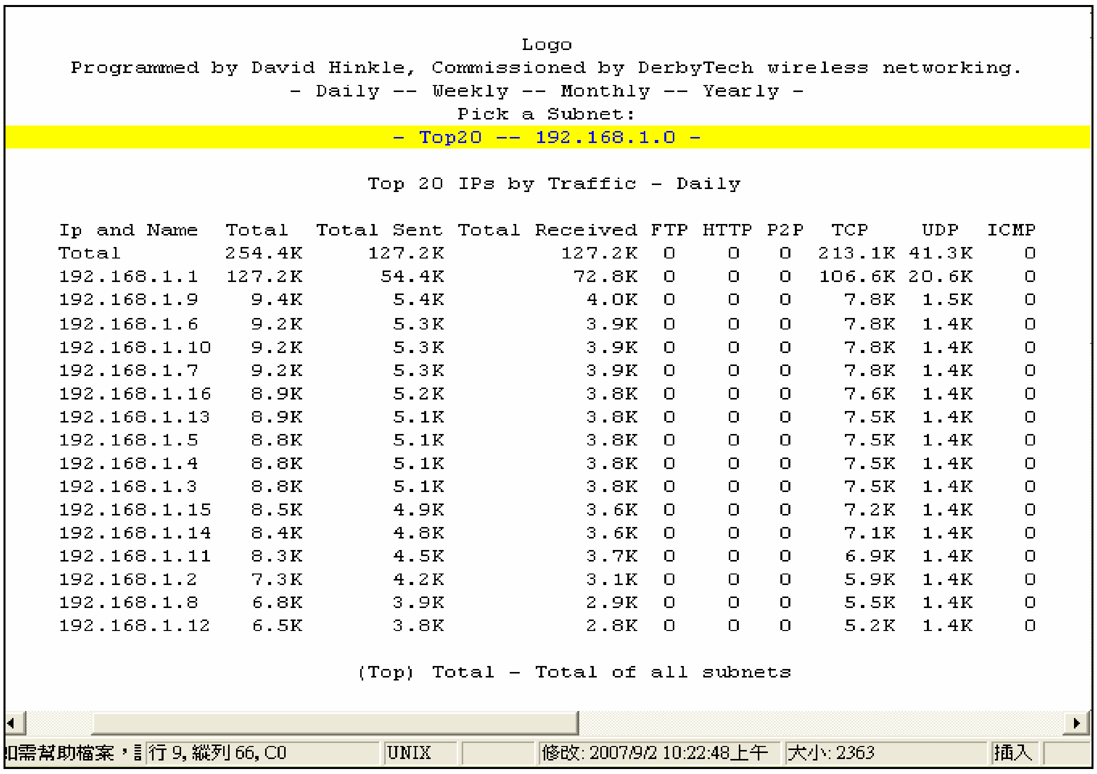
Figure 5.2 shows that the machine's image will turn into different status when the idle rate of CPU utilization is changed. We can find out that "pc0-101" node image is red in Figure 5.2. Because of PC host name "pc0-101" is our Cluster server, many services run on it result in heavy loading, therefore, the utilization is often over 100%. Table 5.4 is the default Ganglia node image legend.

TABLE 5.4: GANGLIA NODE IMAGE LEGEND

Node Image	Meaning
Red	Over 100% Utilization. Utilization: (1 min load) / (number of CPUs) * 100%.
Orange	75-100%
Yellow	50-74%
Green	25-49%
Blue	0-24%
Crossbones	The node is dead. We consider a node dead when the reporting node has not heard from it in 60 sec.

5.4 BandwidthD

Besides, “BandwidthD” [26] is an Open Source software to track usage of TCP/IP network subnets and builds html files with graphs to display utilization. Charts are built by individual IPs, and by default display utilization over 2 day, 8 day, 40 day, and 400 day periods. Furthermore, each Ip address's utilization can be logged out at intervals of 3.3 minutes, 10 minutes, 1 hour or 12 hours in cdf format, or to a backend database server. HTTP, TCP, UDP, ICMP, VPN, and P2P traffic are color-coded. It first is as a standalone application that produces static html and png output every 200 seconds. The second is as a sensor that transmits its data to a backend database, which is then reported on by dynamic php pages. Our experiment BandwidthD result is shown as Figure 5.3.



