Building a High Performance Cluster through Computer Reuse

A Major Qualifying Project Report submitted to the Faculty of the Worcester Polytechnic Institute in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science

By

Christopher T Clark

Submitted On: 26 October, 2010

Submitted To:

Professor Xinming Huang, Advisor, Electrical and Computer Engineering

Professor Erkan Tüzel, Advisor, Physics

Abstract

The goal of this MQP was to use “outdated” commodity PCs to build a cluster computer capable of being used for computationally intensive research. Traditionally, PCs are recycled after being taken out of use during a refresh cycle. By using open source software and minimal hardware modifications these PCs can be configured into a cluster that approaches the performance of state-of-the-art cluster computers. In this paper, we present the system setup, configuration, and validation of the WOPPR cluster computer.

Acknowledgments

The Author would like to thank the following people:

Professors **Erkan Tüzel** and **Xinming Huang**, for their continual guidance and advice in their respective roles as the PHY and ECE department advisors for this project.

Robert Lowry and Paul Vasiliadis for their work on the design and fabrication of the rack enclosure and for their contributions to Chapter 4 of this project.

The WPI Computing and Communications Center and the High Performance Computing Group for their support and assistance in gathering all of the initial components for the cluster.

Table of Contents

Table of Figures

Table of Equations

Executive Summary

The purpose of this project was to use 'outdated' commodity PCs to build a cluster computer capable of being used for computationally intensive research. Professor Tüzel of the Physics Department had the need of a cluster computer capable of running multiple processes simultaneously to run gliding microtubule assay simulations. He brought about the idea of using some of the older computer equipment scheduled for recycling to build a cluster computer.

With the growing number of PCs and other electronics steadily filling landfill areas, any opportunity to prolong the lifecycle of these components should be taken. WPI operates on a rough 3- 5 year refresh cycle of their computer components. This provided an opportunity to take a group of 4 year old PCs to use in this project.

The first step in deciding the feasibility of using these computers was to do an energy cost comparison with several comparable computers from mainstream suppliers. After balancing the configurations to get the most even comparison, the older components were tested and compared to the state-of-the-art computers. After collecting data and factoring in the initial cost of the new equipment, it was seen that even though the electrical cost of the reuse cluster was higher on a month-to-month basis, the reuse cluster still managed to be the cheaper alternative within the 3-to-5 year refresh cycle.

The WOPPR cluster was built using 10 old Dell GX620 computers for the nodes, and an 11th for the frontend. For space considerations, an old Compaq server rack was gutted and modified to hold a rack made up of the 10 motherboards and power supplies in a vertical configuration. There is room to expand the cluster in the Compaq rack by adding additional drawer configurations. The rack could hold as many as 40 computers of

the same motherboard form factor. Airflow testing and temperature monitoring were conducted to ensure that the rack design was adequate for the cooling of the cluster.

Rocks Cluster software was chosen as the operating system for the cluster. This open source software is designed to ensure that customized distributions for individual nodes of the cluster could be automatically maintained. The software accomplishes this by making the complete OS installation on a node the default management tool. The Rocks software is a specialized Linux distribution designed from Red Hat Linux 5.1. Third party rolls have been added to add functionality to the system for security, message passing, administration, job control and scheduling, and various other functions.

The project documents the detailed setup and configuration of the software. This includes the Rocks software and additional rolls as well as the software installed for testing purposes. The software for testing consisted of HPL which is a portable implementation of the popular Linpack benchmarks and is used to benchmark the Top 500 Supercomputers of the world.

The testing performed on the WOPPR consisted of 3 parts. The first being a full complement of tests on the entire 10 node cluster using HPL. The second part was completed after replacing the frontend computer for the cluster. This part concentrated on testing only 8 of the nodes in order to compare to the 8 core testing of the Apple Xserve machine. The third part of the testing involved a real world application written by Professor Tüzel for gliding microtubule assay simulation.

HPL and Assay testing was completed on an Apple Xserve computer for comparison. The HPL testing showed that this type of cluster is comparable to the newer processors and architecture used by the Apple Xserve. The efficiency is lower but is

dependent on variations in tuning and environment differences such as the Linear Algebra system used and the version of MPI used. The results of the Assay testing clearly showed that for single thread operations of the WOPPR cluster it is superior to the Xserve. However, when using hyperthreading to run two processes per core, the WOPPR was a distant second to the Xserve performance.

The project proved that using recycled computers to build this type of cluster is a viable option for the older equipment depending on what type of programming needs to be run. Multiple instances of single - process, distributed - memory programming would use the cluster to its full potential. Based on the results of this project, we recommend the following:

- From a cost savings perspective, we recommend building a Rocks cluster from "outdated" computers being recycled as opposed to buying a new cluster computer. The cost savings is realized in the equipment as well as the open source software used for the cluster.
- From an applications perspective, we recommend this type of cluster for any computational research where the programming is to be developed as part of the research. This type of low cost cluster is well suited to research and development environments.
- From the recycling perspective, this cluster configuration is highly recommended for its flexibility. Rocks allows non-heterogeneous mixtures of computer equipment to be added to a single cluster and provides the tools, such as Sun Grid Engine, to take advantage of the different groups of computers added.

Chapter 1: Introduction

This project explores the use of outdated computer components to build a cluster computer that is capable of efficiently running parallel and serial jobs for any computational research. The concept of cluster computing has been around since the 1950's with the SAGE cluster built for NORAD under an Air Force contract with IBM. Later technological advancements continued to contribute to the idea and in the late 1980's there were some notable uses of clusters such as a computational cluster of 160 workstations used by the NSA[1]. In 1994, the first Beowulf commodity type cluster was developed and built at NASA's Goddard Space Flight Center using 16 100MHz Intel 80486 based PCs connected by 10-Mbps Ethernet LAN[2]. The use of commodity cluster computers for computational research has continued to grow since then and many of the Top 500 computers today are some form of cluster design[3]. Commodity clusters are normally built of a homogeneous group of computers or *nodes* and can be configured in several different architectural methods such as high availability clusters, high performance clusters, load balancing cluster, and more.

The concept of "outdated computers" is typically defined by the standard government or corporate refresh cycle of 3 to 5 years. As computers are cycled out of use, they are subject to re-purposing, recycling, or disposal. As more and more computer equipment is being cycled out of use and slated for possible disposal, it is growing ever more important to find alternative uses for them.

Even with the growing popularity of commodity cluster computing, the actual use of clusters has stayed mainly within the realm of academia and enthusiasts. This could be contributed to the steep learning curve required in order to configure, administer, and

maintain a cluster computing environment. The Rocks Group of the University of California has been working on a specialized Linux distribution designed for the use on cluster computer since May 2000, with the underlying concept of making it easier to deploy, manage, scale, and upgrade clusters.

For this project, the Rocks software was chosen as the Operating System. The components for the cluster itself were obtained from a large batch of computers that WPI had removed from the campus labs and common areas and had determined to have no further value and scheduled for recycling.

This project will focus on the feasibility of reusing outdated computers at WPI to form a cluster computer, named WOPPR, for use in research applications which are computationally intensive. This idea was originally presented by a new faculty member, Professor Erkan Tüzel from the Department of Physics. Professor Tüzel brought his research in coarse-pained modeling of complex fluids and living cells to WPI in 2009. As part of his research, he frequently needs the use of distributed-memory, multiple-processor computing resources. He had an idea to reuse the computer resources that were periodically being cycled out of use from his own department, as well as those from the entire campus.

This project encompassed the assembly, configuration, tuning, benchmarking, and evaluation using programming involved in research on the dynamics of biopolymers and their interactions with molecular motors. The testing of the cluster is accomplished with industry standard benchmarks and methodology for determining maximum performance. Other factors included in the test plan were power consumption data and thermal loading

characteristics. For comparison, a top of the line Apple Xserve was included in the testing as well as a dual Xeon server built on-site.

This report discusses the details of this project through conception, construction, testing, and evaluation.

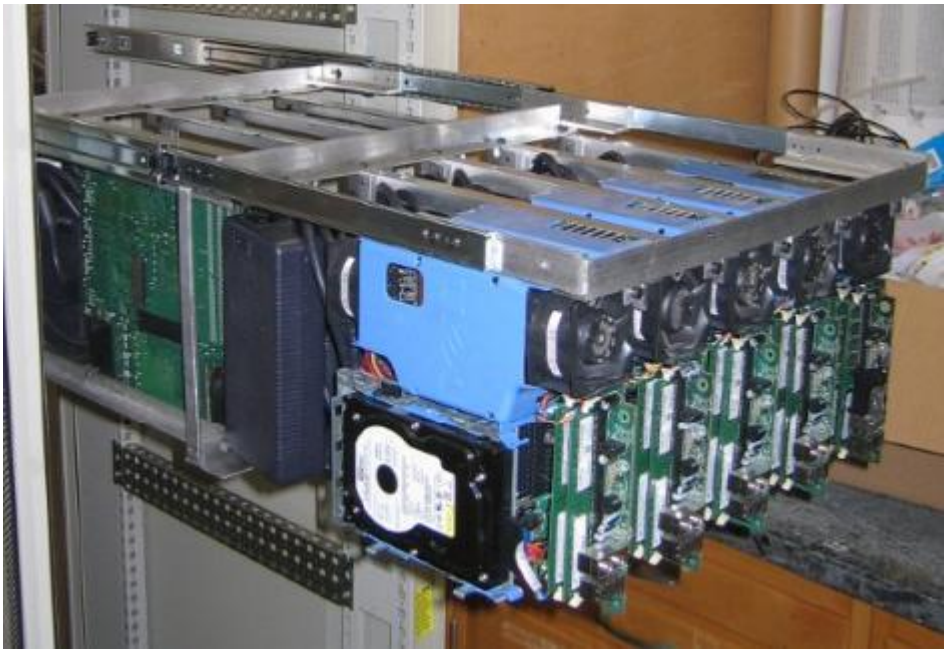


Figure 1 The WOPPR cluster.

Chapter 2: Computer Refresh, Reuse, and Recycling

All organizations that have large quantities of PCs must grapple with the reality of periodically refreshing their equipment. IT departments are consistently tasked with reducing costs and one of the major ways to influence their costs is the PC refresh cycle an organization chooses to use. While some organizations try to push out their pc acquisitions in an attempt to reduce their present budget, the overall cost tends to increase over time after a certain point is reached in the life cycle of computers. This increase is due to items such as IT Help Desk Support, patch support, out-of-warranty repairs, onsite support, and others. In a study presented by Intel Corporation in 2004 the Total Cost of Ownership (TCO) was shown to increase after 3 years as in Figure 2.

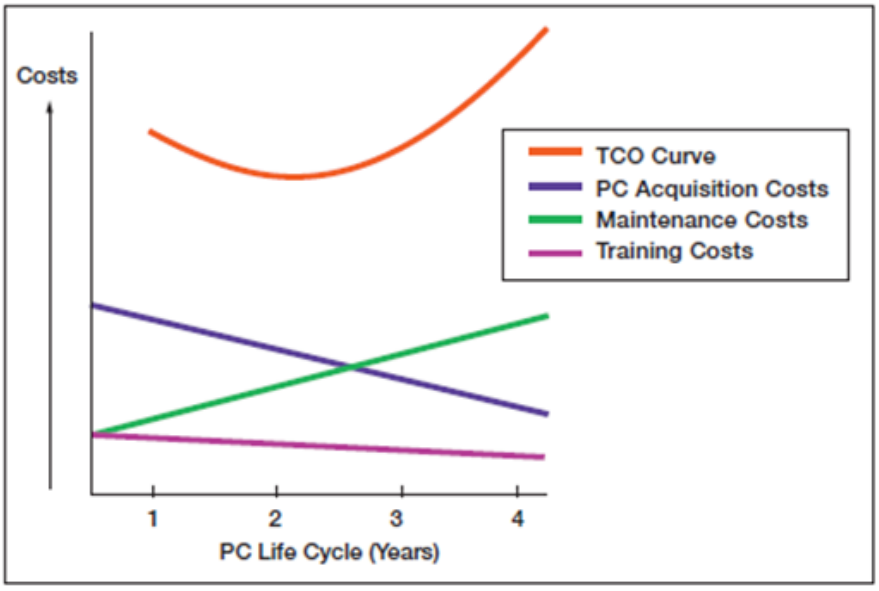


Figure 2 Total cost of ownership based on the PC life cycle with three cost variables [4].

The two main categories of disposition for old computer equipment are disposal and recycling. Recycling normally consists of several options including the reuse of components, and the stripping and segregating of materials for physical recycling and materials reclamation. In June of 2008 the number of installed PCs worldwide was estimated at over 1 billion with a projected increase of 12% per year. This would put the total installed base of personal computers over 2 billion by 2014[5]. The installed base is the number of personal computers in use as opposed to the number of computers shipped. With worldwide PC shipments of over 306 million in 2009, and more than 166 million in the first half of 2010, there is a greater than 30% potential annual addition to the installed base [6]. The difference between the increment of installed base and the units shipped being the amount of computers that are taken out of service. This leaves approximately 200 million computers worldwide being disposed of through recycling in the next year. In 2007 there were approximately 99,000 tons of computers collected for recycling with another 441,000 tons of computers making their way into landfills [7].

In addition to the raw material waste, there are large amounts of hazardous materials such as lead, cadmium, and mercury which are released by these discarded computers, which are harmful to the environment [8].

One major factor which contributes to the low reuse rate of computers, aside from the corporate refresh cycle mentioned earlier, is the Moore's law, which states that the number of transistors on a chip doubles about every two years[9]. With the increases in computing power available the software application design follows suit and pushes the boundaries of performance available. That being said, there is still a use for outdated components. Computers that may not be useful in applications requiring cutting edge

technology and bleeding-edge speed may still be useful for other purposes. A computer that has a processor 4 to 5 years old is still well enough equipped for web browsing, word processing, and other basic computing tasks and when configured correctly these could be used in cluster computers.

For a company or other large business entity such as a school or other organization which has large numbers of computers being cycled out of use, the cost of recycling can be high. Most states in the U.S. have enacted bans preventing the disposal of CRT monitors and electronics equipment in landfills. This limits the disposal options for the components to some form of recycling. This could be manufacturer take-back programs, reuse programs, donations, stripping and segregation of components for physical hazardous waste recycling or disposal, or in some cases just maintaining the items in storage (although this last is only a delaying tactic, not a solution).

Some manufacturers and retailers offer recycling and reuse programs. Dell, HP, Intel, LG, Motorola, AT&T, and Best Buy are just a few of the larger companies that offer comprehensive programs to take back consumer electronics, computers, cell phones, and CRTs for recycling and reuse programs[10].

