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

多重個人電腦叢集系統之建置與評估

Construction and Performance Evaluation of Cluster of Linux PC Clusters Environment

研 究 生：廖 峻 陞

中華民國九十四年六月

摘要

在個人電腦叢集於平行處理領域的發展中，我們可以使用網路位置轉換 NAT (Network Address Translation) 機制來建置叢集系統叢集 (Cluster of Clusters, CoCs)，這類的應用也稱為多重叢集 (Multi-cluster)。應用此技術，我們可以將不同的個人電腦叢集系統透過高速的網路設備將其整合成一個單一且大型的平行處理系統並且可以得到令人滿意的效能。另一方面，在目前 IPv4 網路環境之下，真實 IP 位置 (Public IP Address) 嚴重的不足，而使用多重叢集的架構也能為我們省下多餘的網路位置。在這篇論文裡面，會包含以下三個部份：第一部份是利用 NAT 技術在東海大學的內部網路內將四套分散在不同地點的個人電腦叢集來建置我們的多重叢集。第二部份是為了方便使用多重叢集我們建置了一套在叢集系統上面運行的監控系統，並且可以透過手持的行動裝置來進行機器的監控和控制。第三部份則是對這個平台進行效能的分析，我們會利用不同的 MPI 平行程式的實例來進行多重叢集系統對於不同平行程式架構下的分析。

關鍵字： 個人電腦叢集、多重叢集、叢集計算、訊息傳遞、網路位置轉換

Abstract

Multi-cluster with NAT (Network Address Translation) is a kind of building cluster of clusters (CoCs) in parallel processing, we can easily combine two or more PC clusters which had setting on difference place to form a big one parallel system to reach the acceptable performance and resolving the issue of insufficient public IP address. In this thesis, it includes three parts: First we construct a CoCs with four PC clusters which connected with NAT on difference place in LAN environment. Second, we build a monitor system in our system in order to control and observe this system in a convenient way; this monitor system also has a client with mobile availability for users without computers. Third, we evaluate our system different MPI parallel programs for the analysis of different parallel models.

Keywords：PC Clusters, Cluster Computing, Cluster of Clusters, Message Passing, NAT

Acknowledgements

I would like to thank all of these people who have supported and helped me through the completion of this thesis and all of my working for the business of our lab. In particular, I would like to thank my advisor, Dr. Chao-Tung Yang, who introduced me to parallel computing and give me the broad support and guidance. I would also like to Professor Kuan-Ching Li, Professor Wen-Chung Chiang, Professor Chao-Chin Wu, and Professor Cheng-Chung Chu for the valuable comments and advice given while serving on my reading committee.

There are many people whom I would like to thank, my classmate Shih-Chieh Yen give me the discussion for the original idea of my thesis. The members of HPC lab includes Chuan-Lin Lai, Po-Chi Shih, and Yi-Chun Hsiung, and group-mate Ping-Yi Chen and all the other members of HPC lab, they have give me a lot of support of completing my thesis. I would like also thanks the people who support and encourage me for this research. Thanks for your help and considerations.

Finally, I would like to thank my family, my girl friend, and all of my friends. Because of your unconditional support I could made this thesis complete.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Extraordinary technological improvements over the past few years in areas such as microprocessors, memory, networks, and software have made it possible to assemble 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 [3, 4, 9, 22, 26].

Inexpensive systems such as Beowulf clusters have become increasingly popular in both commercial and academic sectors of bioinformatics community. Clusters typically consist of a master node that distributes the bioinformatics application amongst the other nodes (slave nodes).

In our laboratory, we have several Linux PC clusters. Their configurations usually consisted of one master node and three, seven or more slave nodes with dual-processor SMP for reaching maximal performance. Unfortunately, the public IP addresses are not enough for setting used on all slave nodes. It means that the virtual IP address is used for slave nodes in a PC cluster. If we want to use a cluster with 64 CPUs or more for some experimentation like gene sequence analysis or large computing job. Currently, our hardware setting is not suitable for allocating all computing resources. Therefore, to find a solution that can combine more Linux PC clusters for parallel computing is our main motivation in this thesis.

Multi-cluster with NAT (Network Address Translation) is a kind of building cluster of clusters (CoCs) [2] in parallel processing, we can easily combine two or more

PC clusters which had setting on difference place to form a big one parallel system to reach the acceptable performance and resolving the issue of insufficient public IP address. Performance is one of the important concerns of both cluster users and system developers. However, there is no clear and widely accepted CoCs performance definition. In this thesis, we built a multi-cluster with four PC clusters which connected with NAT on difference place in LAN environment. We use many parallel applications like the matrix multiplication application to measure the MPI message passing library in our testing environment and finding out the performance issue and some characteristics of building it.

Monitoring the status of a Beowulf-style cluster can be a daunting task for any system administrator, especially if the cluster consists of more than a dozen nodes. While Linux is extremely stable, hardware problems can cause nodes to crash or become inaccessible, and chasing down problem nodes in a 500-node cluster is painful. Managing and monitoring a cluster is both a tedious and challenge task, since each node is designed as a stand alone system rather than a part of a parallel architecture. Beowulf systems will need a richer set of software tools to improve usability and re-configurability.

We present our effort to resolve this problem by developing a PC cluster monitoring system. This system also provides web service and application to monitor for large scale clusters or grid environment.

1.2 Contributions

We conduct our multi-cluster system that with NAT mechanism that shows in our testing parallel application it has good performance and in the additional evaluation we could find that the machine availability could reflect the performance on multi-cluster environment. We also conduct our effort on the monitor system of cluster, we use the master-slave architecture to implement our monitor and make a

special client for mobile equipments based on Java. We can easily monitor in everywhere even if we don’t have a generic computer, we could also receive the cluster status on our mobile devices.

1.3 Thesis Organization

The rest of this thesis is organized as follows. In chapter 2, we review the Cluster Computing, introduction of Linux OS, the two major implementations of Message Passing Interface, and related work. In chapter 3, we introduce our multi-cluster system and discuss details of technologies used in the construction of such platform and present our cluster monitor system, while our experimental results are discussed in chapter 4. Finally, in chapter 5, we list some conclusion and feature work.

Chapter 2

Background

2.1 Beowulf Cluster

A Beowulf cluster uses multi-computer architecture, as depicted in Figure 2.1. 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 node 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 [1]. 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 [26]. Once on the master node, users can prepare and compile their

parallel applications, and also 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) [14, 15] and Parallel Virtual Machine (PVM) [20], 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 Linux

Linux is a robust, free and reliable POSIX compliant operating system. Several companies have built businesses from packaging Linux software into organized distributions; RedHat is an example of such a company. Linux provides the features typically found in standard UNIX such as multi-user access, preemptive multi-tasking, demand-paged virtual memory and SMP support. In addition to the Linux kernel, a large amount of applications, system software and tools are also freely available. This makes Linux the preferred operating system for clusters.

The idea of the Linux cluster is to maximize the performance-cost ratio of computing by using low-cost commodity components and free-source Linux and GNU software to assemble a parallel and distributed computing system. Software support includes the standard Linux/GNU environment, including compilers, debuggers, editors, and standard numerical libraries. Coordination and communication among the

processing nodes is a key requirement of parallel-processing clusters. In order to accommodate this coordination, developers have created software to carry out the coordination and hardware to send and receive the coordinating messages. Messaging architectures such as MPI and PVM, allow the programmer to ensure that control and data messages take place as needed during operation.

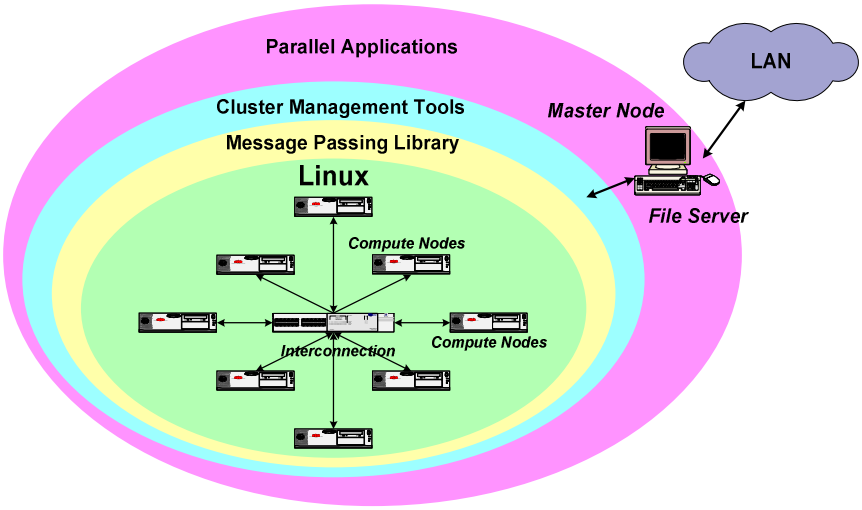

Figure 2.1: Logic view of a PC Cluster

2.3 Message Passing Interface

There are many MPI implementations for parallel processing [5, 18, 24]; most of the used libraries are MPICH [16] and LAM [10].

2.3.1 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.

2.3.2 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

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.

2.4 Multi-Cluster Environments

The components of personal-computer is cheaper today and the ratio of cost to price is even more valuable when we use the Bewulf cluster for parallel processing and applications. When we purchase groups of PC to build a Beowulf cluster in batches. The problem is that we can't easily import the devices such as the main processor which was appeared years ago because of the Moore's Law. We could not get higher performance because we can't easily combine these heterogeneous computational resources.

We have several ways to combine our heterogenous cluster to build a new computational resource to get a better performance in recent years [13, 19]. This is a domain of Grid Computing. The purpose of the Grid Computing is that we can get any type of resources to resolve many of demanding problems [6, 7]. In other words, reseachers has many contributions about to use the improvements or the extentions of MPI implementation, for combination of getting higher computational power.

The overall solutions of coupling serveral clusters into a distributed computing enviroment with MPI can be divided into four ways [17].

2.4.1 Grid Computing Environments

If all of our cluster nodes have public IP addresses there are no major problem. In such situation we can build a large computing farm using the Grid-enabled MPI Implementation [2, 23]. All nodes of cluster platform need install the Grid Computing middleware such as GLOBUS [6]. In GLOBUS the inter-cluster communication in Grid Computing enviroment is using the GLOBUS-IO mechanism, intra-cluster communication is still using the suppoted MPI libraries. MPICH-G2 is the implementaion of the widely distributed MPICH-library for Grid-envrionment that

support GLOBUS. It implements a device for MPICH and giving the user the possibilities to rely on many GLOUS mechanisms like authentication and authorizations.