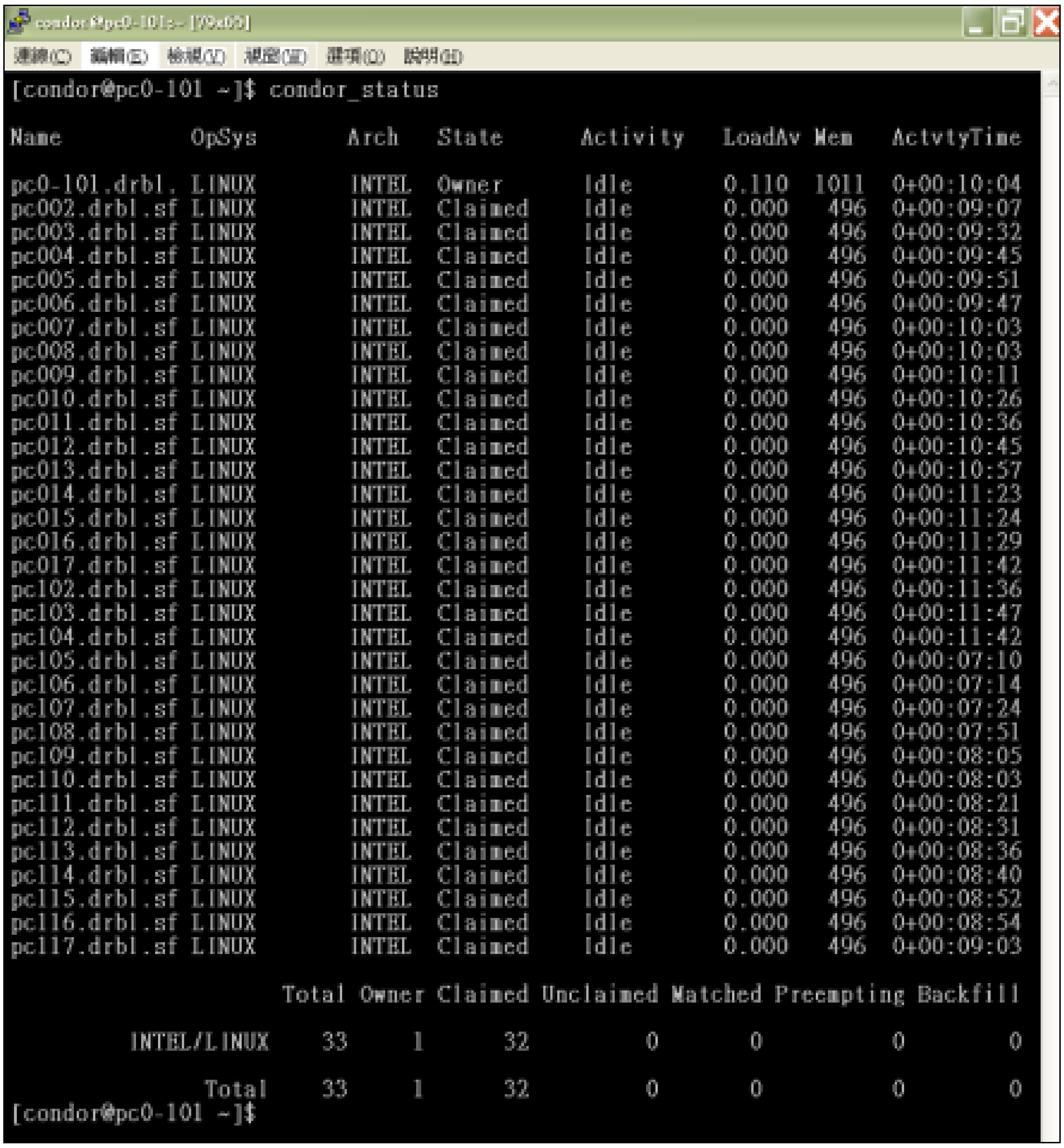
```
                                  Logo
     Programmed by David Hinkle, Commissioned by DerbyTech wireless networking.
                     - Daily -- Weekly -- Monthly -- Yearly -
                                Pick a Subnet:
                            - Top20 -- 192.168.1.0 -

                         Top 20 IPs by Traffic - Daily

    Ip and Name   Total  Total Sent Total Received FTP HTTP P2P  TCP    UDP  ICMP
    Total         254.4K     127.2K         127.2K  0    0   0  213.1K 41.3K   0
    192.168.1.1   127.2K      54.4K          72.8K  0    0   0  106.6K 20.6K   0
    192.168.1.9     9.4K       5.4K           4.0K  0    0   0    7.8K  1.5K   0
    192.168.1.6     9.2K       5.3K           3.9K  0    0   0    7.8K  1.4K   0
    192.168.1.10    9.2K       5.3K           3.9K  0    0   0    7.8K  1.4K   0
    192.168.1.7     9.2K       5.3K           3.9K  0    0   0    7.8K  1.4K   0
    192.168.1.16    8.9K       5.2K           3.8K  0    0   0    7.6K  1.4K   0
    192.168.1.13    8.9K       5.1K           3.8K  0    0   0    7.5K  1.4K   0
    192.168.1.5     8.8K       5.1K           3.8K  0    0   0    7.5K  1.4K   0
    192.168.1.4     8.8K       5.1K           3.8K  0    0   0    7.5K  1.4K   0
    192.168.1.3     8.8K       5.1K           3.8K  0    0   0    7.5K  1.4K   0
    192.168.1.15    8.5K       4.9K           3.6K  0    0   0    7.2K  1.4K   0
    192.168.1.14    8.4K       4.8K           3.6K  0    0   0    7.1K  1.4K   0
    192.168.1.11    8.3K       4.5K           3.7K  0    0   0    6.9K  1.4K   0
    192.168.1.2     7.3K       4.2K           3.1K  0    0   0    5.9K  1.4K   0
    192.168.1.8     6.8K       3.9K           2.9K  0    0   0    5.5K  1.4K   0
    192.168.1.12    6.5K       3.8K           2.8K  0    0   0    5.2K  1.4K   0

                        (Top) Total - Total of all subnets
```

```
如需幫助檔案，請 行 9, 縱列 66, C0   UNIX   修改: 2007/9/2 10:22:48上午   大小: 2363   插入
```