2.1 WPI Recycling and Reuse

There are three main entities within WPI that handle the majority of the purchasing, disposition, and recycling of computer components. The three Departments are the Computing and Communications Center (CCC), Electrical and Computer Engineering Department (ECE), and Computer Science Department (CS). CCC manages approximately 85% of the general PC assets on campus while the ECE (with the exception of one research lab) and CS Departments manage the purchasing and disposition of PC assets for their own respective departments.

CCC Assets

The PC assets that are managed by the CCC are tracked annually for turnover. There is a rough 3 year cycle that is monitored for groups of computers with the high demand areas such as labs taking priority over lower demand areas such as lounges and common area computing. This tracking allows CCC to plan for whole area turnover as a group of computers age.

When a group of computers are cycled out of use in an area they are delivered to CCC for disposition. Computers that are functional and evaluated to still be of use at WPI's present level of software are cleaned and refreshed and then reused in one of the many non-primary roles around campus. Computers that are evaluated as being functional but too limited at WPI's present level of software are cleaned and wiped before being scheduled for project or community donations. The computers that are physically damaged and unusable are sent to the Facilities Department for physical recycling.

Chapter 3: Energy Cost Analysis for Cluster Computers

One of the aspects used to determine the feasibility of building a cluster of "retired" computers involves an energy consumption comparison with newer state-of-the-art cluster computers. This comparison is performed to calculate the relative expense involved in using the older hardware compared to the newer and more energy efficient components used in the new cluster computers. The two state-of-the-art computers being compared to the WOPPR cluster are the Apple Xserve and the Dell Poweredge R610.

3.1 Systems Used for Comparison

1. Computer Reuse Cluster (WOPPR)

   The reuse cluster configuration used for comparison is composed of GX620 small form factor motherboards, their associated power supplies, hard drives, and fans. The general specifications are listed in Table 1. The Pentium 4's used for comparison are 3.4GHz processors with an 800MHz bus speed and a 2048K L2 cache. The cluster has 10 nodes with the same configuration but for comparison purposes, only 8 nodes are used.

Table 1 General Specification for GX620 Computer

• Pentium 4 640 Prescott Processor, 3.4GHz
• 220W Power Supply
• 2GB of 533MHz SDRAM
• Hard Drives,80G,S2,7.2K,9G 3.5,WD-UNIC

2. Apple Xserve

   The Apple Xserve configuration is shown in Table 2.

Table 2 General Specifications for Apple Xserve Computer

• Two 2.93GHz Quad-Core Intel Xeon X5570 (Quad Core)
• 3GB (3x1GB)
• 160GB Serial ATA ADM @ 7200-rpm
• 8x SuperDrive DL (DVD±R DL/DVD±RW/CD-RW)
• NVIDIA GeForce GT 120 256MB Graphic Card
• Single 750W Power Supply

3. Dell PowerEdge R610

   The Dell PowerEdge R610 configuration is shown in Table 3. The Dell R610 uses the same processor as the Apple server.

Table 3 General Specifications for Dell PowerEdge R610 Computer

• Two Intel® Xeon® X5570(Quad Core), 2.93Ghz, 8M Cache,Turbo, HT, 1333MHz Max Mem
• 4GB Memory (4x1GB), 1066MHz Single Ranked UDIMMs for 2 Processors, Adv ECC
• No Operating System
• 73GB 10K RPM Serial-Attach SCSI 2.5" Hot Plug Hard Drive
• High Output Power Supply, Redundant, 717W

3.2 Power Data

1. Reuse cluster Load testing was completed on several nodes. The power usage is shown in Table 5. Also noted, the highest peak instantaneous amperage is 1.17A. By using the highest value (peak) we can calculate the (worst case) maximum power usage for the cluster, shown in Table 4.

Table 4 Maximum Reuse Cluster Power Usage

Peak current during 100% load = 1.17A
Line voltage = 114.8VAC
Power usage per node = 134.3watts
Cluster power usage = 1075watts

Table 5 Average Current of Reuse Cluster nodes

Usage	Min [Amps]	Max [Amps]
off, unplugged	0	0
off, plugged in	0.03	0.06
booting	0.58	0.98
idling	0.49	n/a
100% CPU	1.01	1.09

2. Apple Xserve power consumption benchmark data was provided by running SPECpower_ssj™2008[11] and is shown below in Table 6. The cost for the system using this configuration is $5,444 on 4/28/2010.

Table 6 Apple Xserve Power

Load	Power
Idle	173W
100% Full Load	334W

3. Dell does not publish their power data so a 50% capacity on the 717W power supply at 115V was used for comparison, see Table 7. The cost for the system as of 1/26/2010 is $4,930 (not including an operating system).

Table 7 Dell PowerEdge R610 Power

Load	Power
100% Full Load (estimated)	359 W

3.3 Total Cost Comparison

Table 8 Full Load Values

Reuse Cluster	1075W
Xserve	334W
R610	359W

Table 8 summarizes the power consumption of these three computers. The Reuse Cluster will use 321% *more* power than the Apple Xserve and 300% *more* power than the Dell R610. However, this is not the sole factor in the energy comparison as the actual cost of the clusters in operation over time must be taken into consideration as well as the purchase price.

The following assumptions are taken in order to perform the comparison:

1. 80% average operation, 24 hrs a day, 365 days a year.
2. Electricity cost of 14.17 cents/kwh[12].
3. The cost of the Dell and Apple clusters is an unrecoverable expense taken at time (0).
4. The cost of the recycled computers is negligible since they are retired from WPI lab use.

The total cost, including the new equipment procurement cost, and the energy consumption cost, is compared in Figure 3.

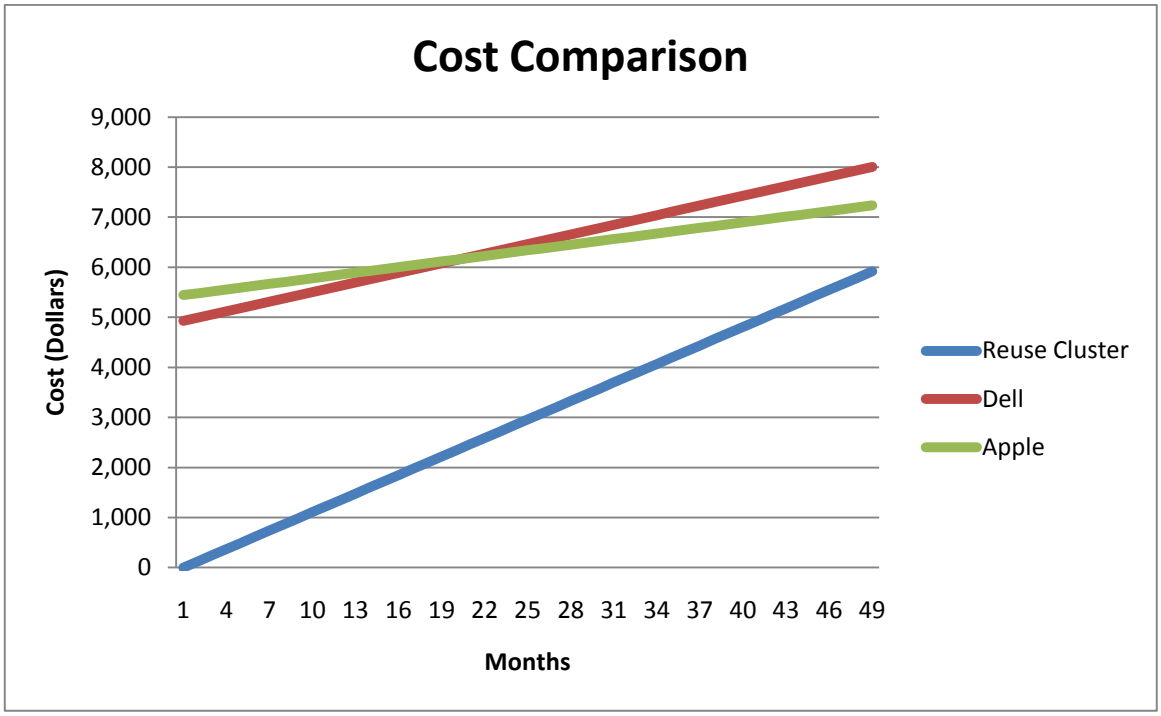


Figure 3 Cost comparison based on initial cost of clusters and the projected average energy usage by month.

3.4 Conclusions

Based on a set of assumptions and projected average energy use, we can see that the Reuse Cluster is more cost effective than the Dell and the Apple Xserve within a normal 36-48 month refresh cycle. After approximately 85 months (not shown), the better energy efficiency of the Dell and Apple clusters would start to outweigh the lack of initial cost involved in the Reuse Cluster. But, with the refresh cycle happening every 36 to 48 months, newer and more powerful and energy efficient computers will be added to the cluster, replacing the older less efficient PCs in the process. This reduction in energy usage coupled with increased performance would have the effect of periodically causing an inflection point in the Reuse Cluster graph followed by a reduced slope. This in turn would move the convergence points with the Dell and Apple machines farther out.

There are some possible disadvantages to using reuse computers for a cluster.

- Space considerations. New server nodes come standard with dual quad-core processors or the newest dual hexacore servers. This allows you to buy one server node with either 8 or 12 cores instead of putting together multiple older PCs.
- Setup and maintenance costs. With a new server, there is often service contract and setup assistance from the vendor. With an older reuse cluster, the responsibility of setup and maintenance rests solely on your IT staff.
- Software programming and maintenance. With this reuse cluster we chose to use an open source solution for the OS and support programs. This does require a more hands-on approach than getting the newest Windows or Apples software installed.

Chapter 4: Building the WOPPR Cluster with Used Computers

4.1 Cluster Computer Architecture

Cluster computing is not a new area of computing. Since the 1950's there has been an interest in cluster computing and a corresponding interest in grouping computer resources in order to facilitate the simultaneous processing of parallel code. A cluster computer is a grouping of computers normally connected by some sort of fast local network so as to make the cluster perform as a single entity. As mentioned earlier, in 1954 the IBM Corporation built the SAGE cluster for the U.S. Air Force as a semi-automated air defense system. By 1961 there were more than 20 operational SAGE sites in the United States, and all of them were remote linked in a computer-to-computer network. While this is an extreme predecessor to today's clusters, and certainly not a commodity cluster, it was one of the first to embody the idea of operating interconnected systems across a fast network. Today there are several different types of cluster architectures each offering different advantages. The three major types of clusters are the High Availability Cluster, Load-Balancing Cluster, and High Performance Cluster.

High Availability Clusters are normally designed to ensure constant availability of the server by the use of redundant nodes. This eliminates the chance of down time due to a single point failure. Load Balancing Clusters use devoted frontend nodes to route work to all of the other available nodes. This provides for a load balancing across the available interconnected nodes thereby increasing the efficiency of the system. High Performance Clusters (HPC) or High Performance Cluster Computing uses clusters of relatively

inexpensive yet powerful computers to solve difficult computational problems. They are designed to exploit the parallel processing power of multiple nodes and normally require that the nodes be able to communicate with each other during processing.

Cluster computer programming is normally divided into an architecture model based on the relationship of the programming to the data that the programming will interface with. This has led to 4 well known models: the Single Instruction Single Data type (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD) type, and the Multiple Instruction Multiple Data type (MIMD).

In the Single Instruction Single Data (SISD) model the cluster uses each processor to execute a single instruction stream on a single memory. In the Single Instruction Multiple Data (SIMD) model each processor will execute a single instruction stream on multiple data in memory simultaneously. In the Multiple Instruction Single Data (MISD) model, each of the processors will execute different instruction streams on the same single data in memory. This is a type of parallel computing architecture but not practically useful. The last type is the most common type of parallel architecture, the Multiple Instruction Multiple Data model (MIMD). In this model, the different processors are individually executing different instruction streams on multiple parts of the data. In MIMD systems each processor is autonomous, meaning that it has its own CPU and ALU and no common clock between processors. These types of machines can be either shared memory or distributed memory machines.

In the shared memory type of machine, the memory is available to all processes and is normally interconnected by a network. This network usually takes the form of either a bus-based architecture or a switch-based architecture. In a bus-based design

multiple processors are simultaneously accessing memory thru the common bus. This could lead to bottlenecks caused by the limitations of the bus. In the switch-based design, the shared memory is normally interconnected with a network. One type of network is the crossbar switching network. This type of interconnect is not feasible for a large number of processors. Another form of interconnection used is the hierarchical connection. This uses a hierarchy of buses to connect and give each processor access to the other processor's memory.

In the distributed memory type of machine, each processor has its own individual memory location and the data shared among processes must be passed from one processor to another as a message. The drawback to this type of design is that of having to connect processors to each other in order to share data. One way to minimize this problem is to only connect each processor to several others. This has given rise to some popular designs such as the hypercube and mesh. The Hypercube scheme requires a minimum of 4 processors and they connect to each other forming a cube. As more processors are added, the total number of nodes must be 2^N, where N is the network diameter and each node connects to N other nodes. The mesh scheme uses the processors in a two-dimensional grid that interconnects each processor to 4 of it's neighbors.

Aside from how a cluster handles the programming in relation to its data, cluster programming models can also be divided by how they handle the cluster's inherent parallelism. There are two main categories of programming. The first category is when serial programming is used to exploit the parallelism of a system. The second is where the programming is explicitly made to run as parallel code.

Pfister[13] established the term SPPS (serial program, parallel subsystem) to describe the common technique of running serial programming in parallel on a cluster. A parallel subsystem allows input to each instance of the serial code and takes each instance of output to deliver to the user. Since this is an example of multiple programs operating on multiple data, this is a MIMD system. The two most common ways to implement SPPS programming are through distributed shared virtual-memory and message passing. Message passing is described in some detail in the section discussing the Rocks Software.

This project will utilize Pentium 4 processors which support Intel's hyperthreading technology and distributed memory. Hyperthreading technology allows the operating system to see two processes in each core which can allow programming to use the cluster nodes as any of the architectures; SISD, SIMD, MISD, or MIMD. The overall cluster is considered Multiple Instruction Multiple Data architecture with distributed memory.

4.2 WOPPR Cluster Design

The original components for this cluster were found after the Physics Department cycled a group of computers out of service. This included 10 Dell GX620 mini-form factor computers and one Dell GX620 desktop computer. These particular computers had been in operation within the Physics Department since late 2005 to early 2006. The original specifications for each PC are listed in Table 9 below and a more detailed listing of the BIOS data is included in Appendix A. A significant bonus to using standard PCs for a cluster is that no extra enclosure or rack design is really required. Each of the PCs used is originally in its own desktop (or similar) case including heatsinks and fans. A

small cluster could physically be connected as long as you can fit all of the computers into the same area to be connected to a switch. However, as more and more nodes are added, the space and power required could become a limiting factor.