2.4.2 Network Address Translation

If we don't have many public IP address used for the cluster nodes, the previous solution is not directly available. The Grid-enabled solution needs the support of middleware support. In the Grid Computing environment all IP address must be public IP addresses. The Beowulf type clusters are usually connected together with the private IP addresses. The packets with private addresses are normally not routed through the public IP address area. In a Local Area Network (LAN) enviroment it's still might be possibly to grouping the clusters, because the packets in LAN will be traveling over the switch or router by the specially solution, network address translation (NAT) [21].

NAT is a mechanism to map IP addresses. It usually couples with the packet filting, mangling, and IP masquerade, the mechanism map the private IP address by a public IP address. This mechanism need a node that has the availability to connect between the public and private network. This node privides IP forwarding of outgoing packets in the private network and keeping track all incoming packets to correct node in the private network.

2.4.3 Gateways

Another solution of coupling the clusters is the one of gateway node. The gateway is the node that connect the public IP network and persisted in the MPI commnunication world in the MPI mechanisms. It provides the machenisms that joining the MPI jobs and managing the message passing over the network. Every nodes in the MPICH-G2 has the availability of being a gateway node.

PACX-MPI [8], a MPI implementation of coupling the clusters to a distributed

high-performace computing system. Unlike the MPICH-G2 using the Grid Computing environment, PACX-MPI has its two level hierarchy of network structure, one for inter-cluster communication that perform on a TCP/IP network with high latancy network, while the other is providing the low-latency, high bandwidth network for intra-cluster communication with vendor-supported MPI library. To reach this goal, it uses a daemon-based system to handle these two types of communication level.

2.4.4 Proxies

The proxies are the application service that provide the private and public IP address communication, which is unlike NAT proxies are in the user space solution. The proxy server would have sufficient knowledge to handle the inside and outside communication. Stampi is the example MPI implementation with proxy mechanism. An usually used solution to this solution is the SOCKS v5 server [11].

2.4.5 Comparisons Among These Technologies

In general, Beowulf-Cluster usually using private ip address to inter-connected together and in fact, we have many single clusters that in seperated locations. The grid computing environment providing too much mechanisms and each of the grid computing member would install the grid middleware, if we want to fastly build a multi-cluster environment, these addtional software installtion will cause more overhead in our configuration. The gateway solution is suited when we have more high-speed inter-cluster connection network and we don’t want to use TCP/IP for inter-cluster communication. The Linux kernel now provide the system level of IP filter and NAT module, we don’t need to choose the SOCKS solution for building our multi-cluster, so we choose NAT solution to combine our single clusters to a multi-cluster.

2.5 Monitor systems For Beowulf Cluster

Several tools have been developed to monitor a large number of machines as

stand-alone hosts as well as hosts in a cluster. These tools can be useful because they monitor the availability of services on a host and detect if a host is overloaded, but they do not generally provide performance monitoring information at the level of detail needed to tune the performance of a Beowulf cluster. In contrast with existing systems, which usually display information only graphically, our project integrates performance monitoring with scheduling systems. In the following sections, we discuss open-source cluster-monitoring tools.

Ganglia [12] is an Open Source project (available on SourceForge at http://ganglia.sourceforge.net) with a BSD license. It grew out of the University of California, Berkeley, Millennium Cluster Project (see http://www.millennium.berkeley.edu) in collaboration with the National Partnership for Advanced Computational Infrastructure (NPACI) Rocks Cluster Group. Ganglia provides a complete, real-time monitoring and execution environment based on a hierarchical design. It uses a multicast listen/announce protocol to monitor node status, and uses a tree of point-to-point connections to coordinate clusters of clusters and aggregate their state information. Ganglia uses the extensible Markup Language (XML) to represent data, external Data Representation (XDR) for compact binary data transfers, and an open source package called RRDTool for data storage (in Round Robin databases) and for graphical visualization.

The *SMILE Cluster Management System* (SCMS) [25] is an extensible management tool for Beowulf clusters. SCMS provides a set of tools that help users monitor, submit commands, and query system status; maintain system configuration, and more. System monitoring is limited to heartbeat-type measurements.

Chapter 3

System Implementation

3.1 Multi-cluster with NAT Settings

In this section, the construction procedure is described. We used dual-processor motherboards to reduce the number of boxes to eight, and thus, minimizing the space needed for storage as shown in Figure 3.1 (and the footprint of the cluster). This structure impacts performance because two processors share the memory bus (which causes bus contention but reduces the hardware cost) since only one case, motherboard, hard drive, etc., are needed for two processors. We ruled out the option of rack-mounting the nodes, essentially to reduce cost, but chose to use standard mid-tower cases on shelves. This approach is occasionally given the name of LOBOS (lots of boxes on shelves).

The idea of the Linux cluster is to maximize the performance-cost ratio of computing by using low-cost commodity components and free-source Linux and GNU software to assemble a parallel and distributed computing system. Software support includes the standard Linux/GNU environment, including compilers, debuggers, editors, and standard numerical libraries. Coordination and communication among the processing nodes is a key requirement of parallel-processing clusters. In order to accommodate this coordination, developers have created software to carry out the coordination and hardware to send and receive the coordinating messages. Messaging architectures such as MPI or Message Passing Interface, and PVM or Parallel Virtual Machine, to allow the programmer to ensure that control and data messages take place as needed during operation.

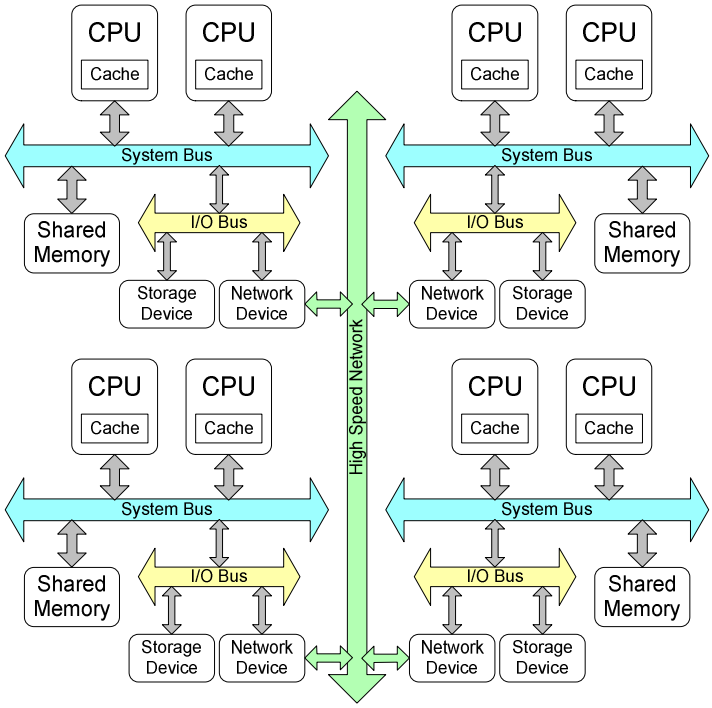

Figure 3.1: NOW using four SMP PCs

When we have many single clusters and wonder build them as a big and single computation resource with NAT, The master node of our single cluster are connected with inter- and intra-cluster networks. Key point is that how could we configure our master nodes of many of our single clusters and make them could communicated over the network.

In general, a PC cluster is constructing by using Linux as their OS environment. The kernel version 2.4 and latter provides a subsystem called Netfilter. It operates a packet filtering job in kernel stack. We use this feature for IP-Masquerade, a solution of network address translation (NAT), in the setting, we just open the capability of packet forwarding module. The reason is that we do not need to protect any type of attack form Internet; we just want to make the client that can communicate around the public network. IP-Masquerade in Linux has several ways to be enabled. You should use the root account to perform the following commands.

The building steps of multi-cluster in each cluster's master node are list below. First, you can use "**sysctl**" command to enable kernel IP forward function like this

command "**sysctl -w net.ipv4.ip_forward=1**". Another way, you can change the "**/proc/sys/net/ipv4/ip_forward**" value to 1. Then, you can use "**iptables**" command to change Netfilter trains, before this configuration you should load kernel module include **ip_tables** and **iptable_nat** through **"modprobe"** or **"insmod"** command, after modules loaded. The command is like below:

iptables -t nat -A POSTROUTING -o <eth> -s <private_net> -j MASQUERADE

under the command, the **"eth"** parameter is the mapped to your master node's public address interface and the **"private_net"** parameter is your cluster's private network address, if your private network is 192.168.1.0 and its netmask is 255.255.255.0. You should type **"192.168.1.0/24"**.

After enabling the NAT support for each cluster's master node, if we want to combine these clusters together, we must build a static routing path that causes every node in this environment can pass messages to each other.

We use a simple example to explain this concept. In Figure 3.2, if we have two 2-client clusters Cluster A and Cluster B. In their private network setting, Cluster A and Cluster B should configure different private network address like as 192.168.1.0 and 192.168.2.0 or other, respectively. After IP-Masquerade in Linux kernel, we can look each master node as a network router, in this concept, we don't mind that private network couldn't route between public networks because every cluster master node in our setting is in public network. Our configuration is concentrated on made the routing policy to each cluster master. The policy is simple, if we must combine N cluster together, each master must add (N-1) route trains to satisfy each cluster communications together.

In this example for cluster A, we can add this command to add a routing train on this cluster master:

route add –net 192.168.2.0 netmask 255.255.255.0 gw <cluster B master IP >

For cluster B, we also add the command on cluster master:

route add –net 192.168.1.0 netmask 255.255.255.0 gw <cluster A master IP >

After this configuration and master node is set to use IP-Masquerade. We easily can combine the two Linux PC clusters for applying MPI parallel applications for obtain large computation power.