Figure 5.3: The BandwidthD of DRBL

5.5 Performance Result with Condor

Condor [27,28], a software system developed at the University of Wisconsin that promises to expand the center's computing capabilities easily by obtaining computer cycles on idle nodes. Condor is a software system that creates a High-Throughput Computing (HTC) environment. It effectively utilizes the computing power of workstations that communicate over a network. Condor can manage a dedicated cluster of workstations. Its power comes from the ability to harness effectively non-dedicated, preexisting resources under distributed ownership. No changes are made to the computing environments on individual nodes. Rather, the software organizes the nodes into Clusters, called pools, or collections of Clusters called flocks, that can exchange resources. From its perch in these computing networks, Condor hunts for idle workstations. When it spots one, it swoops in to run a job. When the owner resumes computing, Condor whisks the job to another machine. A user submits the job to Condor. Condor finds an available machine on the network and begins running the job on that machine. Condor has the capability to detect that a machine running a Condor job is no longer available. Use the command "condor_status" to check the machine status of the condor pool. This information includes the name of the machine, current state, and even loads average. As Figure 5.4 shows, 32 nodes are waiting for submitting job.



```
[condor@pc0-101 ~]$ condor_status

Name          OpSys       Arch   State     Activity LoadAv Mem  ActvtyTime

pc0-101.drbl. LINUX       INTEL  Owner     Idle     0.110  1011  0+00:10:04
pc002.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:07
pc003.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:32
pc004.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:45
pc005.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:51
pc006.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:47
pc007.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:03
pc008.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:03
pc009.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:11
pc010.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:26
pc011.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:36
pc012.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:45
pc013.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:10:57
pc014.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:23
pc015.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:24
pc016.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:29
pc017.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:42
pc102.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:36
pc103.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:47
pc104.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:11:42
pc105.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:07:10
pc106.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:07:14
pc107.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:07:24
pc108.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:07:51
pc109.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:05
pc110.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:03
pc111.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:21
pc112.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:31
pc113.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:36
pc114.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:40
pc115.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:52
pc116.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:08:54
pc117.drbl.sf LINUX       INTEL  Claimed   Idle     0.000   496  0+00:09:03

                     Total Owner Claimed Unclaimed Matched Preempting Backfill

         INTEL/LINUX    33     1      32         0       0          0        0

               Total    33     1      32         0       0          0        0
[condor@pc0-101 ~]$
```