Since this particular cluster was going to be housed in a very small room, we decided to look for a more traditional case to mount the individual nodes inside. The cluster was to be formed with the initial GX620 mini-form factors making up the blades and the desktop providing the Frontend for the cluster. As more computers become available in the future, they can be added to the cluster. To accomplish this, the initial cluster rack design had to incorporate a space fabricated to the dimensions of the mini-form factor, and another space that could accommodate either ATX or micro-ATX form-factors.

We were able to find an enclosure to use for housing the cluster in a pile of metal recycling. The enclosure was an old Compaq rack, and is shown in Figure 4. The internals of the rack had been removed leaving only a shell to work with. The initial design concept for the mini form factor space was to manufacture a horizontal sliding rack to support 10 or 12 of the mini form factor motherboards in a vertical orientation in order to take advantage of the natural airflow path from bottom-to-top in the server rack.

Table 9 Specifications of Computers in the WOPPR Cluster

Node	woppr-0-0	woppr-0-1	woppr-0-2	woppr-0-3	woppr-0-4	woppr -0-5
Processor	Pentium 4	Pentium 4	Pentium 4	Pentium 4	Pentium 4	Pentium 4
Clk Spd	3.4 GHz	3.4 GHz	3.4 GHz	3.4 GHz	3.2GHz	3.4 GHz
Bus Spd	800 MHz	800 MHz	800 MHz	800 MHz	800 MHz	800 MHz
L2 Cache	2 Mb	2 Mb	2 Mb	2 Mb	2 Mb	2 Mb
Memory	2Gb	2Gb	2Gb	2Gb	2Gb	2Gb
Mem Spd	533 MHz	533 MHz	533 MHz	533 MHz	533 MHz	533 MHz
Node	Original Frontend	New Frontend	woppr-0-6	woppr-0-7	woppr-0-8	woppr-0-9
Processor	Pentium 4 Prescott Dt, 640	Core 2 Quad Q8400	Pentium 4	Pentium 4	Pentium 4	Pentium 4
Clk Spd	3.2 GHz	2.66GHz	3.2GHz	3.4 GHz	3.4 GHz	3.4 GHz
Bus Spd	800 MHz	1333 MHz	800 MHz	800 MHz	800 MHz	800 MHz
L2 Cache	2 MB	4096 KB	1Mb	2 Mb	2 Mb	2 Mb
Memory	2 Gb	4 Gb	2Gb	2Gb	2Gb	2Gb
Mem Spd	533 MHz		533 MHz	533 MHz	533 MHz	533 MHz

Figure 4 Rack enclosure.

In order to determine the overall stability of each computer, a full-load stress-test was performed for a minimum of 4 hours using both Prime95[14] and OCCT[15]. Each

of these programs tests overall system stability by using computationally intensive mathematical operations such as Fast Fourier Transforms, and are performed in order to determine the maximum number of mathematical operations per second, as well as to ensure that the computer is safe from overheating even under maximum load. Each computer passed the full-load stress-tests without overheating, and so it was concluded that the stock cooling system would be adequate for further use.

After the initial stress-testing of each computer in its case was successful, the computers were removed from their cases in order to give a better understanding of the physical layout of each individual component inside. This was necessary because in order to build a custom enclosure with maximized packing efficiency, the components would need to be re-adjusted and moved around. Upon removing the computers from their cases, it was noticed that each computer had varying levels of dust and grime buildup. This buildup of particles reduced the overall airflow through each case and meant that there was an excess buildup of heat around the central processing unit, which can be damaging and can shorten the lifespan of the computer.

Once each computer was removed from its case, compressed air was used to remove the dust and grime buildup and stress-tests were once again performed using Prime95 and OCCT. Qualitative testing showed that removing the dust led to increased airflow through the computer, which in turn led to lower temperatures under full-load.

Since the air would be flowing vertically, computer motherboards would also remain vertical, as would their external power supplies. The mother board was approximately square, and slightly shorter than the longest dimension of the power supply. The case had considerable depth, which easily facilitated two parallel rows of vertical computers,

so the depth was partitioned into two sections. The width was slightly less than six times the width of the former computer cases. Six units side by side would be too cramped to cool properly, but five would allow for reasonable spacing. The maximum usable footprint was divided into a grid of two rows and five columns.

To allow for ease of access, the entire system would be mounted on drawer slides so it could be partially removed from the case. A pair of 24 inch drawer slides was installed into the existing server mounting holes, using the original case screws (Figure 5).

Figure 5 Picture of drawer slide.

Three pieces of one inch angle aluminum were attached to each of these slides at the center to extend the reach of the ends. To these were attached two pairs of thin walled one inch angle aluminum to form the top and bottom of each row. All these pieces were attached together with 10-24x1/4 button head machine screws. The maximum usable width was measured and divided into five units, of the maximum height of each computer assembly.

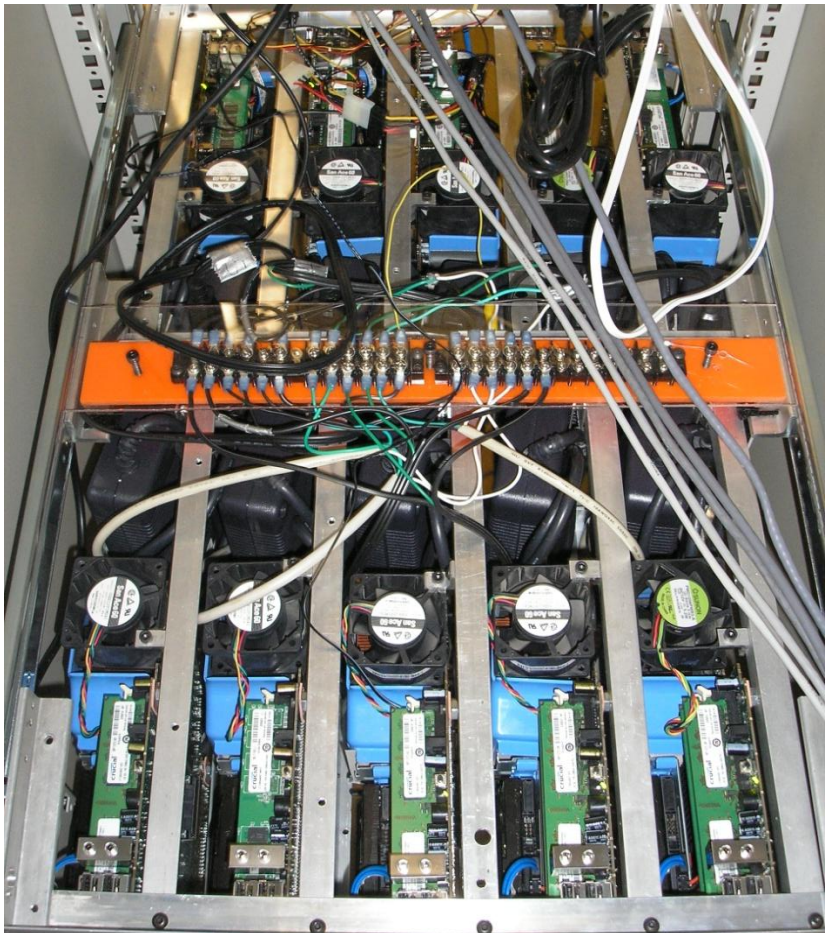


Figure 6: Image of the drawer-rack.

When the final height of the computer assemblies was determined, two aluminum strips were used to hang a pair of the same row attachments as before. These were only half rails, for they would be counteracting only a bending moment. The full second piece of aluminum was deemed unnecessary weight and expense.

Each computer was removed from its case and stripped to the bare motherboard. First, the CPU heat sink mounting bracket no longer had a back plate to affix to. Strips of tenth inch scrap plastic were cut to a convenient rectangular size, and were fashioned with pairs of #6-32 holes spaced to accept #6-32x5/8 socket head machine screws passing through the mounting bracket.

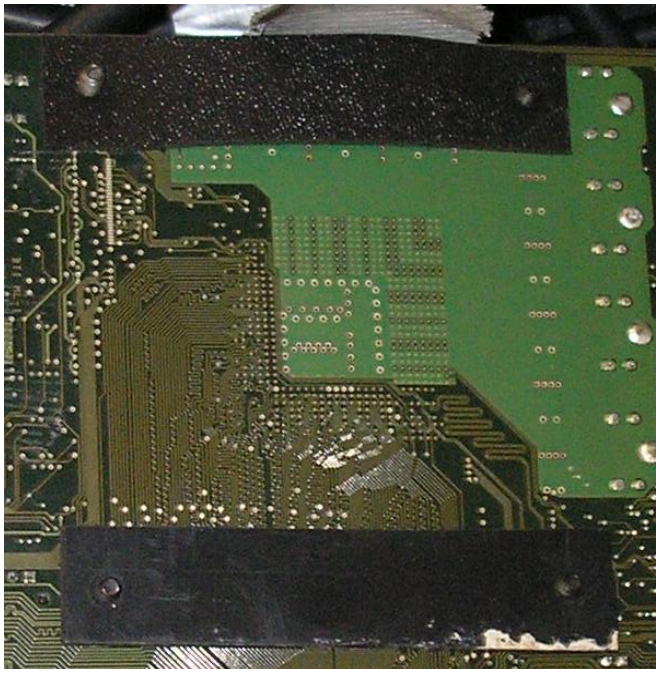


Figure 7: Image of back of computer - black plastic.

Second, the hard drive was mounted aloft over the south-bridge on one inch stand-offs, which formerly threaded into the back plate. These stand-offs were threaded with a #4-40 thread, and were held directly to the motherboard in the same position as before by a pair of nuts.

The one piece motherboard assembly now needed to be hung vertically inside the case to allow for proper airflow. It was decided to orient the wire inputs of the motherboard down for ease of access. The board was mounted on two strips of 3/4" angle aluminum with #6-32x5/8 socket head machine screws, using a $1cm^2$ piece of rubber repair gasket and a 1/4x20 nut as insulation and a spacer, respectively.

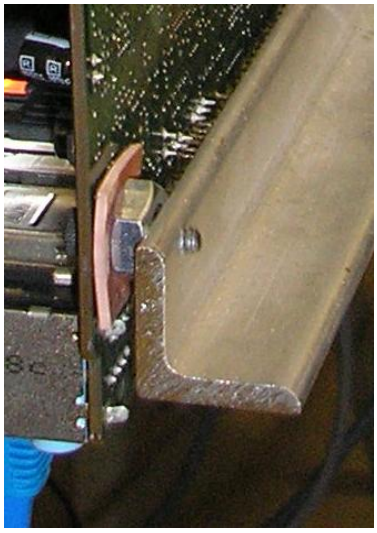


Figure 8: Image of gasket and nuts.

The CPU fans were attached to the rails with half inch lengths of the same aluminum drilled to form angle brackets. The direction of air flow was chosen to be up in order to facilitate the case airflow.

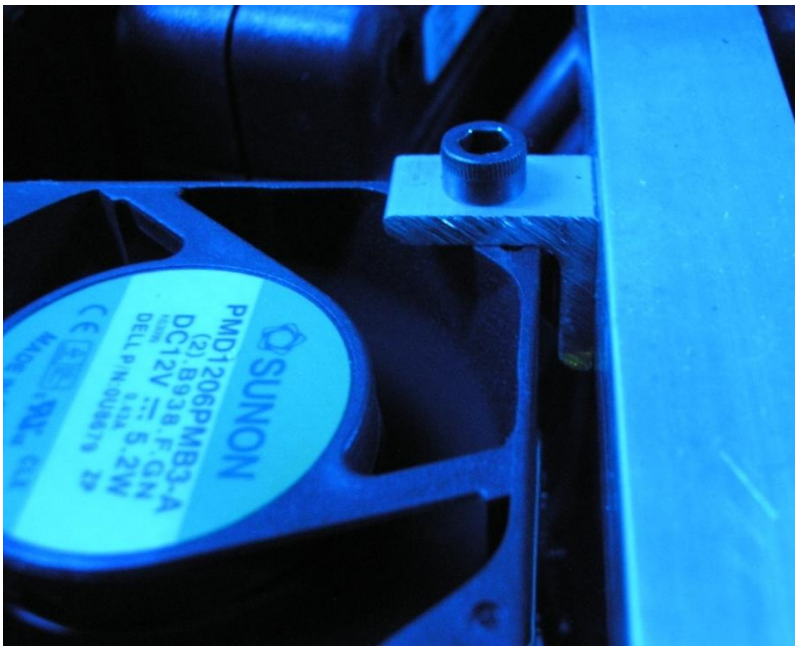


Figure 9: Image of angle bracket.

The last element of the computer was the power supply. Dell® had designed these computers with modular external power supplies. To reduce the overall length of each computer's final shape, the power supply was mounted at a slight angle. This way the effective height would be the same as the CPU heat sink, and allow for a pocket behind the supply for the cord to be stored. The housing of the supply was held together by four specialty screws, which fit into extruded bosses inside the housing. Two of these screws were discarded, and the bosses removed. The two holes chosen were enlarged to allow for #6-32x5/8 socket head machine screws with one #6 washer to pass through and hold the incline brackets to the housing. The remaining screws were used to hold the cover of the supply housing shut. The side on which the supply was mounted was such that the standard power cord port was located upward. Once the cord was coiled and stored behind the power supply housing, the computer blade was completed. This process was repeated for all 10 computers.

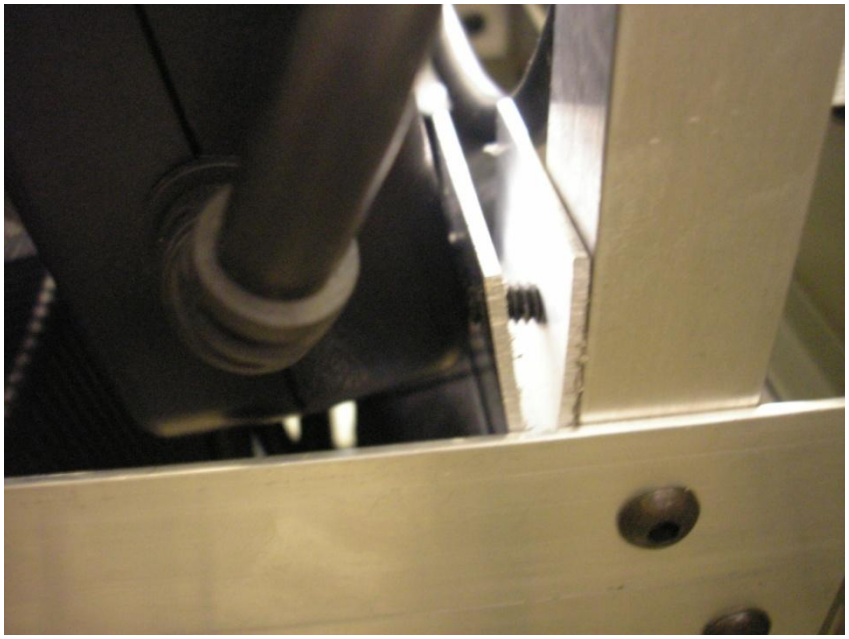


Figure 10: Image of PSU bracket.