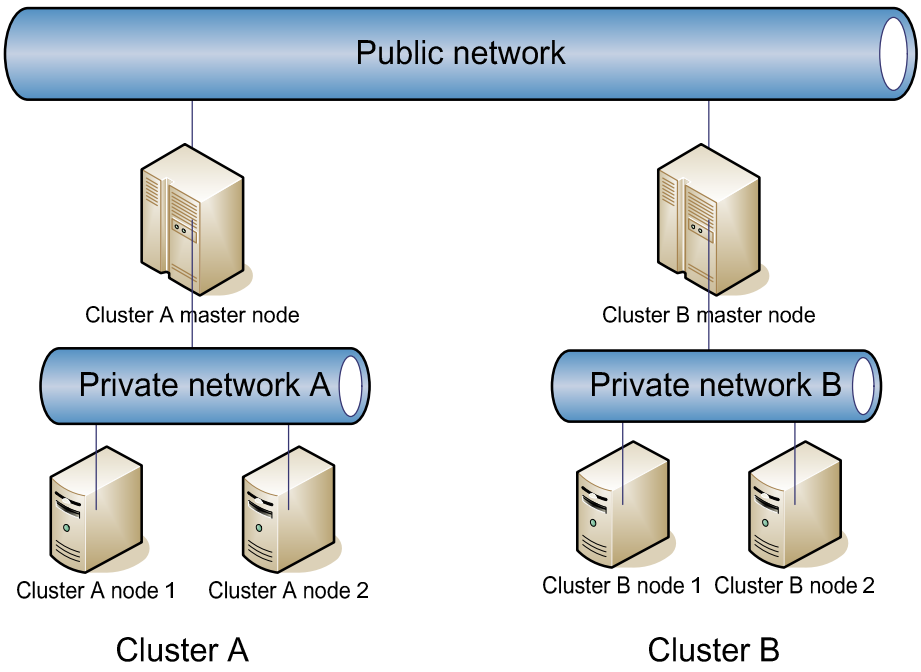

Figure 3.2: A simple PC form consists of two PC clusters

3.2 Monitor System Design and Implementation

The concept of our monitor system is to improve the availability of monitor system in the distributed computing environment. Nowadays, the monitor system is not well developed on user requirements. So we started on the user interaction and the manner of application executing, and developed the applications by the portability of the Java Virtual Machine. Our system can be divided into three applications of the Cluster architecture:

- **Observe server:** The role of the Observe server is running the collect daemon that getting the information of each cluster's total information that observing from the master node and replicate the data to their local file based

database for the usage of the web interface,

- **Master node:** The master node executes the master daemon that could collect the information of their slave nodes to their local file based database and response the Observe server,
- **Slave node:** All other nodes of our cluster must to run the slave daemon. The slave program must get information in user specific metrics like CPU speed, available size of memory, load of this node and other information user interested in.

In the other side, we have a separate role to display and control our cluster in another way, there are two types of this role:

- **Web portal:** We use two tools that generate the web service for controlling and displaying the information of our system; first we use the drawing tool called Round Robin Database Tool (RRDtool) to draw the state chart from the collected information in the Observe server. Second is the web front-end portal created by PHP, when user is connecting to the portal, he can retrieve the information by the state chart and control the system by the web interface.
- **PDA application:** The mobile devices are not suit for displaying detail information and remote controlling. We need to simplify our information and design the appropriate interface for this usage. The Java application framework is suitable for this type of application and we choose it to develop our simple application. Implementation of this work is connecting the Observe server and gets all metrics of our information and directly displaying these to a classified format.

Our system has been implemented for machines with a private network; it resides on one node, which controls all the others with remote commands. That choice allows easy installation and upgrade, and need to have daemons running on computing and

service nodes. On the other hand this choice can scale if the number of nodes is huge. The software has been implemented for managing a cluster of clusters, on public networks.

The flow of this system is shown in Figure 3.3, the master nodes can collect the system information from its slave nodes in the multiple Linux PC clusters. Then the Observe server will gather all information from master nodes, and send to Web server for displaying form remote users and applications. The system architecture and software architecture are shown in Figure 3.4 and Figure 3.5. Also, the Observe server will be called to provide services and information form the Web server.

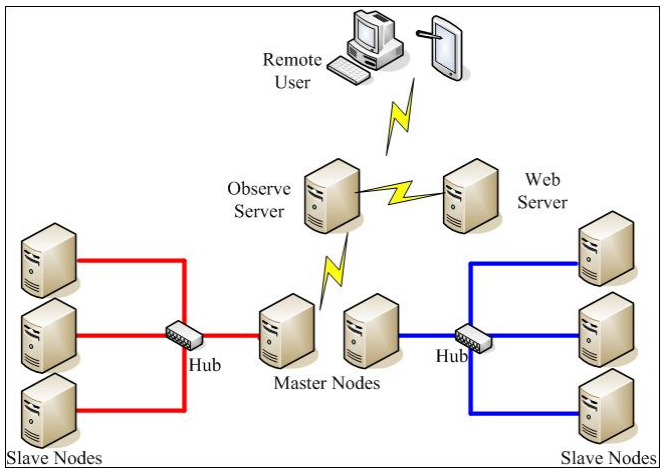

Figure 3.3: Overview of monitor system

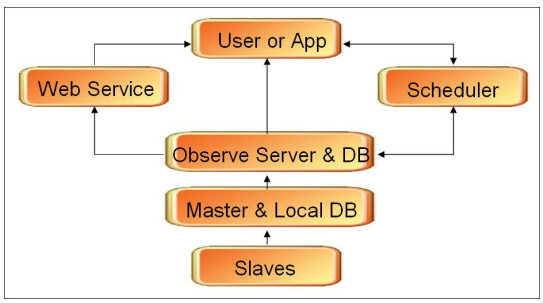

Figure 3.4: Monitor system architecture

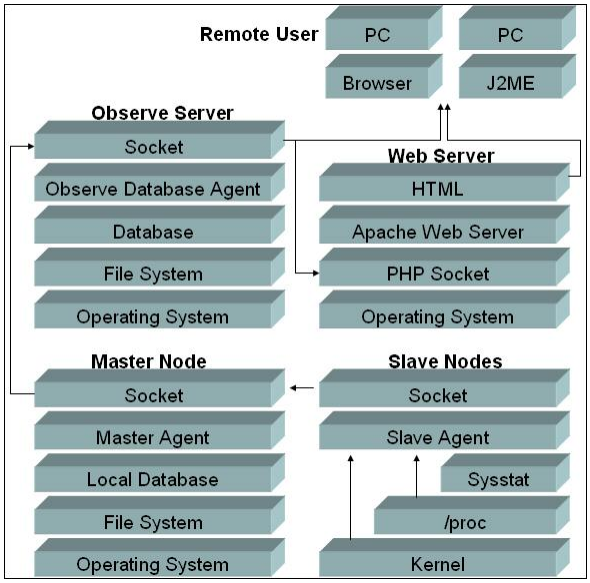

Figure 3.5: Software architecture of our monitor system

The functions of three daemons in our system are listed as below:

- Slave Daemon: The Slave daemon can obtain the related system information of each slave node form Kernel, and provide the services to its Master node in the PC cluster,
- Master Daemon: The Master Daemon is responsible to collect the system information from all slave nodes in its cluster, and put the related information into Local Database. Local Database is used for the purpose that will not allow the high load of master node for an instant. The function of Local Database can be view as a buffer,
- Collect Daemon: This daemon is running on the Observe Server. It is used for collecting the system information of each master node of multiple Linux PC clusters. It can provide services by using database to applications or the remote users.

All server daemons are written in C and the web portal interface is written in PHP, We could get some information through our web portal and they are shown from Figure 3.6 to Figure 3.9. For the user without computer, we have also implemented a JAVA-based PDA version of our monitor shown in Figure 3.10.