Figure 5.4: The Condor State for 32 Nodes

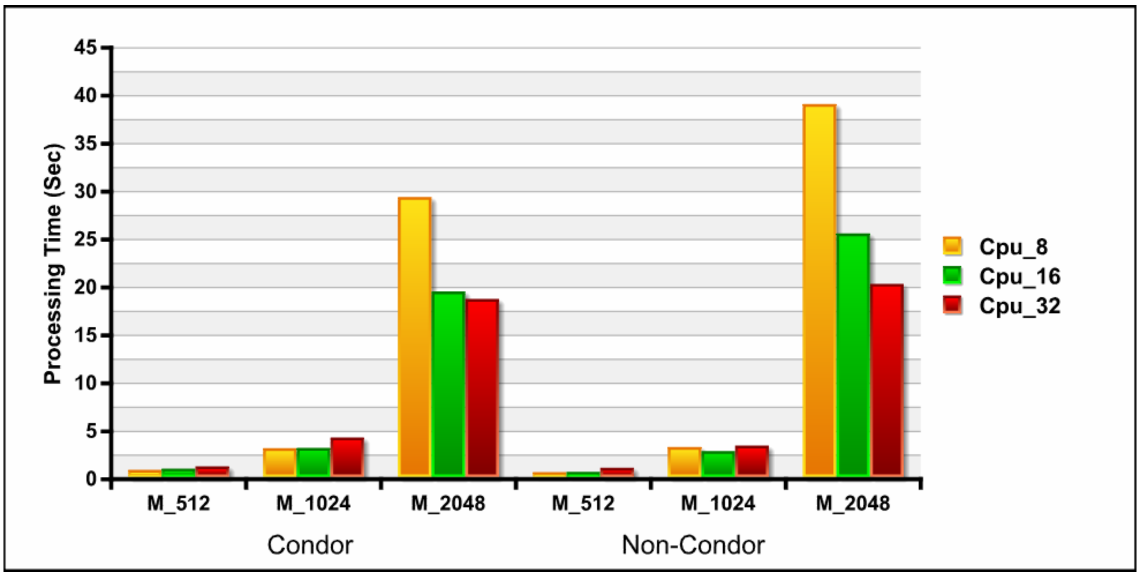


Figure 5.5: Condor Performance Analysis

We submitted jobs for Matrix Multiplication problem sizes from 512×512, 1024×1024 and 2048×2048, which are tested by different CPU numbers. As Figure 5.5 shows, the parallel computing performance running in Condor is better than Non_Condor.

Chapter 6 CONCLUSIONS AND FUTURE WORK

After actual comparing two Cluster software in this experiment, we find out that DRBL is suit for deploying in computer classroom of campus. This thesis has created switching mechanism in different systems automatically or manually with DRBL. The mechanism can freely switch different systems among Windows, Free Software Teaching and PC Cluster and it is similar to PnP.

In Free Software Teaching aspect, no matter booting machine or loading software time, DRBL is satisfied in most teaching environment after actual test. The single booting machine time takes about 100 seconds and even if 32 PCs all in computer classroom never take over 130 seconds, I think that DRBL performs well which it can be acceptable for most educators and students. The three kinds of Free Software such as OpenOffice, Gimp, Mozilla, their loading time are also performed well that loading time is fewer 6 seconds. DRBL is a NFS and NIS server that offers authentication and boot services to the clients. Once clients are authenticated, the DRBL server loads the packages onto the clients and from thereon the clients use their own hardware for further processing of the loaded applications. A typical application of DRBL would be in a lab or a classroom for teaching Linux. All the computers in a classroom can be identically configured simply by installing and pre-configuring the DRBL environment on a single central server. Existing operating systems on the individual computers will not be affected. Compared with the traditional method of classroom management, this software greatly increases management effectiveness, reduces hardware requirements, and lowers licensing costs normally associated with individual computer setup.

In Cluster parallel computing aspect, we use 32 nodes in the computer classroom to achieve the greatest performance and it is helpful to achieve higher performance by using multiple NICs to expand the network bandwidth. Our objective is to deploy a Cluster-based architecture with Diskless nodes and it is not only maximizes the utilization of idle resources but also enhances and glorifies the Cluster application. Client machines can be workstations at daytime, and become Cluster computing nodes at night. These nodes can be very quickly integrated into a cluster without any alteration of the main OS stored on their disks. The bioinformatics program, mpiBLAST, is executed smoothly in the Cluster architecture as well. Finally, HPCC is used to demonstrate Cluster performance and we obtain 72.5 Gflops for HPL with GOTO of BLAS library. It is discovered that HPL performance efficiency is 65% in our experiment, and it performs not bad. Therefore, we suggest administrators interested in a cost-efficient, powerful, and easy-to-manage thin client server should consider Diskless Remote Boot in Linux (DRBL), a server-based open source application that lets organizations deploy GNU/Linux across many clients. In the future, we hope that Globus Toolkit is added to deploy across area Grid Computing environment, and the Grid portal [29] is shared for whose demand, which need High Performance Computing on the internet.