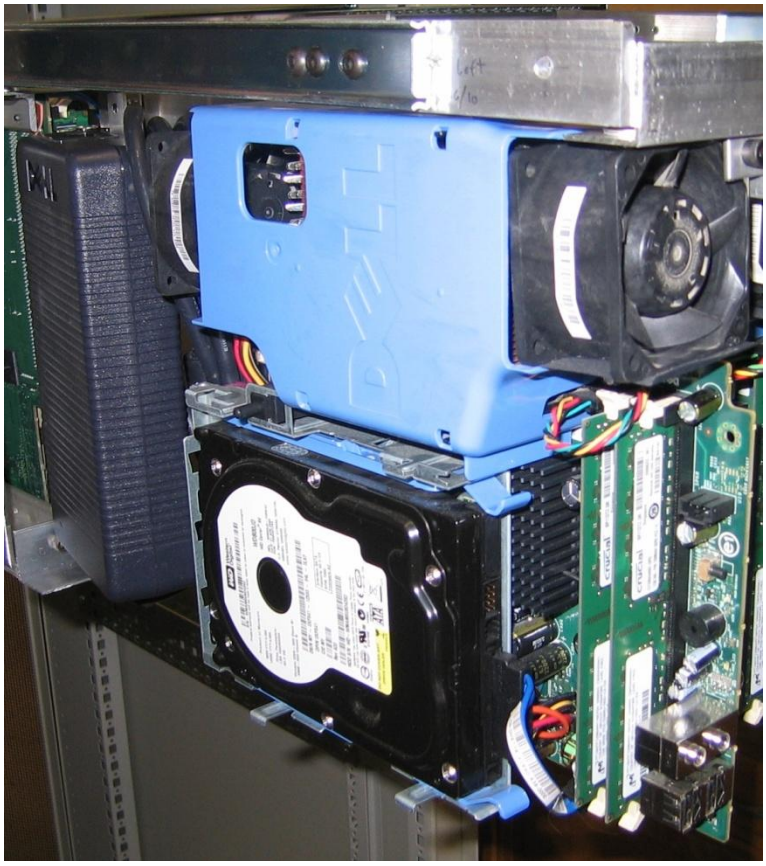


Figure 11: Image of a completed blade.

The nodes were attached to the rack with 10-24x1/4 button head machine screws, two at the ends of the top rail, and one at the end of the bottom rail.

Once all ten computers were brought together, a convenient way to power them all became an issue. Three break-out boards, one for each wire in the power cord, were mounted along the center support between the two groups of blades. The ten power cords were severed at an appropriate length and wire lugs were installed. The boards were mounted on a piece of 1/4 inch acrylic, and covered with a protective sheet of 1/10 inch acrylic. The ten computers were then wired together onto one single power cord.

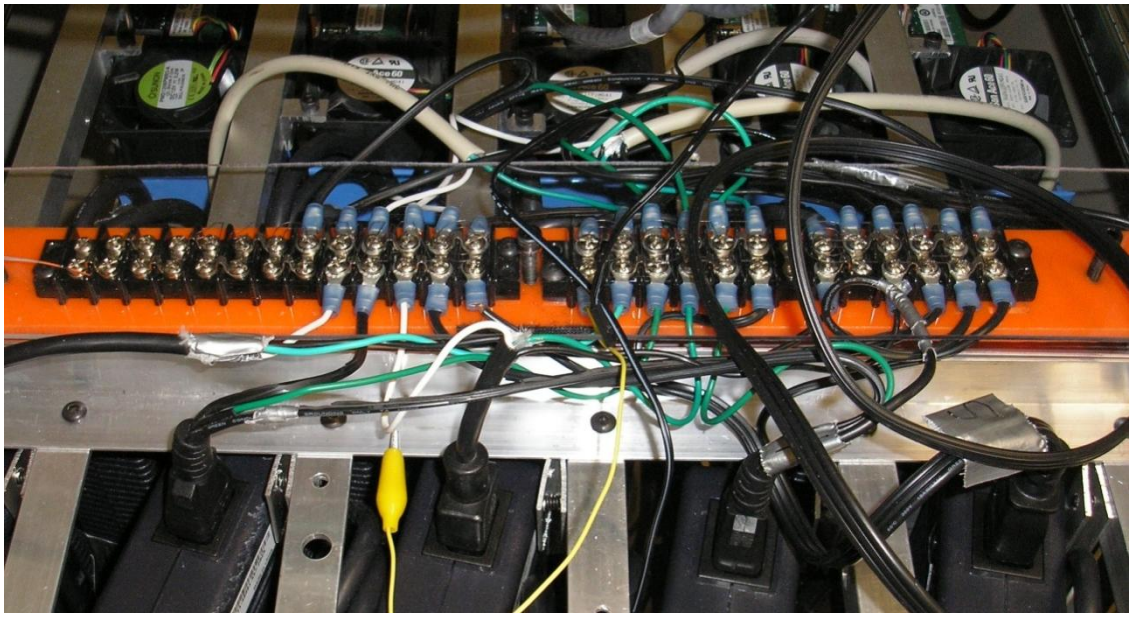


Figure 12: Image of power breakout board.

One computer was tested at several points for the current drawn from the wall. A short extension cord was fashioned with separated wires, and an induction multi-meter was used to measure the current on the hot line as the computer's usage and state were varied. The results are shown in Table 5 above.

Peak instantaneous current draw was 1.17A. After this test was completed, the line voltage was measured at 114.8 VAC. From this, the estimated maximum power draw is 1350W with maximum current 12A.

The first network switch to be installed was also a component rescued from possible recycling. This was a Bay Networks BayStack 350F-HD 10/100 network switch with 24 ports. Unfortunately, the switch made twice the noise and twice the heat as all 10 nodes combined so a replacement had to be purchased. An SMC EZNET-16 10/100Mbps switch was installed above the node rack using the rack mounting brackets provided by the manufacturer. The complete specifications of the switch can be found in Appendix C.

Initially a generic power strip was tie-wrapped to the top of the network switch to provide the required 115vac for the rack. This was subsequently replaced with a permanent rack mounted surge suppressor/power strip with both 15 and 20amp outlets. The specifications of the power strip can be found in Appendix D.

An additional ATX power supply was added near the top of the rack to provide ready connectors for case lighting, temperature monitoring, etc. Temperature monitoring was provided by a digital display with 4 thermocouples attached. The thermocouples are placed as follows: one approx 15" below the blades, one at the top of the cluster rack where the air exhausts, one on node 4 attached to the motherboard, and one next to the power adapter for node 4. After the cluster was moved into its present home and all nodes placed online, a max load was placed on the cluster in order to do an initial temperature monitoring survey. A small serial program running in an infinite loop was loaded on each thread resulting in 100% load for all 20 node threads over a period of 7 days. The temperature results are listed in Table 10.

Table 10 Temperature Monitoring Data

Location	**Average Temperature (°C)**
Top of Rack	30.0
Node 3 CPU Heat Sink	37.2
Node 7 Power Supply	24.8
Bottom of Rack (near floor level)	20.3
Room Temperature (measured at 6' height)	21.8

Chapter 5: Software Descriptions and Installation

This chapter will discuss the software requirements and installation necessary to setup a productive cluster capable of being used for both serial and parallel programming, and the software and setup required to benchmark the system. The main components are the Rocks Cluster distribution of the Linux operating system, a linear algebra system, and the C, C++, and Fortran libraries. The benchmarking setup required the installation of a Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers (HPL). The description and installation of all of these software components will also be discussed below.

5.1 Rocks and Rolls

The Rocks Cluster is available with several rolls that have been formatted specifically for use with the Rocks environment. A roll is a distribution of a program; one or more applications and their associated libraries configured to work with the kernel version of Linux that it was rolled for. The Rolls included in the original build of this cluster are Area51, Ganglia, HPC, pvfs2, SGE, and Web-server. Each of these will be discussed below.

5.1.1 Rocks v5.3

The National Partnership for Advanced Computational Infrastructure (NPACI) at the San Diego Supercomputer Center (SDSC) of the University of California, San Diego (UCSD) began work on NPACI Rocks in 2000 on a grant from the NSF and has continued to make refinements and improvements over the years on further NSF

grants[16][17][18]. The initial philosophy of the group was to develop a toolkit that would simplify the version tracking of the software and also simplify cluster integration[19].

The predominant method at the time for cluster management was a painstaking comparison of configuration across the nodes involved. This was time consuming and impractical as the number of nodes expanded. Their idea was to create a mechanism that built on a popular commercial distribution of Linux and provide a means to ensure that customized distributions for individual nodes could be automatically maintained. They accomplished this by adopting the paradigm of making the complete OS installation on a node the default management tool. This means that whenever there is any doubt as to the nodes integrity or configuration, a complete OS installation can occur in a short period of time to restore the node to a customized default condition. This completely removed the exhaustive comparisons previously required to keep nodes configured and it also allowed for a level of automation that makes management of huge clusters fairly easy and less time consuming. Another key benefit of the ability to create customized distributions for the nodes is that the security patches can be pre-installed in the distribution along with any other changes that define the node or appliance. The different node or appliance types are defined within a machine specific file called a *Kickstart file*.

The Kickstart file is made from a Rocks Kickstart Graph. This is an XML-based tree structure and is used to define the differences between node and appliance types without needlessly duplicating their similarities. The Kickstart File is text-based and contains the descriptions of the software configurations and packages to be installed on

the nodes. This file is user configurable and is used to address all of the questions normally asked during an installation.

The management strategy developed by the NPACI team is designed to simplify management of the cluster and to promote experimentation. All of the software is deployed using the Red Hat Manager (RPM) and designed for scalable services such as DHCP, HTTP, and NFS. The compute nodes use Kickstart's HTTP method to obtain the RPMs from the Frontend for installation. The Dynamic Host configuration Protocol (DHCP) is used for Ethernet configuration of the nodes and the Network File System NFS is used to export all user home directories from the frontend to the compute nodes. The use of RPMs also provides an easy tool for maintaining the software. A tool incorporated into Rocks, called Rocks-Dist, gathers software components that have been installed locally, the 3rd party software that have been installed, and creates a customized distribution which looks just like a Red Hat or CentOS distribution but with more software added. After the initial installation of a node, any subsequent re-installation will not affect any non-root partitions on the nodes. This means that any scratch or user partitions that have been created will be maintained. The default action for nodes on a loss of power is to do a re-install on power-up. The average time for a node re-installation on the WOPPR cluster was measured at 11min 40sec.

The cluster hardware requirements are standardized to conform to other high-performance computing clusters, and are listed in Table 11 below. Also shown below in Figure 14 is a picture of how the cluster should be connected. The cluster uses the Ethernet network for management purposes. A separate specialized network such as

Infiniband, Myrinet, or Gigabit Ethernet can be added for low-latency, high-bandwidth message passing for parallel programs.

Table 11 Cluster Hardware Requirements

Frontend Node	
Hard Disk	30 GB
Memory Capacity	1 GB
Ethernet	2 physical ports
Compute Node	
Hard Disk	30 GB
Memory Capacity	1 GB
Ethernet	1 physical port
BIOS Boot Order	CD, PXE(network Boot), Hard Disk

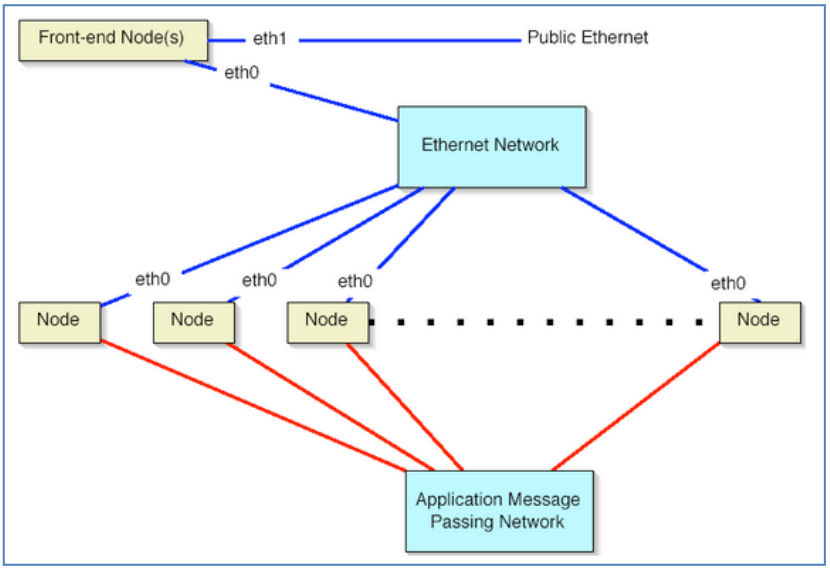


Figure 13 Frontend and compute node setup.

Rocks v5.3 is the operating system used for this cluster project. It is available from the Rocks Clusters website [20] as either several CD images or a single DVD image and comes in either 32-bit or 64-bit versions. The DVD .iso image also comes with the following rolls included, Adaptive Poisson-Boltzmann Solver (apbs), area51, bioinformatics utilities (bio), ganglia, rocks HPC roll (hpc), jumpstart, PVFS2 parallel file system support (pvfs2), Sun Grid Engine (SGE), Torque & Maui job queuing system

(torque), viz, Rocks Web Server roll (web-server), and Xen (xen) for building xen VMs on cluster nodes.

5.1.2 Rocks OS Installation

After making all of the connections between the frontend, nodes, switch, and the internet, the WOPPR was ready for the software installations. The 64-bit Jumbo DVD .iso image had been downloaded from the Rocks website and burned to a DVD[21]. The first thing to check before installing the software is that *eth0* on the frontend must be connected to the Ethernet switch and *eth1* must be connected to the external network. If there are more than two Ethernet connections on the nodes, then you must ensure that *eth0* is connected to the switch.

The minimum requirements to load the OS are the Kernel/Boot CD, Base Roll CD, the Web Server Roll CD, and the two OS Roll CDs. Since we are using the Jumbo DVD image, all the required CDs and extra Rolls are included.