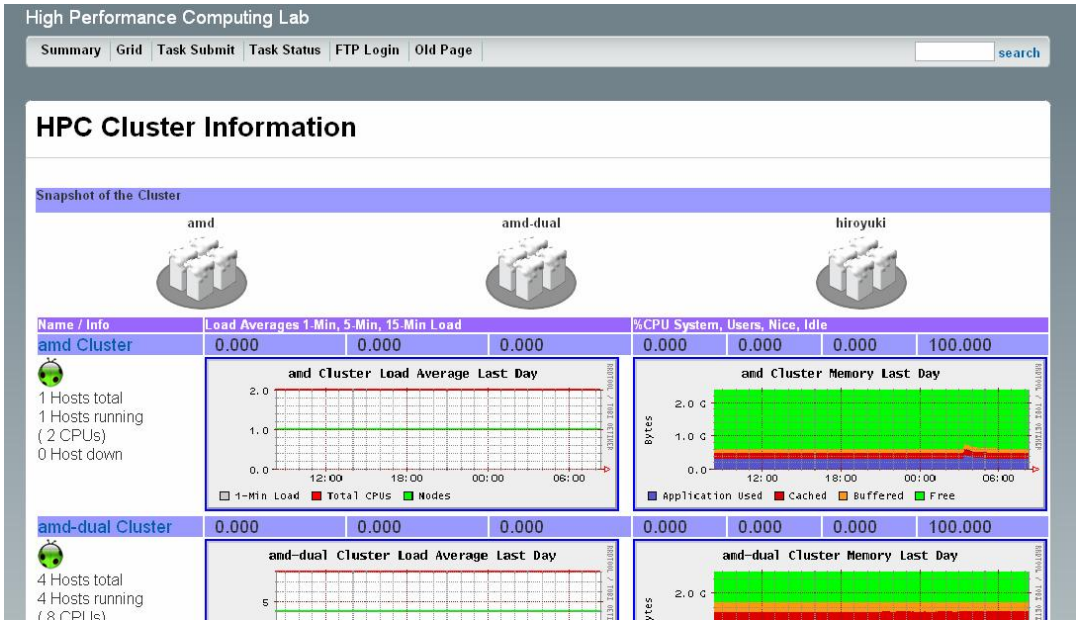

Figure 3.6: Home page created with PHP and RRDTool

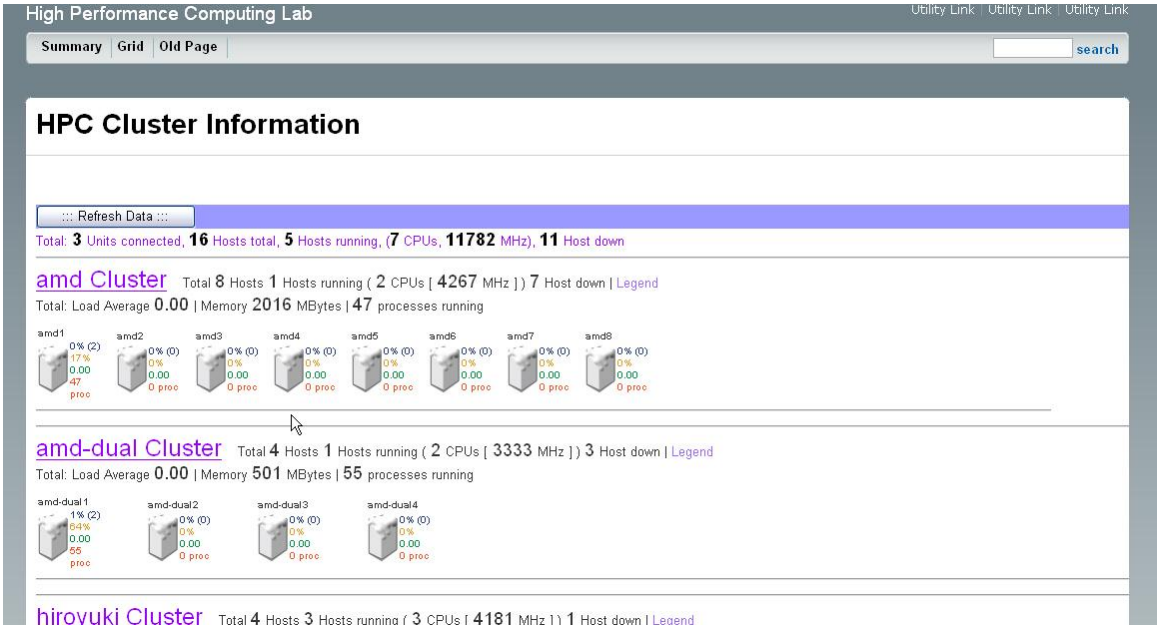

Figure 3.7: Platform information page

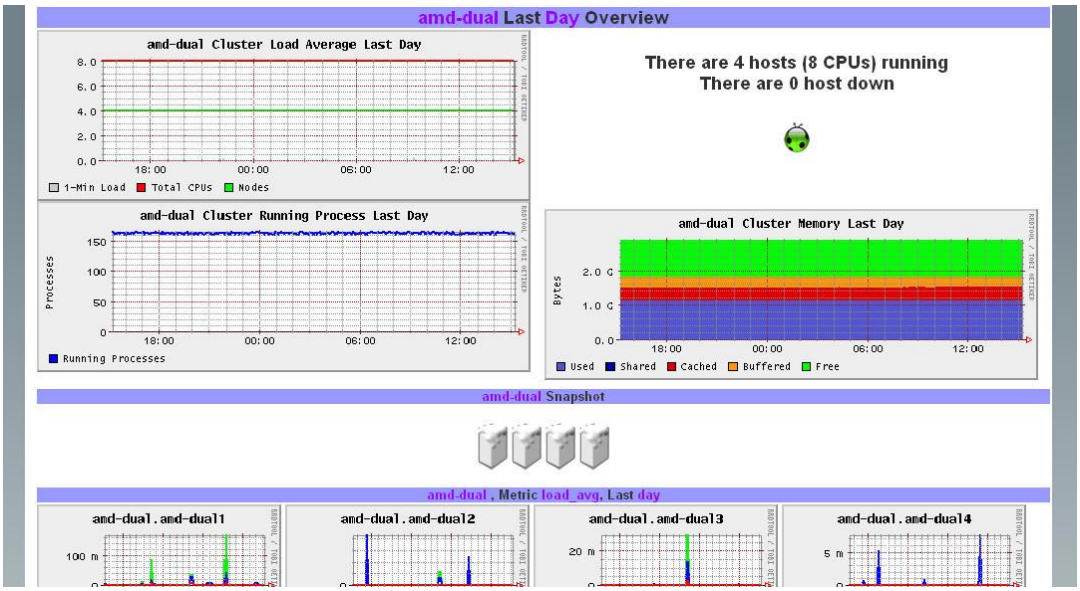

Figure 3.8: The overall system information webpage of one cluster platform

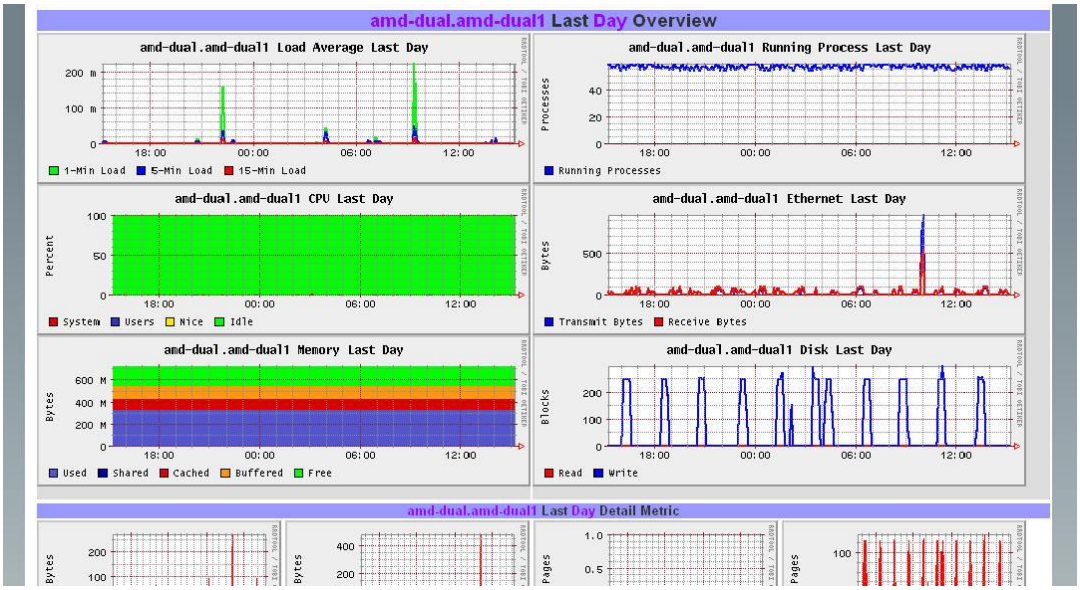

Figure 3.9: The detail system information webpage

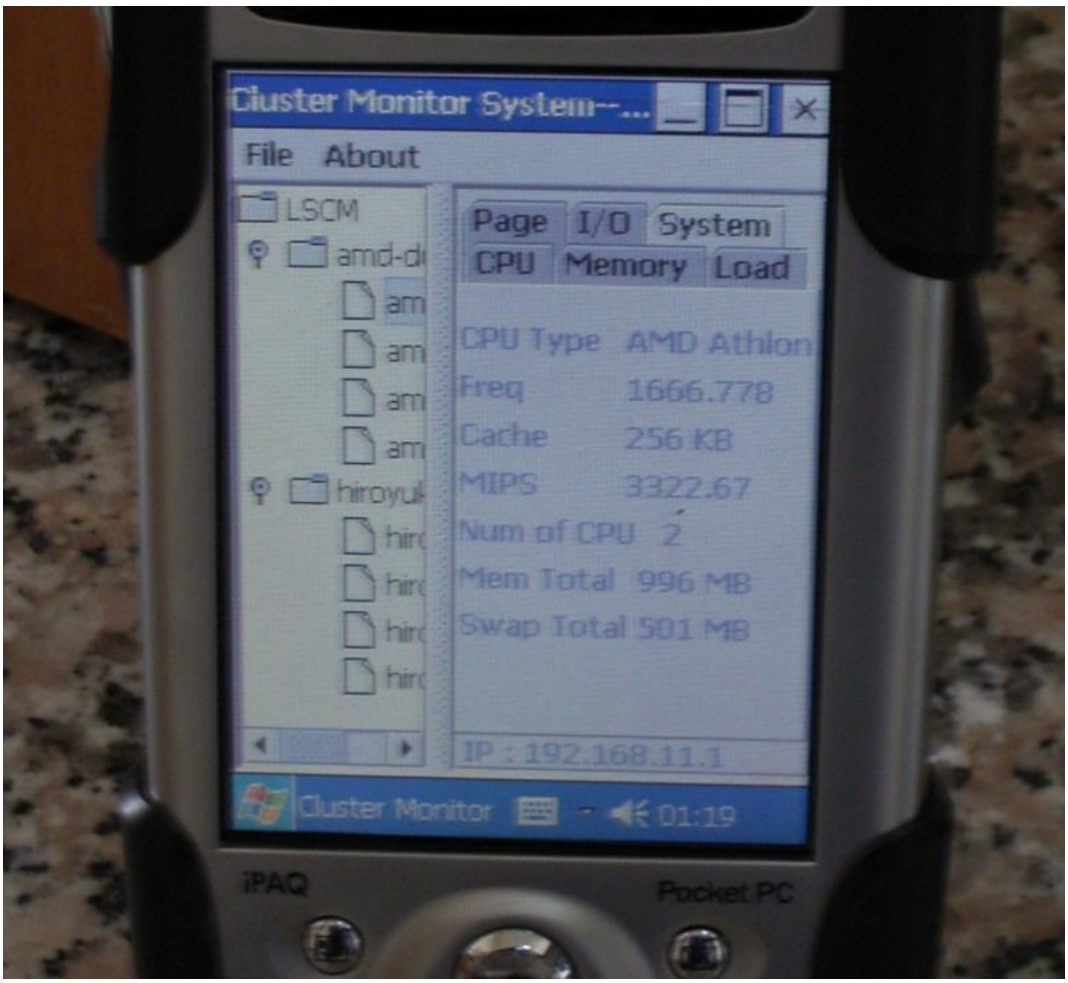

Figure 3.10: Monitoring screen on PDA

Chapter 4

Experimental Results

4.1 Experimental Environment

Table 4.1 shows the hardware specification of our testing environment. The network topology is shown as in Figure 4.1. The hardware specifications of them are list in Table 4.1. Our inter-cluster connection are build on the Fast Ethernet that giving the speed of 100 Mb/s.

Table 4.1: Hardware specification

CPU	Memory	Network Type	# of total node	Cluster hostname
Dual AMD AthlonMP 2400+	Master 2GB Slave 1GB	1Gbps for intra-cluster	Four nodes	amd1
Dual AMD Athlon MP 1800+	512MB	100Mbps for intra-cluster	Four nodes	amd-dual1
Dual AMD Athlon MP 2000+	Master 2GB Slave 1GB	1Gbps for intra-cluster	Four nodes	amd-dual01
Dual AMD Athlon 64 3000+	Master 1GB Slave 1GB	1Gbps for intra-cluster	Four nodes	amd64-dual01

The configuration steps of multi-cluster are listed below.