BIBLIOGRAPHY

[1] Rajkumar Buyya, High Performance Cluster Computing: System and Architectures, Vol. 1, Prentice Hall PTR, NJ, 1999.T. L.

[2] Toshihiro Ikeda, Akira Hara, Takumi Ichimura, Tetsuyuki, Takahama, “Multi-agent Cluster System for Optimal Performance in Heterogeneous Computer Environments,” R. Khosla et al. (Eds.): KES 2005, LNAI 3681, pp. 932-937, 2005.

[3] Diskless, http://en.wikipedia.org/wiki/Diskless_workstation

[4] BCCD, http://bccd.cs.uni.edu/

[5] DRBL, http://drbl.sourceforge.net/

[6] Thomas Sterling, Donald J. Becker, John Salmon and Daniel F. Savarese, “How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters,” second Printing, MIT Press, Cambridge, Massachusetts, USA, 1999.

[7] Khoa N. Nguyen and Thuy T. Le, “Evaluation and Comparison Performance of Various MPI Implementations on an OSCAR Linux Cluster,” Proceedings of International Conference on Information Technology: Coding and Computing (ITCC03), Las Vegas, NV, U.S.A. (April 2003)

[8] MPICH, http://www-unix.mcs.anl.gov/mpi/mpich1/

[9] LAM/MPI Parallel Computing, http://www.lam-mpi.org

[10] PXE, http://www.pxe.ca/index.html

[11] NFS, http://nfs.sourceforge.net/

[12] Sarah M. Diesburg and Paul A. Gray, “High Performance Computing Environments Without the Fuss: The Bootable Cluster CD,” Parallel and Distributed Processing Symposium, Proceedings. 19th IEEE International Volume , Issue , 4-8 April 2005 Page(s): 8 pp. -

[13] Free Software, http://www.gnu.org/philosophy/free-sw.html

[14] GNU, http://www.gnu.org/

[15] Chao-Tung Yang, Yi-Cheng Hu, and Ping-I Chen, “Design and Implementation the High-Performance PC Cluster Architecture with Diskless Slave Nodes in the Computer Classroom,” Proceedings of the Taiwan Academic Network Conference (TANet 2005), 2005 Page(s): 14 pp. -

[16] Chao-Tung Yang, Ping-I Chen and Ya-Ling Chen, “Performance Evaluations of SLIM and DRBL Diskless PC Clusters on Fedora Core 3,” Proceedings of the sixth

IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2005), pp. 479-482, December 5-8, 2005.

[17] Hong Ong , Paul A. Farrell, “Performance Comparison of LAM/MPI, MPICH, and MVICH on a Linux Cluster connected by a Gigabit Ethernet Network,” Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, October 10-14, 2000.

[18] NCBI BLAST, http://www.ncbi.nlm.nih.gov/blast/Blast.cgi

[19] mpiBLAST, http://www.mpiblast.org/

[20] The TOP500 Supercomputers list, http://www.top500.org.

[21] Tobias Wittwer, “An Introduction to Parallel Programming,” 2006.

[22] Chao-Tung Yang, Chun-Sheng Liao and Kuan-Ching Li, “On Construction a Large Computing Farm Using Multiple Linux PC Clusters,” PDCAT 2004, LNCS, Springer, vol. 3320, pp.856-859, Dec. 2004.

[23] Chao-Tung Yang, Chi-Chu Hung and Chia-Cheng Soong, “Parallel Computing on Low-Cost PC-Based SMPs Clusters,” Proceedings of International Conference on PDCAT 2001, Taipei, Taiwan, pp 149-156, July 2001.

[24] Chao-Tung Yang and Chun-Sheng Liao, “On Construction and Performance Evaluation of Cluster of Linux PC Clusters Environments,” Proceedings of the 6th IEEE International Symposium on CCGrid Workshops, Singapore, May 16-19, 2006, pp. 53.

[25] Ganglia, http://ganglia.sourceforge.net/

[26] BandwidthD, http://bandwidthd.sourceforge.net/

[27] Condor High Throughput Computing, http://www.cs.wisc.edu/condor/

[28] Rajkumar Buyya, High Performance Cluster Computing: Programming and Applications, Vol. 2, Prentice Hall PTR, NJ, 1999.

[29] Juan Touriño, Member, IEEE, María J. Martín, Jacobo Tarrío, and Manuel Arenaz, “A Grid Portal for an Undergraduate Parallel Programming Course,” IEEE Transactions on Education, 48(3):391-399, August 2005.