1. Insert the DVD in the frontend and reboot it.
2. After the frontend boots from the DVD the following screen will appear:

Figure 14 Rocks installation splash screen.

Type "build" at the prompt. If this prompt is missed, the installation will continue as if a compute appliance were being installed. The only way to get back is to power down the frontend and restart the process.

3. In this section there are some screens that will not appear if the IP for the cluster was set up as a static IP. If this is the case, then there will not be a DHCP server on the network to answer the DHCP setup requests from the frontend and the user will have to supply the information. Since the WOPPR IP address was set up as static, all the configurations will be shown below.

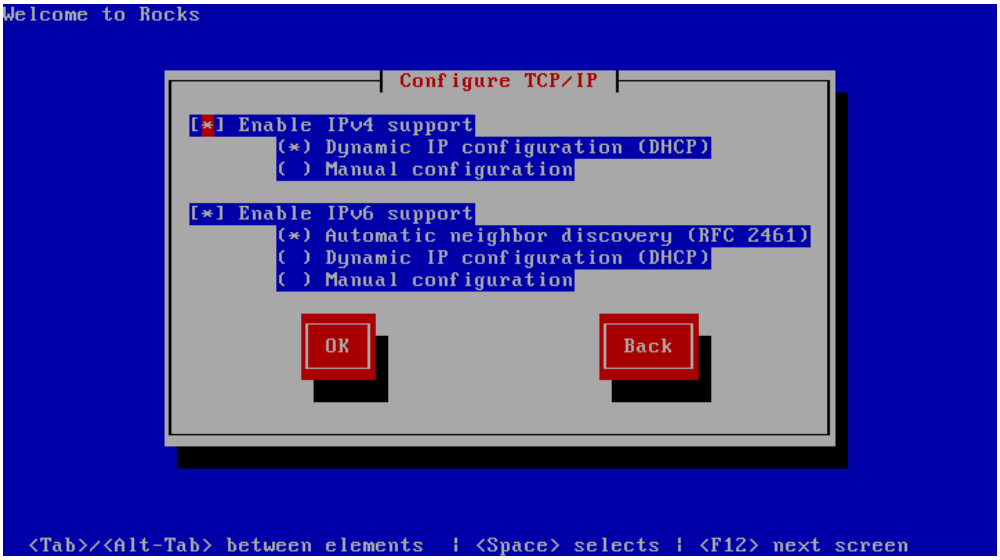


Figure 15 Rocks installation TCP/IP configuration.

This is the first screen to appear. Since we are static, we need to configure everything manually. Enable IPv4 support was selected, and Manual configuration of the IPv4. IPv6 support is disabled. The resulting screen is shown below:

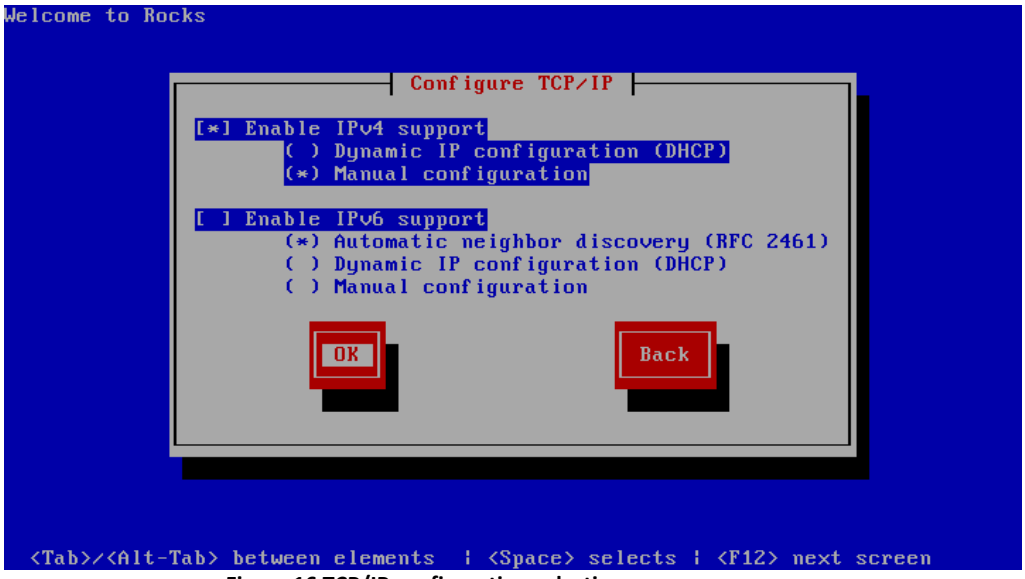


Figure 16 TCP/IP configuration selections.

We select “OK” and the “Manual TCP/IP Configuration” screen will show:

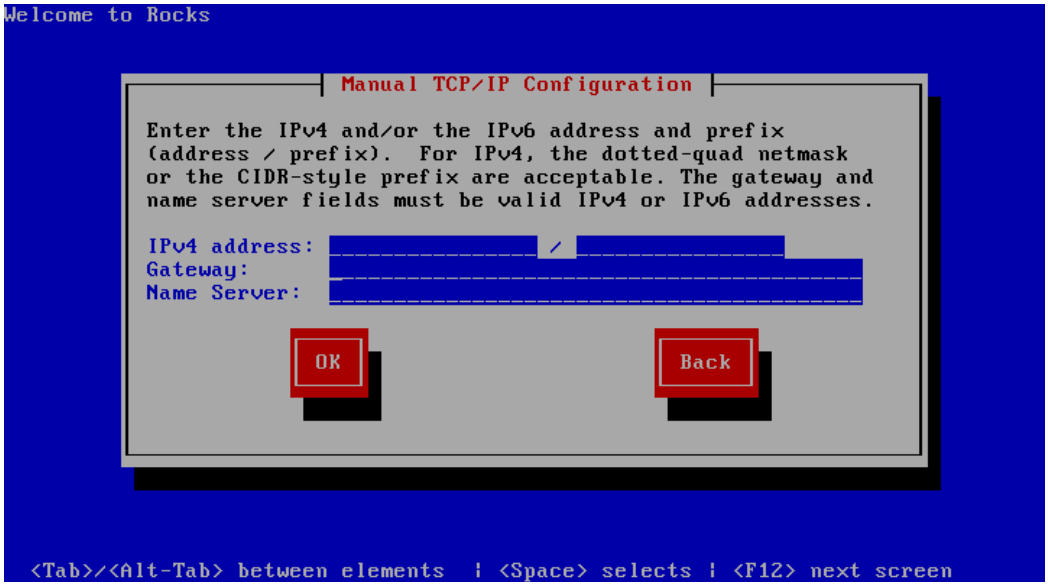


Figure 17 TCP/IP configuration data entry.

The public IP configuration was entered as follows:

- IPv4 address: 130.215.96.65/255.255.255.0
- Gateway: 130.215.96.1
- Name Server: 130.215.32.130, 130.215.39.18, 130.215.36.18

After entering the information and selecting "OK", the "Welcome to Rocks" screen appears.

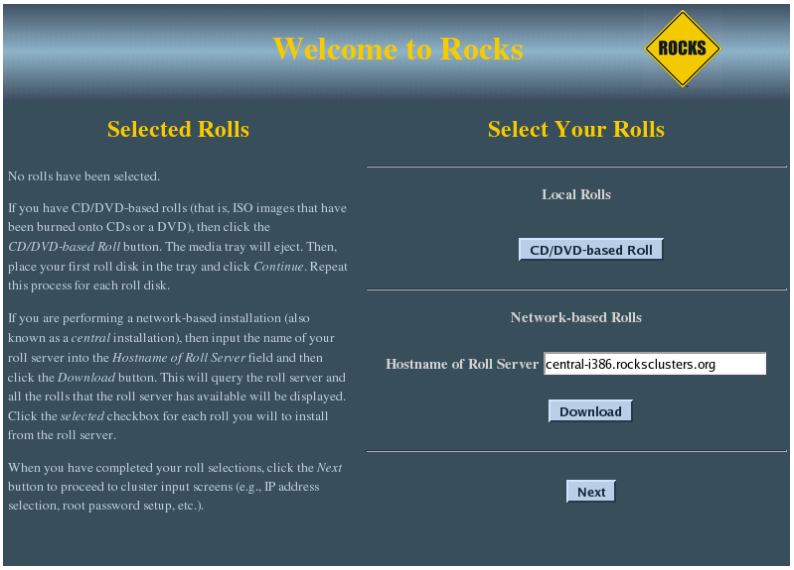


Figure 18 Rocks Roll selection screen.

4. This is the screen we use to select the rolls for installation. There is an option to connect to a server to install the rolls as well as using CD/DVDs. Since we are using the Jumbo DVD we select the CD/DVD button and the CD tray will eject as the following screen appears.

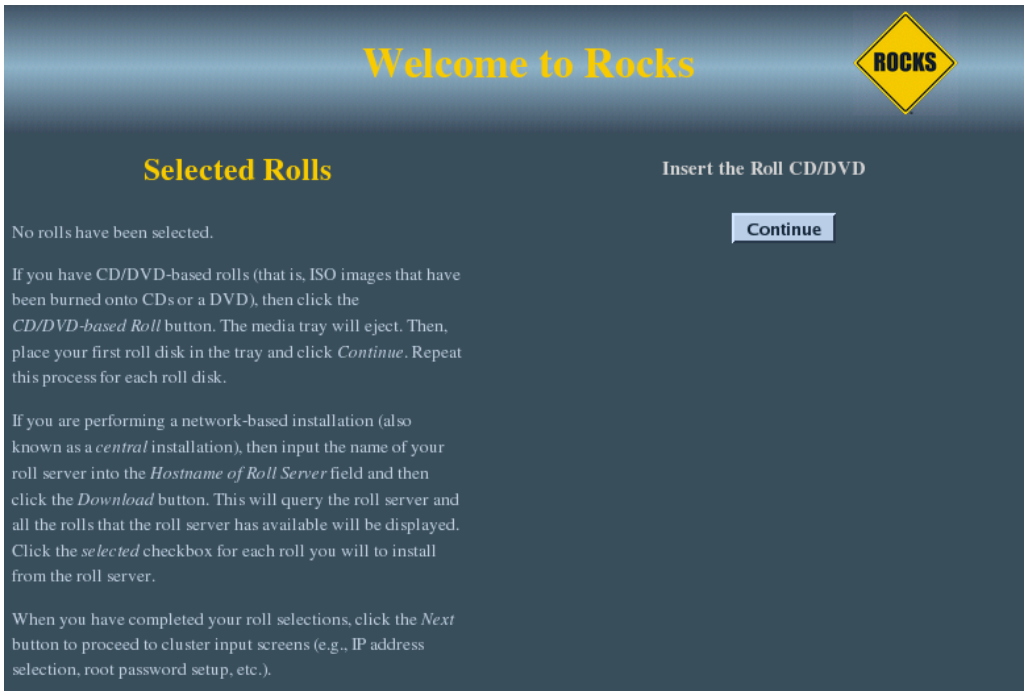


Figure 19 Rocks Roll selection, screen 2.

5. All of the Rolls are on the DVD so the tray is pushed back in and "continue" is selected. The kernel/boot Roll and all other Rolls will be detected and a screen similar to the following will appear:

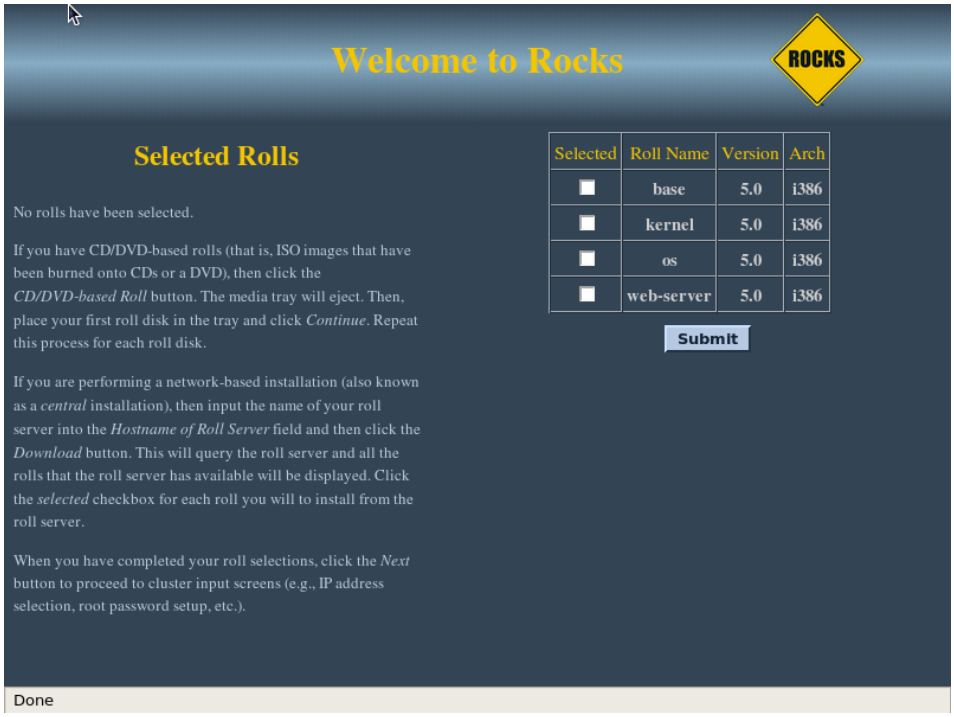


Figure 20 Rocks Roll selection, screen 3.