- First, get all information of public and private IP addresses of these four cluster's master on executing "**ifconfig**" command,
- Second, all the slave nodes default gateway should be the cluster's master node in the file named **"/etc/sysconfig/network"**,
- Third, setting the configurations for the master node of three clusters with the follows in upper section we described,

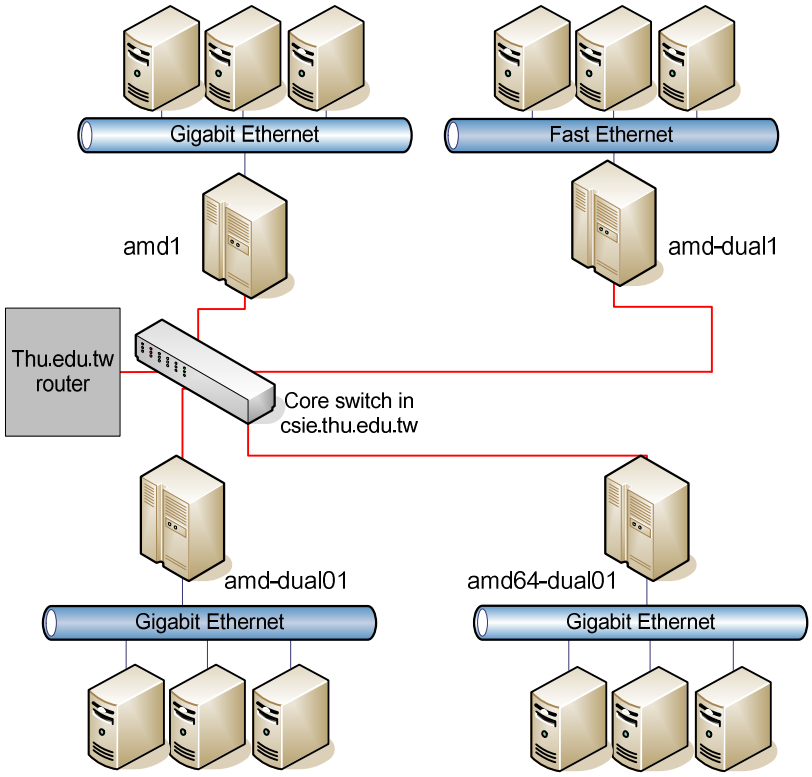

Figure 4.1: Experimental environment in HPC Lab consisting of four PC clusters

In order to compare the difference of various numbers of nodes in multi-cluster combination and our single cluster, we set three cases in the multi-cluster in Table 4.2. In each multi-cluster settings, each node of our cluster has dual CPUs, so the case named “mc8” has four nodes, while the others are the same. In every multi-cluster case is also including the master node of each single cluster to build with. We can regard mc8 cluster as the eight CPUs case of heterogeneous cluster with Fast Ethernet because the master are connected in the LAN environment, and we can use this case to compare the performance to other single clusters in our testing cases.

To compare the performance differences of mc16 and mc32 with single cluster that has the same numbers of CPUs. We build the sc16 and sc32 clusters that the hardware components are the same as amd-dual01, but sc16 is build with 8 nodes and sc32 is build with 16 nodes, and their intra-cluster network environment is Gigabit Ethernet. We can use these two single clusters to compare the total difference when we evaluate with mc16 and mc32.

In another approach, we want to know how the starting node of the MPI job does influence the performance of each testing case. We use the master nodes of our four single clusters to evaluate this issue. We will discuss this issue in the rear of each testing case.

Table 4.2: Test setting of single and multi-clusters

Type	Numbers of nodes			
	amd1	amd-dual1	amd-dual01	amd64-dual01
mc8	1 node	1 node	1 node	1 node
mc16	2 nodes	2 nodes	2 nodes	2 nodes
mc32	4 nodes	4 nodes	4 nodes	4 nodes

In choosing implementation of the MPI library, we use MPICH to build for our experimentation because the new version of LAM/MPI that's after 6.5.8 could not suite to multi-cluster in private to public IP address translation. We use the MPICH 1.2.6 to compile our experimental program in the multi-cluster environment. In our experimental environment, we have two different hardware platform i386 for AMD ATHLON MP and x86_64 for AMD ATHLON 64 platform, so we should make different binary codes for MPICH implementation and of course, the experiment binary codes are also expected.

4.2 Performance Comparisons

In order to compare our multi-cluster performance, we choose four parallel applications for evaluations: Matrix multiplication, PI computation, prime number generation, and the bucket sort.

4.2.1 Matrix Multiplication

The matrix operation derives a resultant matrix by multiplying two input matrices, a and b, where matrix a is a matrix of N rows by P columns and matrix b is of P rows by