6. Select the kernel/base, OS, web-server, and all other Rolls to be installed. Then select the "Submit" button. The following screen is the "Cluster Information" screen below:

Figure 21 Rocks cluster information screen

7. The only vital field on this page is the "Fully-Qualified Host Name" field. All of the other fields are optional. The Host Name is written to dozens of files on both the frontend and compute nodes. After the installation it is almost impossible to change this without causing several services such as NFS, SGE, and more to become disabled. The WOPPR information was entered as follows:

Table 12 Rocks Cluster Information

Fully-Qualified Host Name: woppr.wpi.edu Cluster Name: WOPPR Certificate Organization: WPI Certificate Locality: Worcester Certificate State: MA Certificate Country: US Contact: etuzel@wpi.edu URL: http://users.wpi.edu/~etuzel/Welcome.html

After filling in this information and selecting "NEXT" the following screen appears:

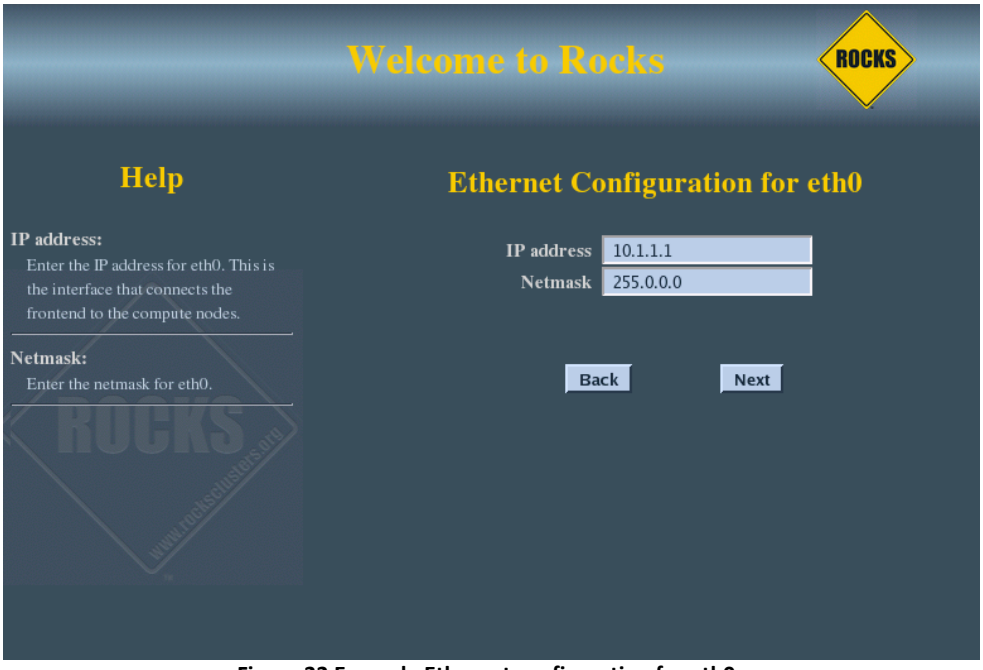


Figure 22 Example Ethernet configuration for eth0.

8. Since this is the setup for the private network between the frontend and the compute nodes, we leave it at the default settings. Continuing to the next screen, we have to set up the external network information.

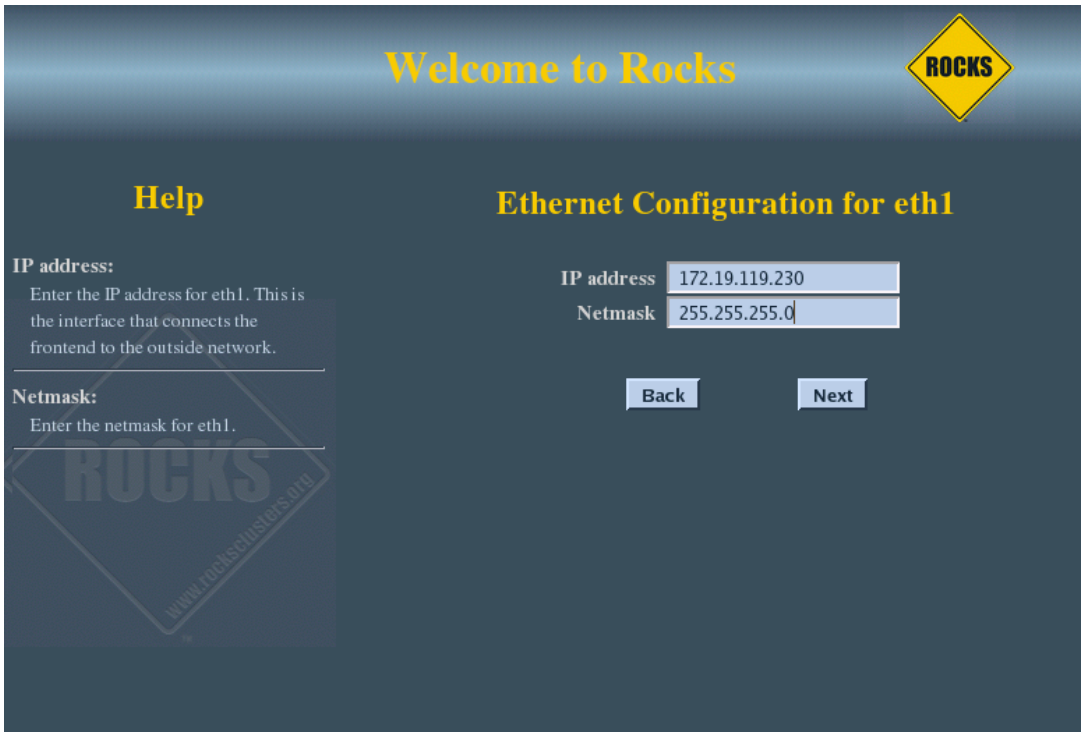


Figure 23 Example eth1 configuration screen.

Our cluster information was entered as follows:

- IPv4 address: 130.215.96.65
- Netmask: 255.255.255.0

Continuing to the next screen is the Gateway and DNS entries in Figure 24.

- Gateway: 130.215.96.1
- DNS Servers: 130.215.32.130, 130.215.39.18, 130.215.36.18

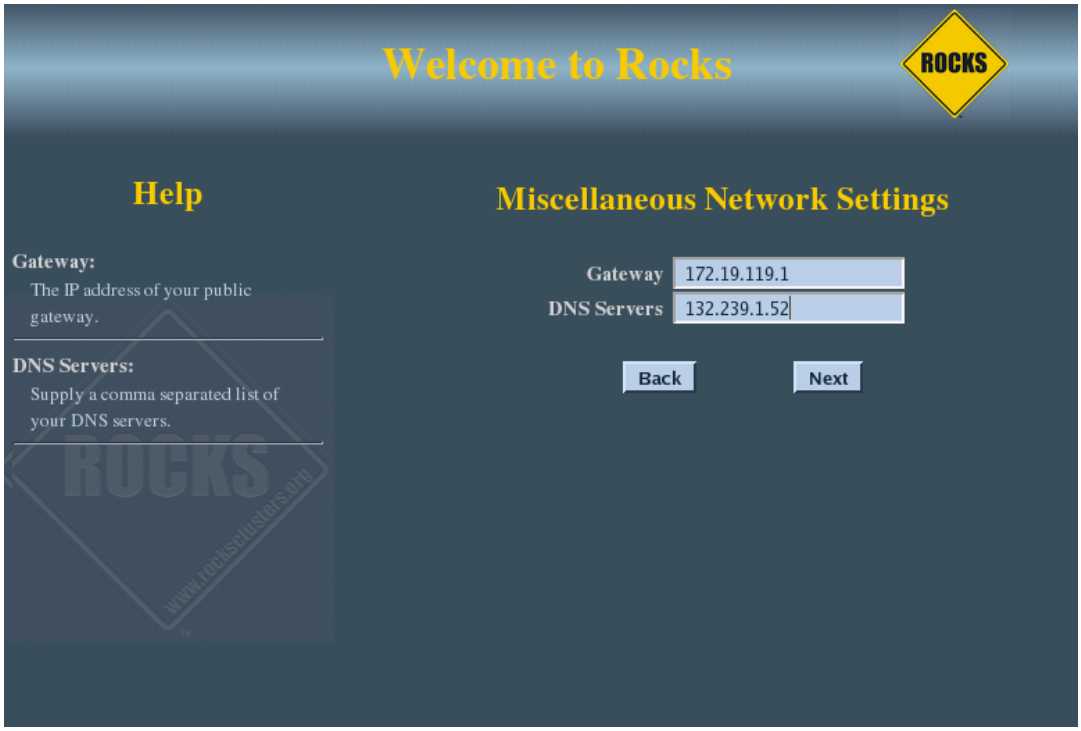


Figure 24 Example gateway/DNS configuration.

9. The WOPPR external information is shown above. Selecting "Next" moves us to the "Root Password" screen. Enter a password and confirm it.
10. The following screen is the time zone and NTP Server. The server can be left at the default "pool.ntp.org".
11. The following screen is the disk partitioning screen shown in Figure 26. If automatic partitioning is selected then Rocks will create default partitions on the frontend drive. These partitions are shown in Table 13.

Table 13 Default Rocks Partitioning

Partition Name	Size
/	16 GB
/var	4 GB
swap	1 GB
/export	the remainder of the root disk

The /export partition is symbolically linked to /state/partition1. The / partition is the root. The /var partition is where all of the services are kept. The /export partition is where the user accounts and directories are stored, also the Rolls. We originally installed the OS with default partitioning on the original frontend. When we changed out the frontends we decided that the default partitioning was too confining, and proceeded with a manual partitioning.

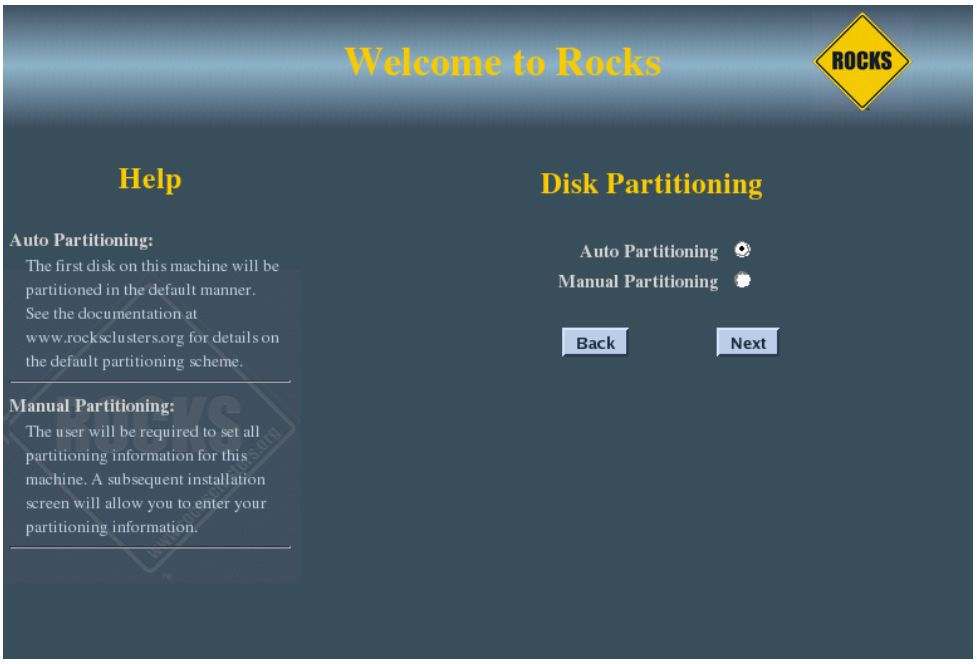


Figure 25 Partitioning selection screen.

12. After selecting Manual Partitioning the Red Hat manual partitioning screen will appear. This is shown in Figure 26. Since this was a new drive there was only free space listed. In order to set up partitions this needs to be deleted first. Then enter "New" and choose the Mount Point, Type, and Size to create a partition. The sizes we selected are shown in Table 14.

Table 14 WOPPR Manual Partitioning

Partition Name	Size
/	100 GB
/var	12 GB
swap	2 GB
/export	the remainder 1TB

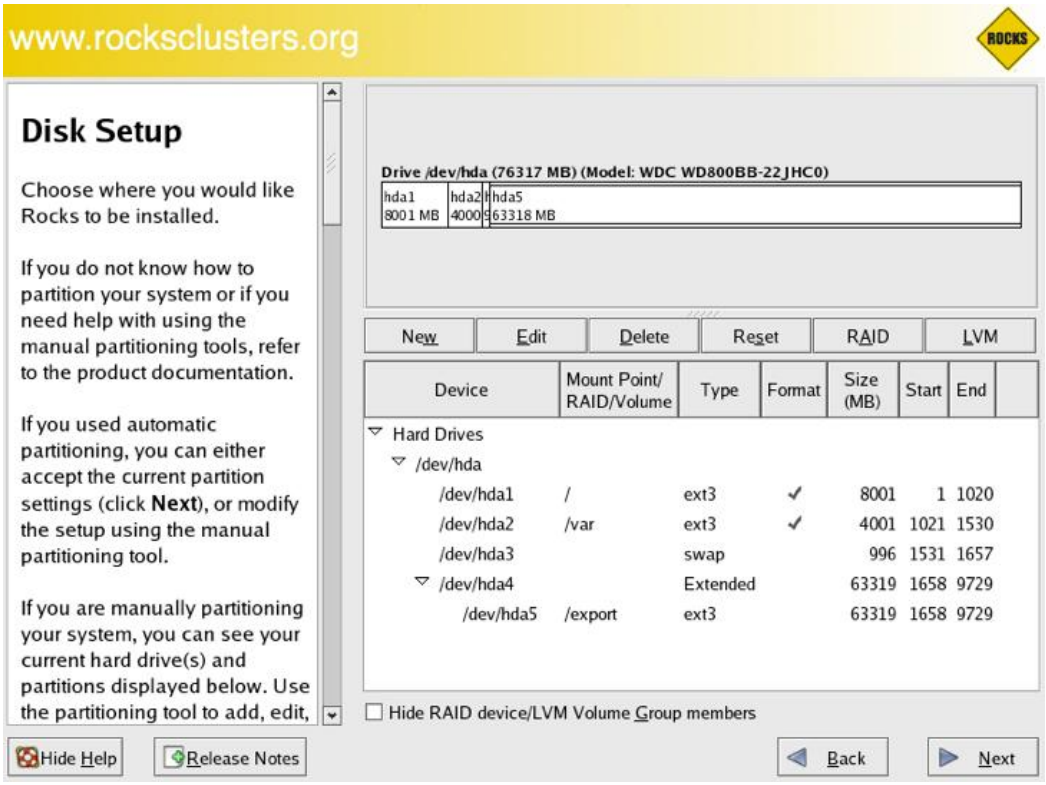


Figure 26 Manual partitioning example screen.

13. After finishing the partitioning and selecting "Next", the frontend will format the file systems. At the end of the format, it will proceed with the Roll installation (because they are on the DVD). After the last of the Rolls is installed, the frontend will reboot to complete the process.

5.1.3 Cluster Node Installation

The Rocks installation uses the Red Hat Linux Anaconda installer. This obtains all the configuration inputs it needs from the Kickstart file. At the time the installation is checking for a Kickstart file for the compute node, it will also do a check for the attached video signal. If it does not detect a monitor attached then the installation will hang at that point. To ensure the installation starts, a monitor needs to be attached to the compute node being installed. This can be bypassed on newer motherboards with Intelligent Platform Management Interface (IPMI) installed and configured. After the installation is complete, the install action can be changed to allow the nodes to function without a monitor attached. See Section 5.1.3.2 for this procedure.

1. Login to the frontend as `root` using the password that was set during the installation.
2. Enter the following to run the program that captures compute node DHCP requests and enters the information into the Rocks MySQL database.

   ```
   # insert-ethers
   ```

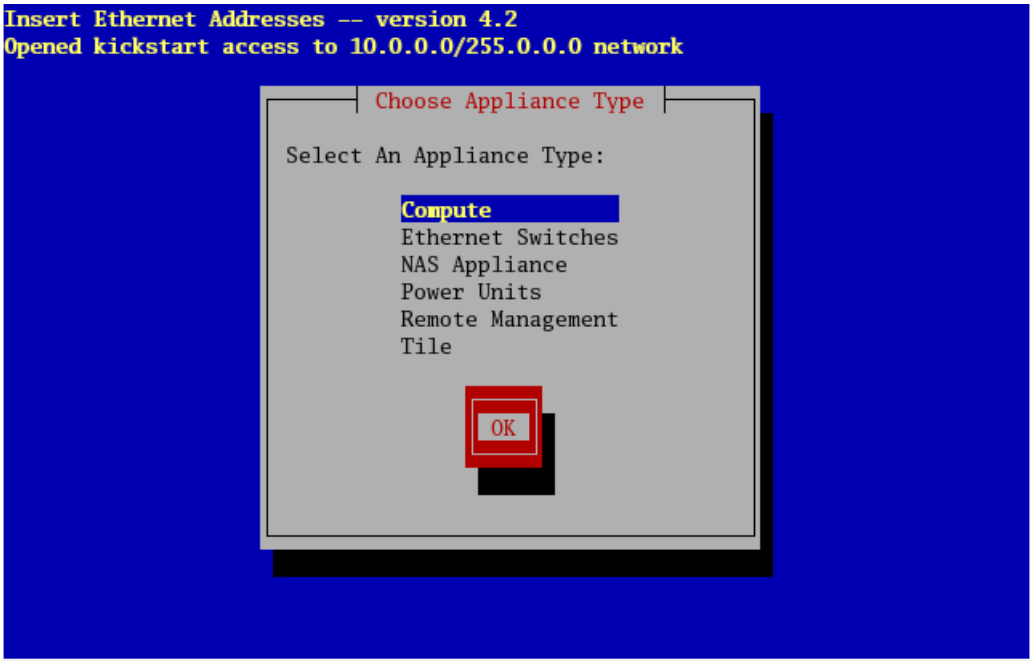


Figure 27 Appliance selection screen.

There are several Appliance types listed. Before the first node is installed, we need to select "Ethernet Switches". This is due to the fact that managed Ethernet switches issue DHCP requests in order to receive IP addresses. When `insert-ethers` captures the DHCP request from the switch it will configure it as an Ethernet switch and enter the information in the MySQL database. The `insert-ethers` may not show any indication that the switch has been configured and the blank `insert-ethers` screen shown in Figure 28 will be shown. After several minutes the F8 key should be pressed to exit. The screen shots included here are from the online manual and have not been updated to reflect the version 5.3 changes which replaced the F10/F11 keys with F8/F9. Restart `insert-ethers` and select "Compute" for the compute node.

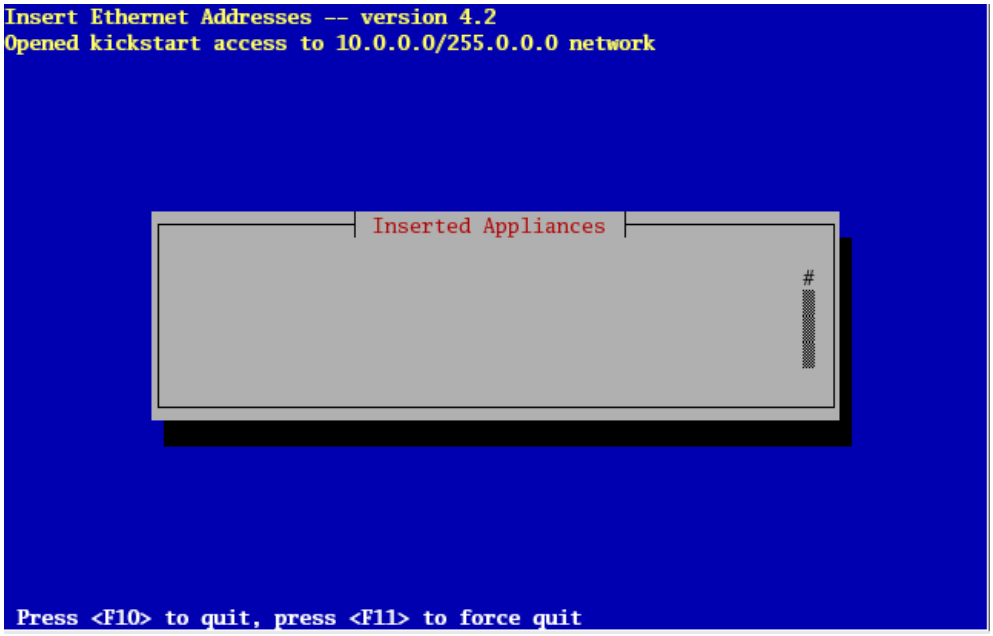


Figure 28 Insert-Ethers screen.

3. The blank `insert-ethers` screen will appear.
4. Start the first compute node. As the node starts up, we entered the BIOS to ensure that the boot sequence is set to PXE (Network Boot) prior to Hard Disk. For PCs that do not support the PXE boot, the installation would have been to be done by inserting the kernel Roll CD and booting from CD first. For the WOPPR, all PCs are capable of PXE boot. As the node progresses through the boot sequence, it will broadcast its MAC ID while it looks for a PXE connection. Insert-ethers receives the DHCP request from the node and displays it on the screen, shown in Figure 29 below.

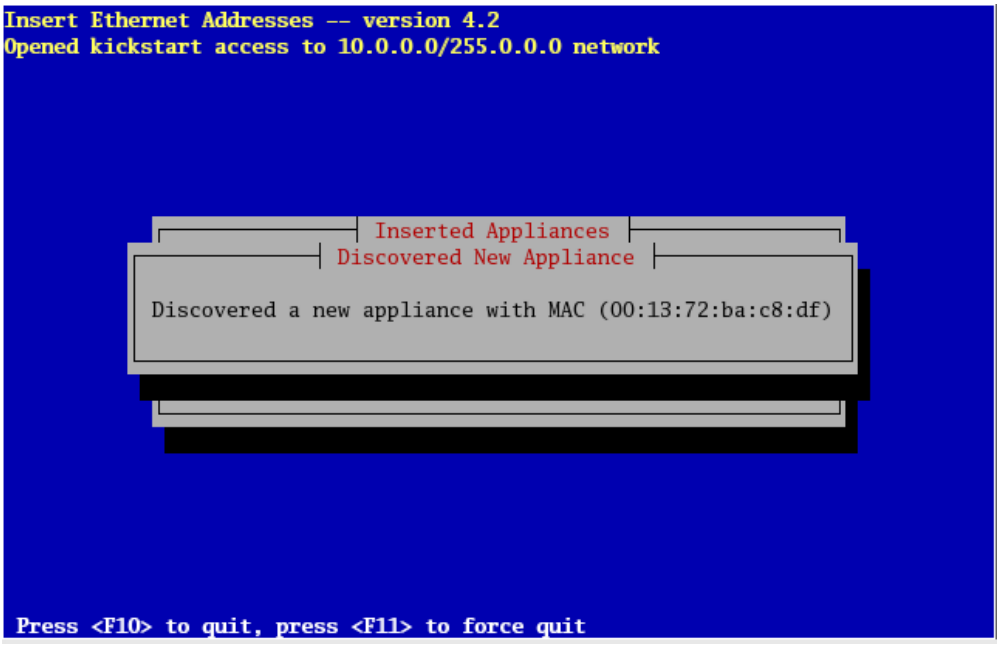


Figure 29 Insert-Ethers recognizes MAC address.

This screen indicates that the DHCP request from the node has been received, inserted into the MySQL database, and that all configuration files have been updated. As soon as that process is completed (lasting about 5 seconds) then the screen in Figure 30 will be shown.

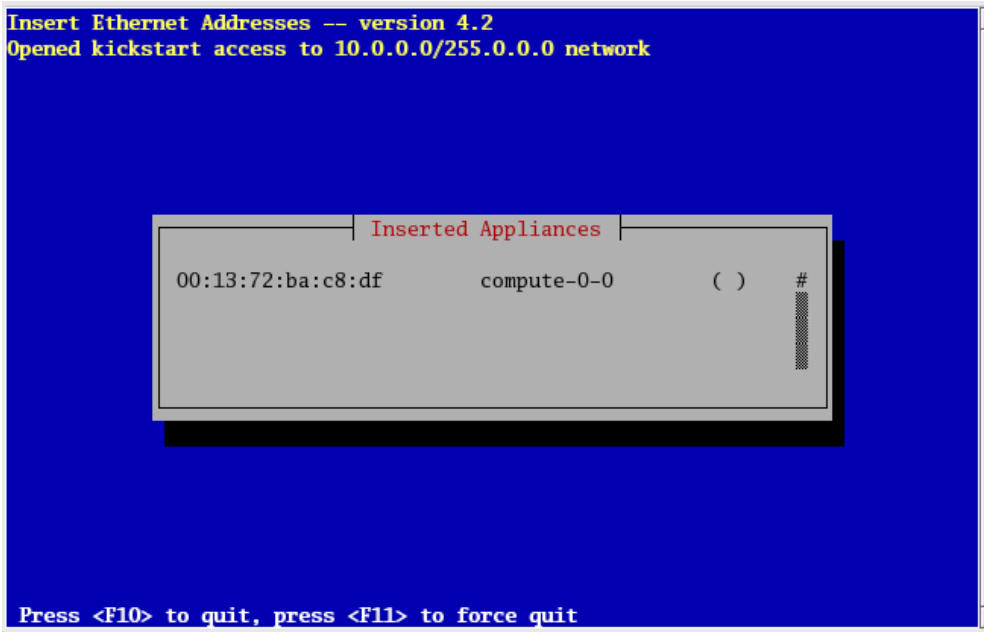


Figure 30 Insert-Ethers assigns the node name.

This indicates that the node has not requested a Kickstart file yet. In this screen the new node name is shown. The default format for Rocks is "`compute-x-x`". If we were going to install a node with a specific node name then we would follow the procedure listed below in Section 5.1.3.1. When the new node has successfully requested the Kickstart file the screen will change to show an asterisk as shown in Figure 31. If there was an error during this part, an error code would be displayed where the asterisk is.

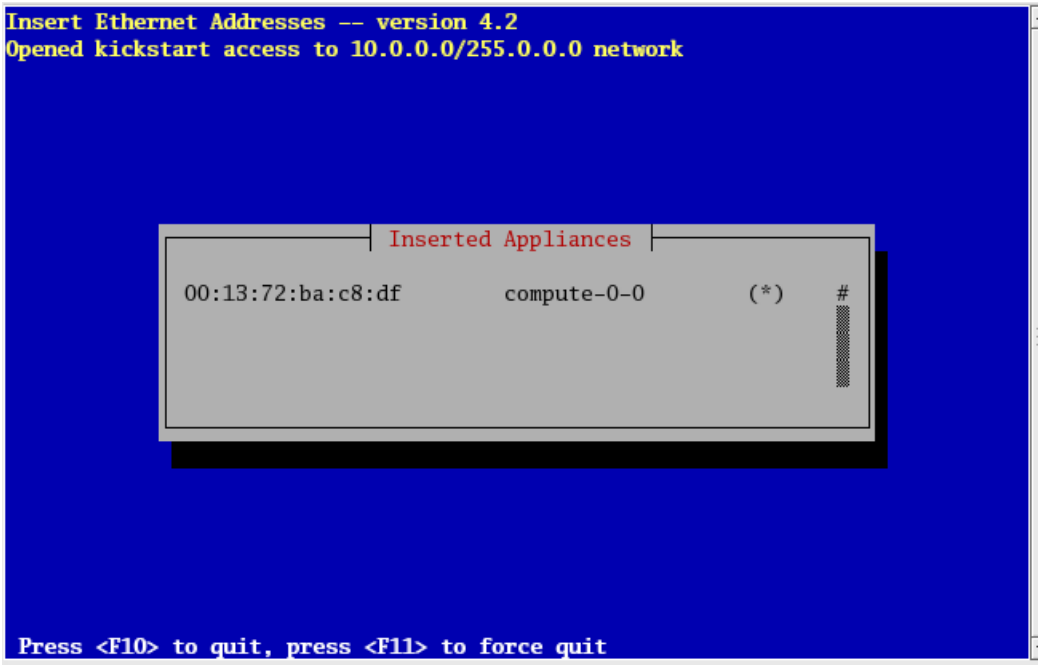


Figure 31 Insert-Ethers, node Kickstart request successful.

5. After the asterisk is shown, F8 can be pressed to exit this screen before starting the next node installation. This process was repeated to install all 10 compute nodes into the WOPPR cluster.

   Since programs and scripts should never be run from the root user if possible, the first thing that was done after the installation was to set up the user accounts. This was done using standard Linux commands as follows:

```
# useradd username
# passwd username
# rocks sync users
```

This ‘sync’ forces the 411 service to update all the configuration files and propagate the changes to the compute nodes.

5.1.3.1 Fun with Node Names

If a specific compute node name is required, then the steps below should be followed. Compute nodes are installed in Rocks in cabinets and each cabinet is filled with its nodes. So under the default nomenclature, compute-0-4 is the 5th node in the first cabinet and compute-2-3 is the 4th node in the 3rd cabinet.

- To assign specific cabinet and rank:

  ```
  # insert-ethers --cabinet=x --rank=y
  ```

  Where x and y are the numbers you want to assign to this particular node. If only the cabinet switch is used, then the rank would start at 0 and increment with nodes.

- To use a name other than the default “compute”:

  ```
  # insert-ethers --basename=”c”
  ```

  Where “c” would be the name you want to use. This would result in nodes c-<cabinet>-<rank>.