M columns, and then the resultant matrix c is of N rows by M columns. The serial realization of this operation is quite straightforward as listed below:

```
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 $O(n^3)$. 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.

In the Figure 4.2, 4.3, and 4.4, we could find out the amd64-dual01 leads the performance in these four single clusters because of its new architecture and the optimization for the x86_64 architecture in new kernel 2.6. The amd-dual1 cluster has the worst performance because of its network speed is only 100Mbps speed and the CPU clock is the slowest of these four single clusters. We can also check the performance of amd-dual1 and mc8. It shows that when the matrix size increased the performance of both cases are lower than other cases in the 100Mbps network.

In the performance evaluation results in Figure 4.2 shows the performance benefit in the multi-cluster system. First the performances of the mc8, mc16 and mc32 clusters are between the performance of amd1 and amd64-dual01 these two clusters. When the matrix size is not very large, the communication overhead occur the execution time of mc32 and mc8 are very closely. But we can get the better performance in the mc16 cluster.

When the matrix size increased as in Figure 4.3 and Figure 4.4, it shows if we increase our nodes to 16 to execute the matrix multiplication program, we could get the

best performance changes. The case of the matrix size increasing to 2048, the performances of single clusters are not the same in the Figure 4.2 and Figure 4.3, but multi-cluster also shows the better performance in the mc16 and mc32 settings. Though, the case of mc8 shows the worst performance because the network speed of the mc8 cluster is the Fast Ethernet and in normal LAN environment communications that is like the case of amd-dual1.

When we compare our multi-cluster settings with single cluster settings with matrix multiplication, our program must send the divided matrix as a multiplier and the full matrix as multiplicand with traditional block send and receive functions. We could find that when the matrix size is 512 in Figure 4.2, sc32 could not have the poor performance than the mc32 because the communication over the single cluster is running through a single switch that has bigger intra-communication bandwidth. The results in larger matrix size are more significant when we have strong intra-communication bandwidth in Figure 4.3 and Figure 4.4. In this type of parallel processing type that has large message passing over the computation nodes, the multi-cluster only has little advantage even has bed performance.

In the experiment on starting node of the MPI job. As shown in Figure 4.5, we respectively conduct the relation in the execution time and the availability of the MPI job start location. We can get the better performance when we choose the amd64-dual01 as the node for job starting than other nodes, the amd1 is the second fast, the amd-dual01 is the third and the last is the amd-dual1. It could be compared with the performance of single cluster in Figure 4.2, 4.3 and 4.4. We can find out that if we choose node that gets better performance in single cluster case to starting the MPI job, we can get better performance when we combine our single cluster to multi-cluster.

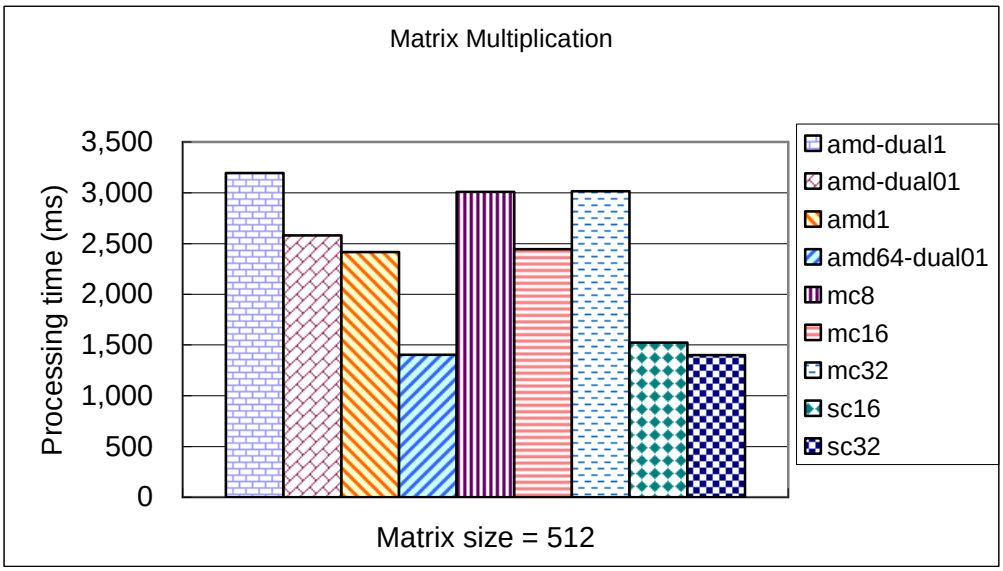

Figure 4.2: Comparison of matrix multiplication when the matrix size is equal to 512

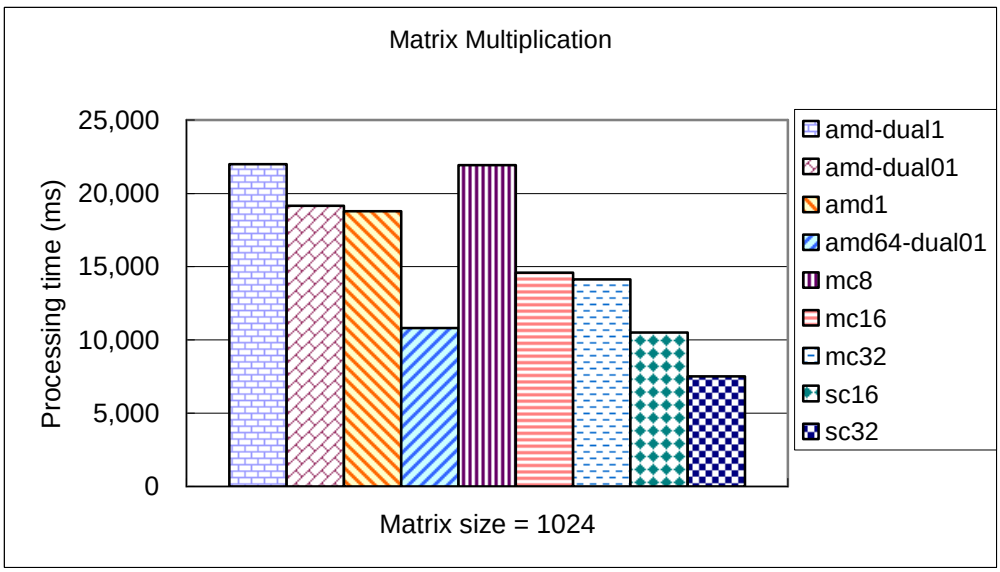

Figure 4.3: Comparison of matrix multiplication when the matrix size is equal to 1024

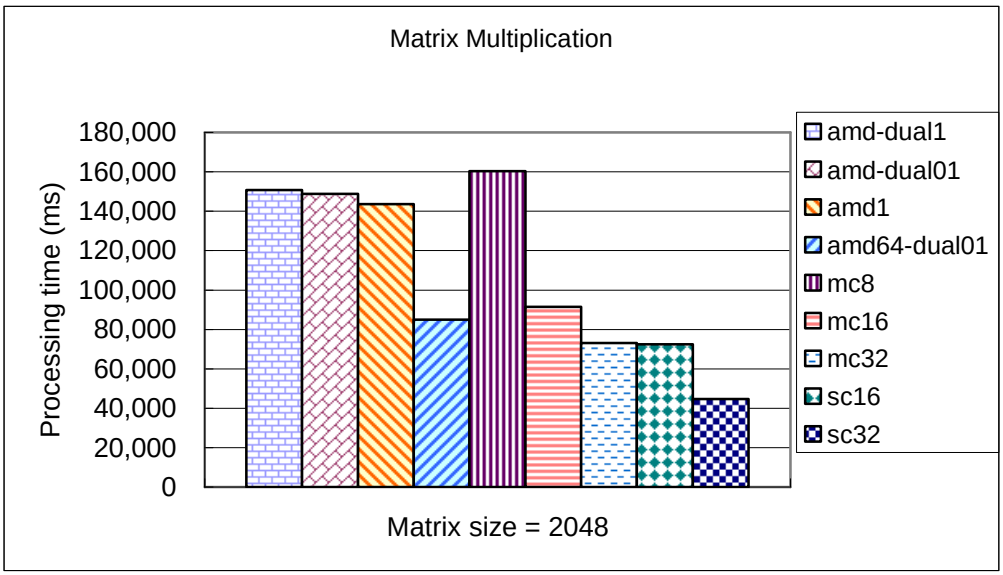

Figure 4.4: Comparison of matrix multiplication when the matrix size is equal to 2048

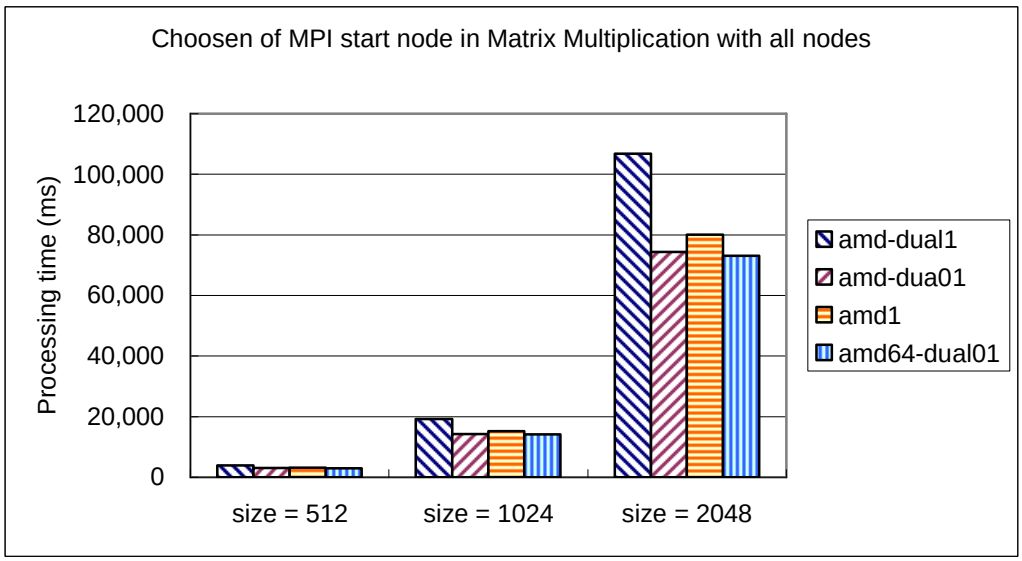

Figure 4.5: Comparison on MPI jobs starting locations in matrix multiplication

4.2.2 PI Computation

The PI computation program calculates the value of PI by numerical integration in order to estimate the area of the curve $f(x) = 4/(1+x^2)$ between 0 and 1. We can compute an approximation by dividing the interval zero and one into numbers of subintervals to each parallel process and then let each parallel process compute the area of their own subset.

As the previous section we first compare the performances of four single clusters. The order of processing time in descending order is amd64-dual01, amd1, amd-dual01, and the last is amd-dual1 cluster and this result is no matter the subspaces is bigger. The results reflect the processing time on different MPI jobs starting nodes in Figure 4.8.

In Figure 4.6 and Figure 4.7 we could find out that the case of mc16 and mc32 have good performance when the node increasing. Comparing this result with the mc8, it shows the communication in heterogeneous cluster with eight CPUs in the Fast Ethernet environment has very heavy overhead and when we add more nodes the problem is dividing to smaller sets that cause good performance.

The totally performance comparison of mc16, mc32, sc16, and sc32 is described below. The MPI program of the PI problem broadcasts divided part of each computation node with MPI_Broadcast function and last the master receive the results using MPI_Reduce function; the communication overhead of this problem is very small, and this problem speed is not the primary factor. So the results of multi or single clusters are almost the same.

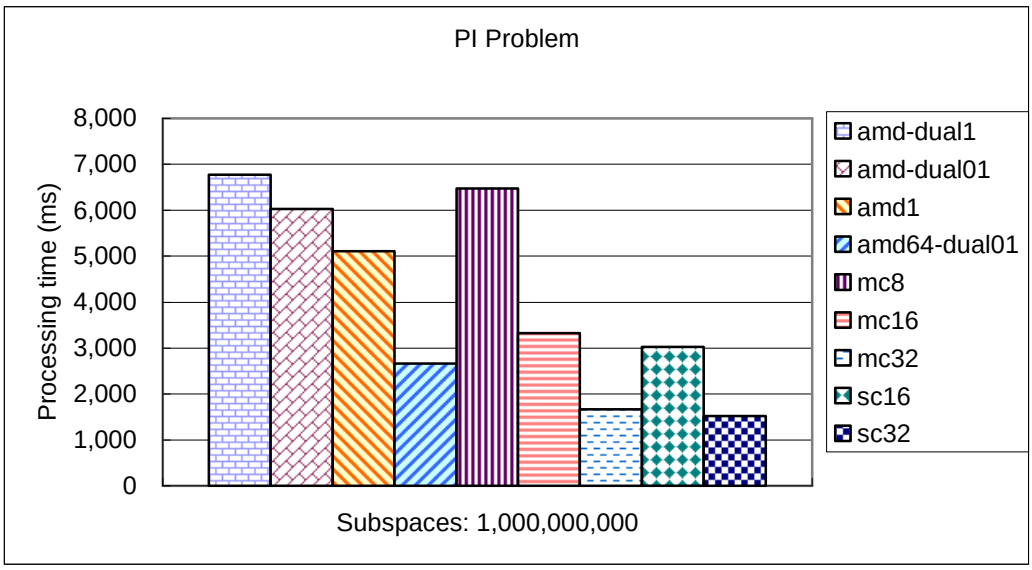

Figure 4.6: Comparison of the PI problem when the subspaces are equal to 1,000,000,000

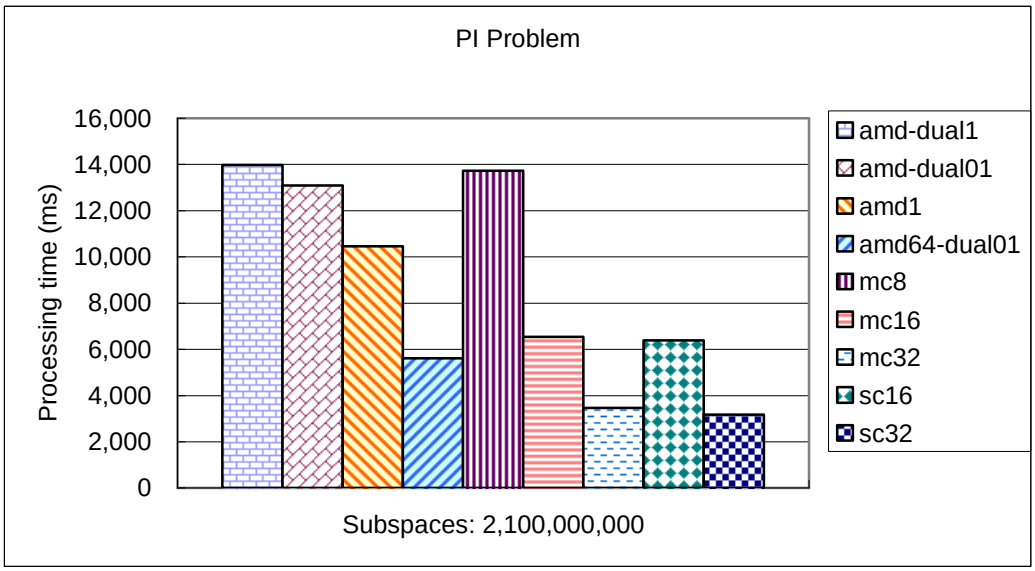

Figure 4.7: Comparison of the PI problem when the subspaces are equal to 2,100,000,000

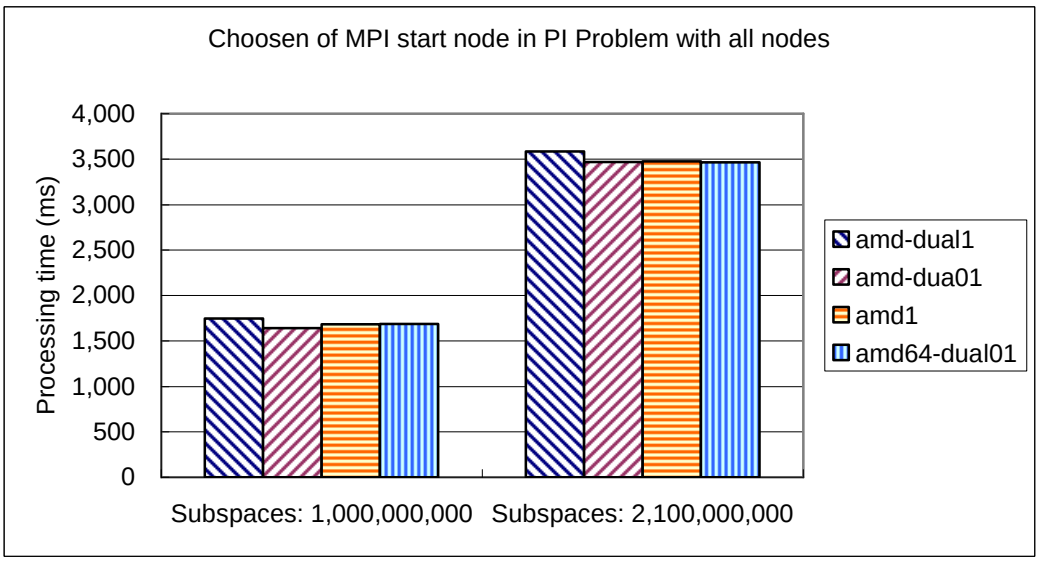

Figure 4.8: Comparison on MPI jobs starting locations in PI problem

4.2.3 Prime Number Generation

This problem is to generate the largest prime number between 1 and one given number. For instance we want to find the largest prime number between 1 and 20,000,000. A parallel program that initially runs on a lead node and sends the task of testing number 101-200 to node 1, and sends the task of testing number 201-300 to node 2, and so on. Along with the testing task, there would also be an instruction to return whatever primes a slave node has discovered to the lead node. When all nodes

have completed their tasks, there will have a message to tell you how many primes are found and what the biggest prime number is.

In this case, we evaluate two different integer ranges to observe the performance. The performance comparisons are shown in Figure 4.9 and Figure 4.10. In our environment when the available nodes increased like mc16 and mc32, we have the better performance than single cluster settings. So the performance ratio increased regardless the problem size even when we add our number of nodes to 32.

The order of processing time in descending order is amd1, amd64-dual01, amd-dual01 and amd-dual1. The amd1 leads the performance of single cluster and the reason is that the prime number generation is not using the float-point computation power, this problem needs the integer computation performance. The bogomips value shows the integer performance of specific processor in the kernel information that consisting in /proc/cpuinfo file. The bogomips value of amd1 is 4230.52 and the bogomips value of the amd64-dual01 is 3915.77, so we can use this value to explain why the amd1 leads the performance.

In the prime number generation problem, however the communication through each computation node is very frequency and the size for each communication is very small, the primary factor for this problem is the computing power of each processor. So in the totally performance evaluation of this problem, the mc16 and mc32 still has the advantage than the sc16 and sc32.

We still evaluate the performances on starting node of the MPI job; we can compare it in Figure 4.11 with performances in single clusters in Figure 4.9 and Figure 4.10. The performance of single clusters reflects the starting node of the MPI job.

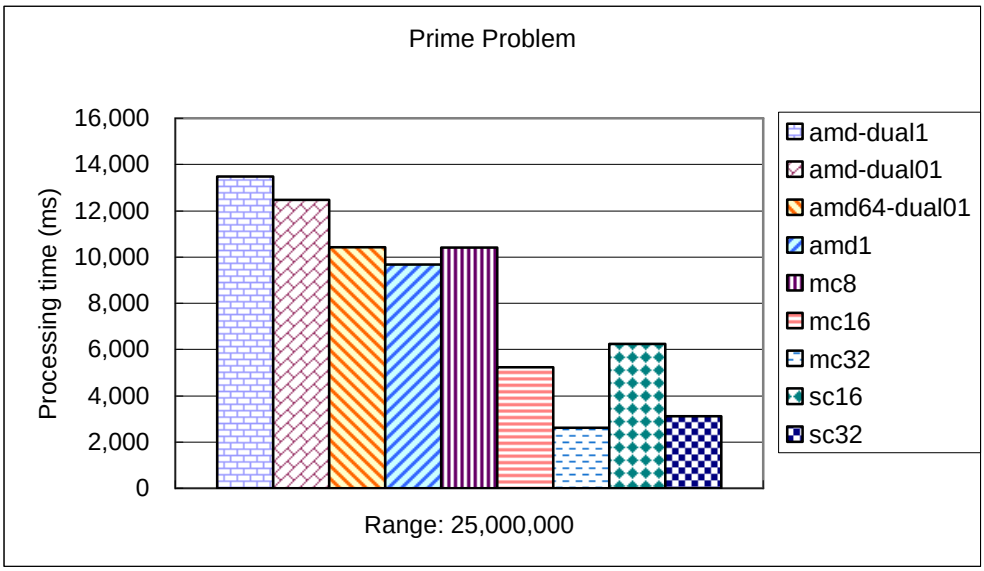

Figure 4.9: Comparison of the prime problem when the range is equal to 25,000,000

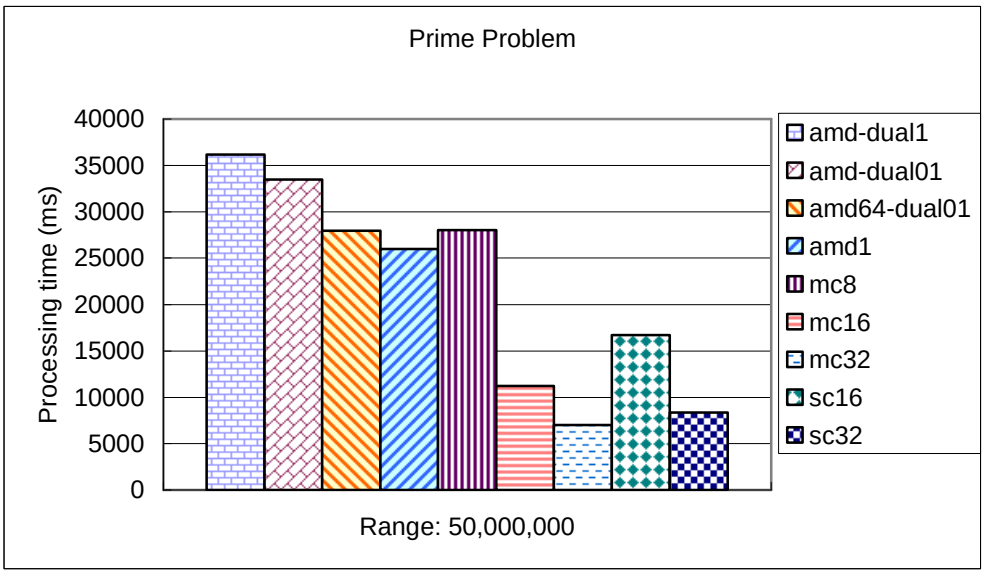

Figure 4.10: Comparison of the prime problem when the range is equal to 50,000,000

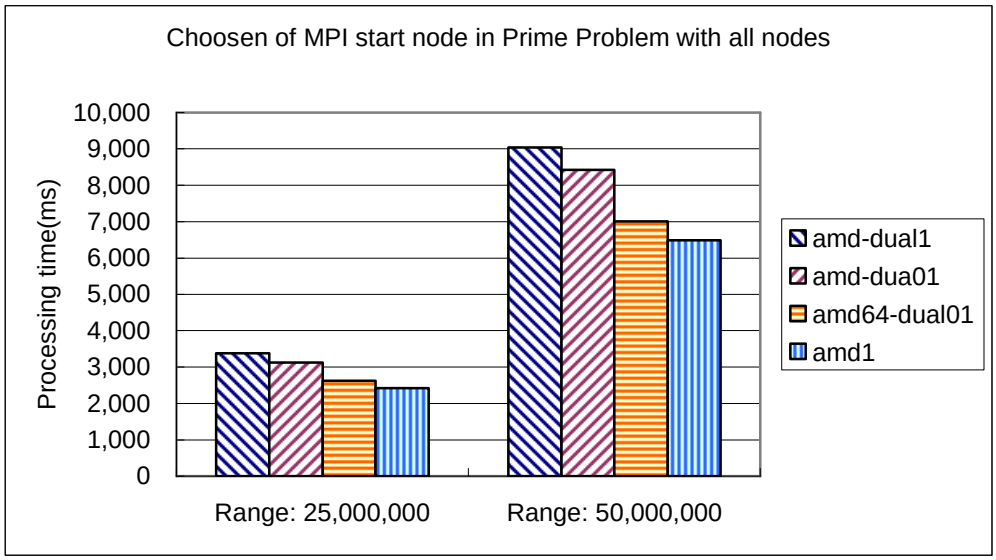

Figure 4.11: Comparison on MPI jobs starting locations in the prime problem

4.2.4 Bucket Sort

This problem is that we have a amount of unsorted numbers in a list and we want to sort it to a numerical order. The bucket sort utilizes a divide-and-conquer method to solve this problem. In parallel version of the bucket sort, there are n numbers in a unsorted list and p processors to solve problem. First we partition this list to p regions, one region for each processor. Each processor maintains p "small" buckets and separates the numbers in its region into its own small buckets. These small buckets are then emptied into the p final buckets for sorting, which requires each processor to send one small bucket to each of the other processors.

We also use two different size of array to test our single and multi-clusters. In Figure 4.12 and Figure 4.13 show the performance difference between single and multi-clusters. We observe that the amd1 also lead the single cluster performance as we have described in Section 4.2.3 and this results also reflect the starting node of the MPI job. This case also shows that when we use 8 CPUs cluster like mc8 the bucket sort application brings the communication overhead and when we increase our number of nodes like mc16 and mc32 we finally get good performance.

The different of bucket sort and the prime number generation is that because the bucket sort communication overhead is heavy than the prime number in data exchange, We find out that un the MPI jobs starting locations comparison in Figure 4.14, The size of the array could not effect the performance of this problem, the multi-cluster performance would be very closely in this case. Totally we can observe when we build a multi-cluster using 16 nodes and 32 nodes we could still conduct good performance in this case of jobs.

In the totally performance evaluation for multi and single clusters with mc16, mc32, sc16, and sc32, the property for communication overhead is likely as the matrix multiplication problem and the computation is very heavy with each computation node. When we use the small a list for sorting, the single cluster has the better performance but using larger size of list in Figure 4.13 the network advantage for single cluster couldn't help too much in this problem. The computation advantage for multi-cluster could eliminate the communication overhead in this type of problem. So in large computation and communication overhead are both exist such like this problem, multi-cluster could get some advantage even has the same performance in single cluster.

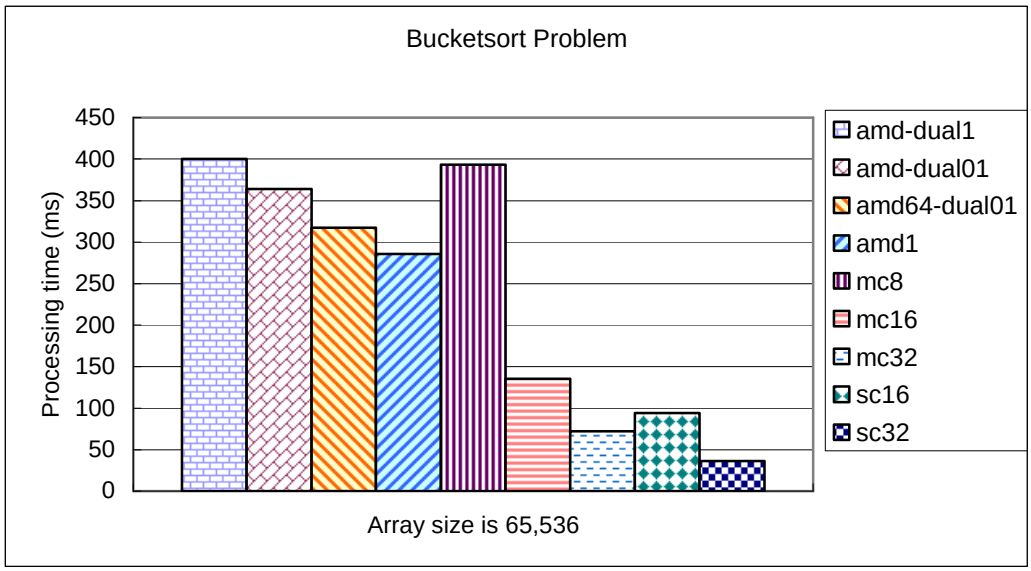

Figure 4.12: Comparison of the bucket sort problem when the sizes are equal to 65,536

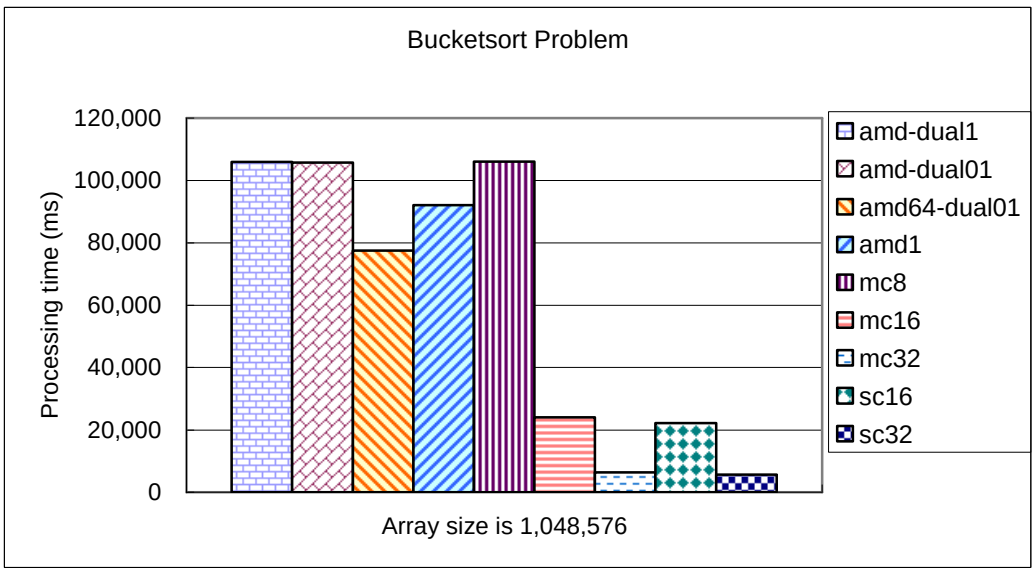

Figure 4.13: Comparison of the bucket sort problem when the array sizes are equal to 1,048,576

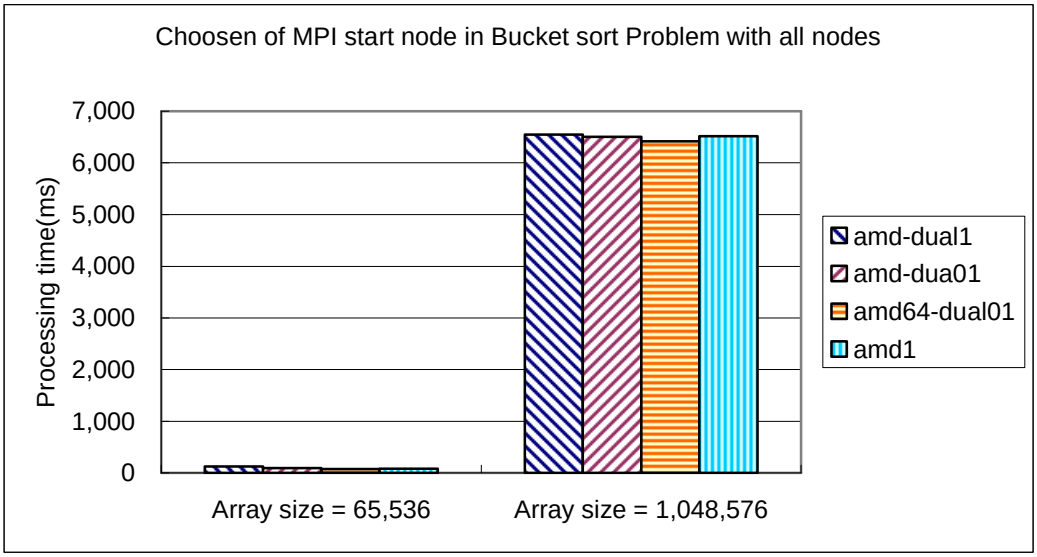

Figure 4.14: Comparison on MPI jobs starting locations in the bucket sort problem

Chapter 5

Conclusions and Future work

To build a cluster computing platform, we commonly inter-connect a number of personal computers or workstations to achieve high computing power. In general case, we don’t have enough space for setting up many PCs on it, and questions we are concerned about arises, for instance, network topology, electric power, and management question etc. Then we can use several places for setting up a number of machines on each place, and enable networking for transparent computing on each cluster node for enable an enormous nodes cluster in LAN environment. In this thesis, we use the NAT mechanism to inter-connect two or more clusters to connect a big one. An experimental environment we use four different speeds of PC cluster in a Fast-Ethernet inter-cluster network environment for testing many parallel applications. We could find that the multi-cluster with NAT bring us the more computation power. The NAT mechanism is a easier way to build such a multi-cluster environment than other solutions. In the future we would like to evaluate different message passing library such as PVM or different implementations of MPI to help us enhance the performance in multi-cluster setting, or integrate the multi-cluster environment with scheduler system and the second generation Beowulf-Cluster architecture.

Bibliography

[1] T. Anderson, D. Culler, and D. Patterson, A Case for Network of Workstations, IEEE Micro, 15(1):54-64, Feb. 1995.

[2] D. Balkanski, M. Trams, W. Rehm, "Communication middleware systems for heterogeneous clusters: a comparative study", *Proc. of 2003 IEEE International Conference on Cluster Computing*, pp 504- 507, IEEE Computing Society, 2003.

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

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

[5] F. Cappello, O. Richard, D. Etiemble, "Understanding performance of SMP clusters running MPI programs", *Future Generation Computer Systems*, (17) pp 711-720, Elsevier Science, 2001

[6] I. Foster and C. Kesselman, "*The Grid: Blueprint for a Future Computing Infrastructure"*, Morgan Kaufmann Publishers, 1999.

[7] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Journal of Supercomputing Applications*, vol. 15(3), 2001.

[8] E. Gabriel, M. Resch, T. Beisel, and R. Keller, "Distributed computing in a heterogeneous computing environment", *Proc. of the 5th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pp 180-187, Springer-Verlag, 1998.

[9] Al GEIST, Cluster Computing: The Wave of the future, Springer Verlag Lecture Notes in Computer Science, May 1994.

[10] LAM/MPI official site, http://www.lam-mpi.org/.

[11] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "*SOCKS Protocol Version 5*", RFC 1928, Internet Engineering Task Force, 1996.

[12] M. L. Massie, B. N. Chun, D. E. Culler," The ganglia distributed monitoring system: design, implementation, and experience", *Parallel Computing*, (30) pp 817-840, Elsevier Science, 2004.

[13] M. Matsuda, T. Kudoh, Y. Ishikawa, "Evaluation of MPI Implementations on Grid-connected Clusters using an Emulated WAN Environment", *Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)*, IEEE Computing Society, 2003.

[14] Message Passing Interface Forum, http://www.mpi-forum.org/.

[15] MPI Forum. "MPI: A message-passing interface standard". *International Journal of Supercomputer Applications*, 8 (3/4) 165-416, 1994.

[16] MPICH official site, http://www-unix.mcs.anl.gov/mpi/mpich/.

[17] M. Müller, M. Hess, E. Gabriel, "Grid enabled MPI solutions for Clusters", *Proc. of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'03)*, IEEE Computing Society, 2003.

[18] K. N. Nguyen and T. T. Le, "Evaluation and Comparison Performance of Various MPI Implementations on an OSCAR Linux Cluster", *Proc. of the International Conference on Information Technology: Computers and Communications (ITCC.03)*, IEEE Computing Society, 2003.

[19] H. Ong and P. 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.

[20] PVM – Parallel Virtual Machine, http://www.epm.ornl.gov/pvm.

[21] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator

(Traditional NAT),” RFC 3022, Internet Engineering Task Force, 2001.

[22] T. L. Sterling, J. Salmon, D. J. Backer, and D. F. Savarese, *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*, 2nd Printing, MIT Press, Cambridge, Massachusetts, USA, 1999.

[23] Y. Tanaka, M. Sato, M. Hirano, H. Nakada, S. Sekiguchi, “Performance Evaluation of a Firewall-compliant Globus-based Wide-area Cluster System”, *Proc. of the Ninth IEEE International Symposium on High Performance Distributed Computing*, pp 121-128. IEEE Computing Society, 2000.

[24] D. Turner and X. Chen, “Protocol-dependent message-passing performance on Linux clusters”, *Proc. of the 2002 IEEE International Conference on cluster computing*, pp 187-194, IEEE Computing Society, 2002.

[25] P. Uthayopas and A. Rungsawang, “SCMS: An Extensible Cluster Management Tool for Beowulf Clusters”, Supercomputing 99, Portland (OR), 1999.

[26] B. Wilkinson and M. Allen, *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall PTR, NJ, 1999.

[27] C. T. Yang, C. C. Hung, and C. C. Soong, “Parallel Computing on Low-Cost PC-Based SMPs Clusters,” *Proc. of the 2001 International Conference on Parallel and Distributed Computing, Applications, and Techniques* (*PDCAT 2001*), Taipei, Taiwan, pp 149-156, July 2001.