- To correct any errors or incomplete setup of nodes, the database entries have to be removed, and the configuration files need to be synchronized before continuing. The following steps should be used:

```
# insert-ethers --remove "compute-x-x"

# insert-ethers --update

# rocks sync config

# insert-ethers --cabinet=x --rank=y
```

5.1.3.2 "Headless" Installs

In order to configure the compute nodes for re-installations without having a monitor hooked up, the INSTALLACTION needs to be changed. Rocks supports several install actions, these are listed below.

```
[root@woppr var]# rocks list bootaction
ACTION    KERNEL              RAMDISK              ARGS
install:  vmlinuz-5.3-x86_64 initrd.img-5.3-x86_64 ks ramdisk_size=150000 lang= devfs=nomount pxe kssendmac
selinux=0 noipv6
install headless: vmlinuz-5.3-x86_64 initrd.img-5.3-x86_64 ks ramdisk_size=150000 lang= devfs=nomount pxe kssendmac
selinux=0 noipv6 headless vnc
memtest:  kernel memtest     ------------------ ------------------------------------------------------------------------
os:       localboot 0        ------------------ ------------------------------------------------------------------------
pxeflash: kernel memdisk bigraw pxeflash.img          keeppxe
rescue:   vmlinuz-5.3-x86_64    initrd.img-5.3-x86_64 ks ramdisk_size=150000 lang= devfs=nomount pxe kssendmac
selinux=0 noipv6 rescue
```

For each node to be configured, the following was entered:

```
# rocks set host installaction compute-0-0 action="install headless"
```

After all the applicable nodes have been set, the following command was run to check them:

```
# rocks list host
```

The output is listed below:

```
[root@woppr var]# rocks list host
HOST          MEMBERSHIP CPUS RACK RANK RUNACTION INSTALLACTION
woppr:        Frontend   4    0    0    os        install
compute-0-3: Compute    2    0    3    os        install headless
compute-0-5: Compute    2    0    5    os        install headless
compute-0-6: Compute    2    0    6    os        install headless
compute-0-7: Compute    2    0    7    os        install headless
compute-0-8: Compute    2    0    8    os        install headless
compute-0-9: Compute    2    0    9    os        install headless
```

```
compute-0-1: Compute    2    0    1    os        install headless
compute-0-2: Compute    2    0    2    os        install headless
compute-0-0: Compute    2    0    0    os        install headless
compute-0-4: Compute    2    0    4    os        install headless
```

5.1.4 Ganglia

The Rocks installation includes Ganglia which provides a web based graphical interface for monitoring all metrics recorded from the cluster. Ganglia is a scalable distributed monitoring system for high performance computing systems such as clusters and Grids[22]. Ganglia uses a multicast listen and announce type of protocol to monitor a cluster. This provides the ability to automatically detect the presence or absence of nodes. This is accomplished by maintaining a heartbeat on each node. If the heartbeat is missing for a period of time from a node, then that node is considered inactive. The heartbeat indicator is shown below in Ganglia's Node view of the cluster, Figure 32.

Figure 32 Ganglia node physical view.

Each node collects metric data on itself and distributes the information whenever an update occurs. All nodes listen for these updates; therefore all nodes maintain data on the status of the entire cluster. The implementation of ganglia is accomplished through the use of several daemons and command line tools. Each node or machine that will be

monitored runs the *gmond* daemon. This is a small service running in background that collects the metric data from the machine it is on and uses the listen/announce protocol to transmit this data over TCP. The *gmetad* daemon is a meta daemon used to collect data from all *gmetad* and *gmond* sources and store the information to disk. The *gmetric* command line application allows some customization of metrics on hosts being monitored. Another command line tool is *gstat* which can be used to query a specific *gmond* for information. A screen shot of the Ganglia frontend is shown in Figure 33 below. This is the primary means of monitoring the cluster activity and health.

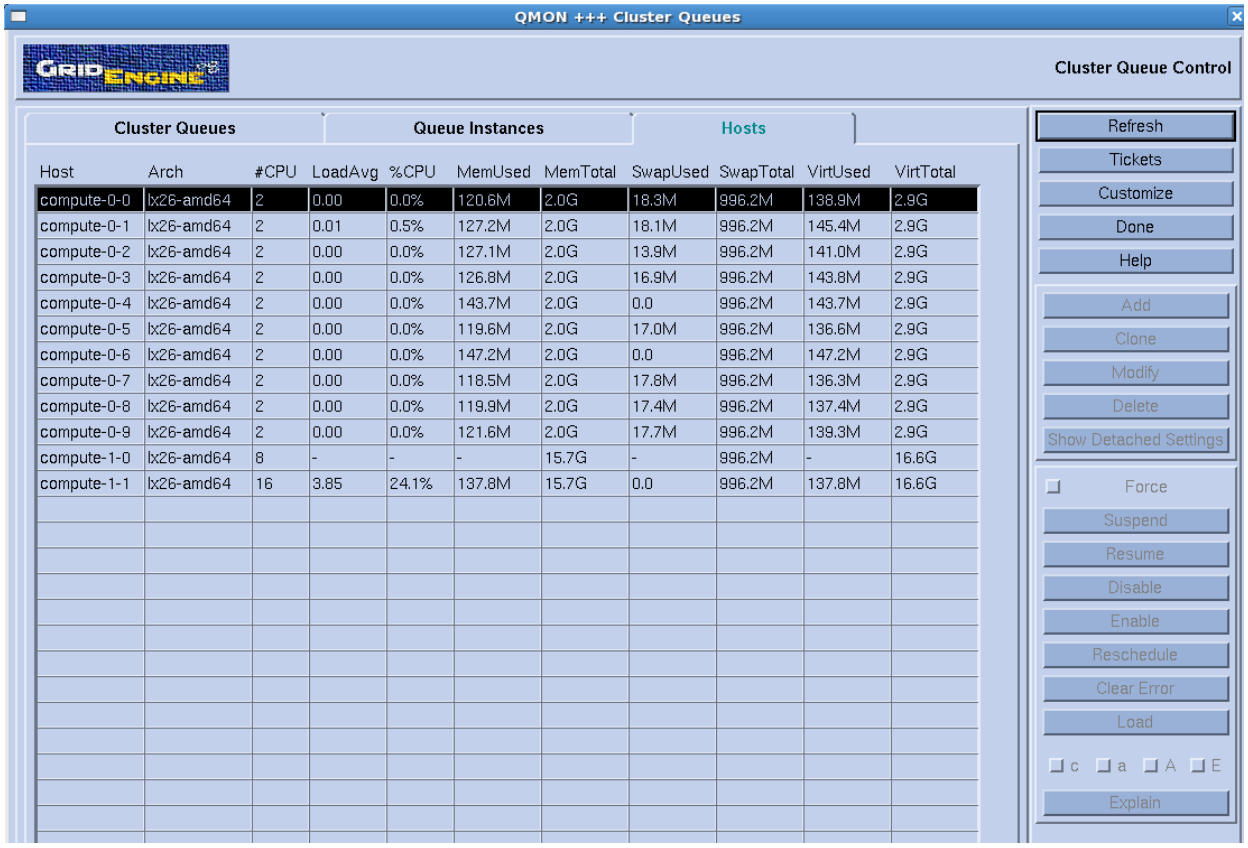
Figure 33 Ganglia frontend for cluster.

5.1.5 Sun Grid Engine

The Sun Grid Engine (SGE) is a workload manager used to manage and balance the distribution of jobs across the cluster. The SGE Roll was one of the Rolls selected and installed during the Rocks Cluster installation. The frontend pc runs the *qmaster* daemon and the nodes all run the *execution* daemon. The concept behind the SGE is to have a management tool in place that can automatically perform the job of scheduling and running submitted jobs across the cluster. When a cluster has only several nodes, this would seem to be a trivial concern. But since a cluster could reach thousands of nodes in size (10,000 is the current limit on SGE), this task would be next to impossible to manually perform.

To submit a job, a user would use one of the submission commands such as *qsub* or use the graphical user interface, QMON, to submit it. The QMON Main Control panel is shown in Figure 34. Once a job is submitted, it passes through the following three states: pending, scheduled, and running. While it is pending, the *qmaster* uses policies that may have been set up to determine its ranking. If no policies are set, then all jobs have the same importance. As soon as the number of slots required is available, the job will be scheduled to a machine. After the job is scheduled, it is sent to the execution daemon on that machine to be run. The QMON hosts screen is shown in Figure 35. This view shows what hosts are available, how many CPUs are available, and information on memory that may be needed to configure jobs. (Figure 35 also shows two nodes, compute-1-0 and compute-1-1, that were added to the cluster for some testing not directly related to this project)

Figure 34 QMON main control panel.



Host	Arch	#CPU	LoadAvg	%CPU	MemUsed	MemTotal	SwapUsed	SwapTotal	VirtUsed	VirtTotal
compute-0-0	lx26-amd64	2	0.00	0.0%	120.6M	2.0G	18.3M	996.2M	138.9M	2.9G
compute-0-1	lx26-amd64	2	0.01	0.5%	127.2M	2.0G	18.1M	996.2M	145.4M	2.9G
compute-0-2	lx26-amd64	2	0.00	0.0%	127.1M	2.0G	13.9M	996.2M	141.0M	2.9G
compute-0-3	lx26-amd64	2	0.00	0.0%	126.8M	2.0G	16.9M	996.2M	143.8M	2.9G
compute-0-4	lx26-amd64	2	0.00	0.0%	143.7M	2.0G	0.0	996.2M	143.7M	2.9G
compute-0-5	lx26-amd64	2	0.00	0.0%	119.6M	2.0G	17.0M	996.2M	136.6M	2.9G
compute-0-6	lx26-amd64	2	0.00	0.0%	147.2M	2.0G	0.0	996.2M	147.2M	2.9G
compute-0-7	lx26-amd64	2	0.00	0.0%	118.5M	2.0G	17.8M	996.2M	136.3M	2.9G
compute-0-8	lx26-amd64	2	0.00	0.0%	119.9M	2.0G	17.4M	996.2M	137.4M	2.9G
compute-0-9	lx26-amd64	2	0.00	0.0%	121.6M	2.0G	17.7M	996.2M	139.3M	2.9G
compute-1-0	lx26-amd64	8	-	-	-	15.7G	-	996.2M	-	16.6G
compute-1-1	lx26-amd64	16	3.85	24.1%	137.8M	15.7G	0.0	996.2M	137.8M	16.6G

Figure 35 QMON cluster queues page.

The job will continue to run until it either completes, fails, or is terminated or interrupted. After completing or failing, the execution daemon notifies *qmaster,* and the job is removed from the list of active jobs. Figure 36 below shows an example of the Finished Jobs screen from QMON.

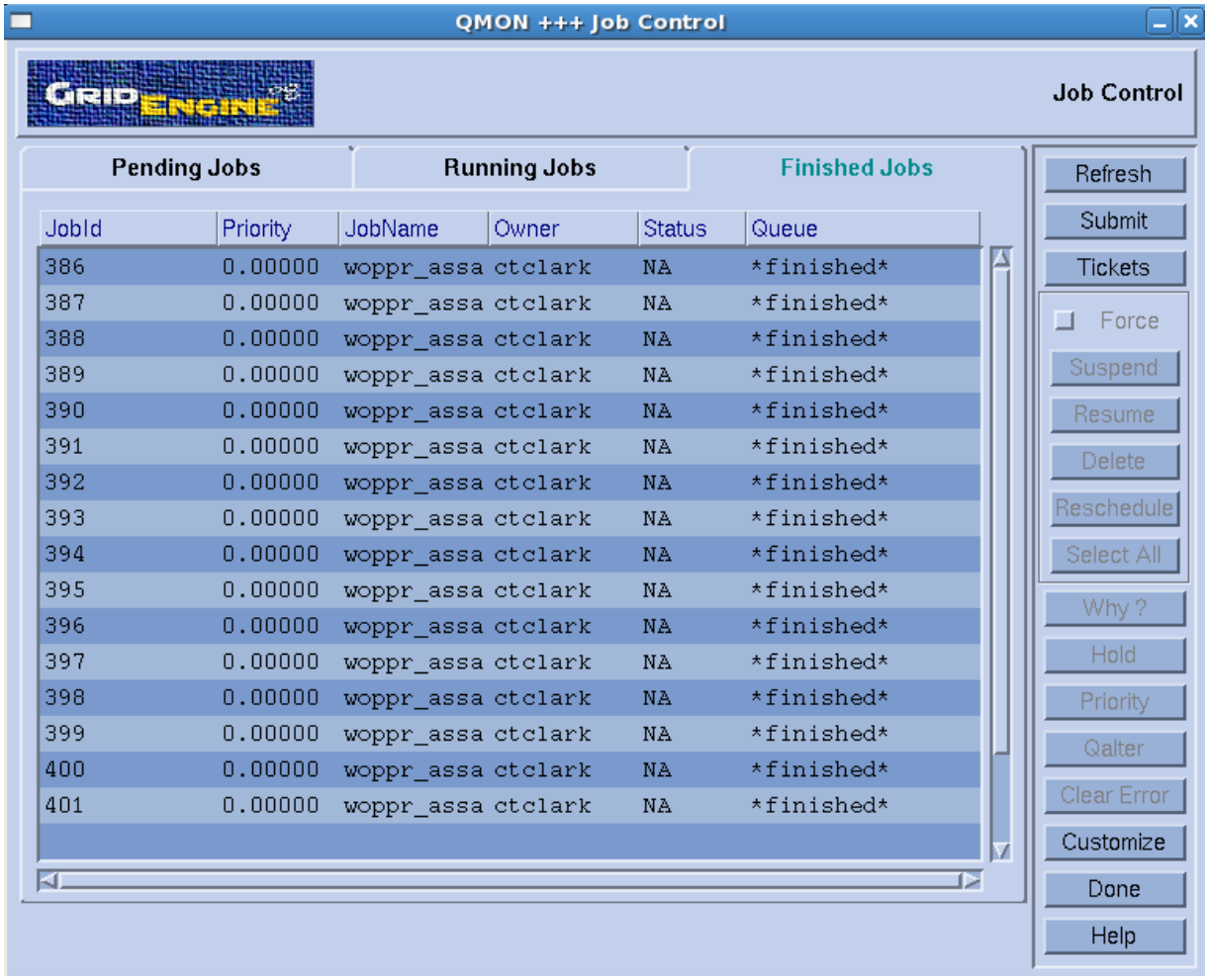


Figure 36 QMON job control screen.

SGE is capable of running interactive jobs, parallel, and what it terms 'array jobs'. Parallel jobs are handled with the MPICH2 that is included in the Rocks distribution to manage the message passing required for parallel processes. The 'array' jobs are serial jobs. The difference between the parallel jobs and the serial jobs is that SGE must run all the parallel processes simultaneously while the array jobs can run serially or in tandem.

5.1.6 Area51

The Area51 Roll was also selected and installed during the Rocks installation. This contains the following two software packages: Tripwire[23] and chkrootkit[24]. Tripwire is a free software security and data integrity tool based on code originally contributed by Tripwire, Inc in 2000. The Tripwire scans the filesystem and stores information about it in a database. Subsequent scans can compare the information to the baseline and alert the user to any changes. After Tripwire is installed on the system, it can be viewed from a tab on the main cluster web page shown in Figure 37.

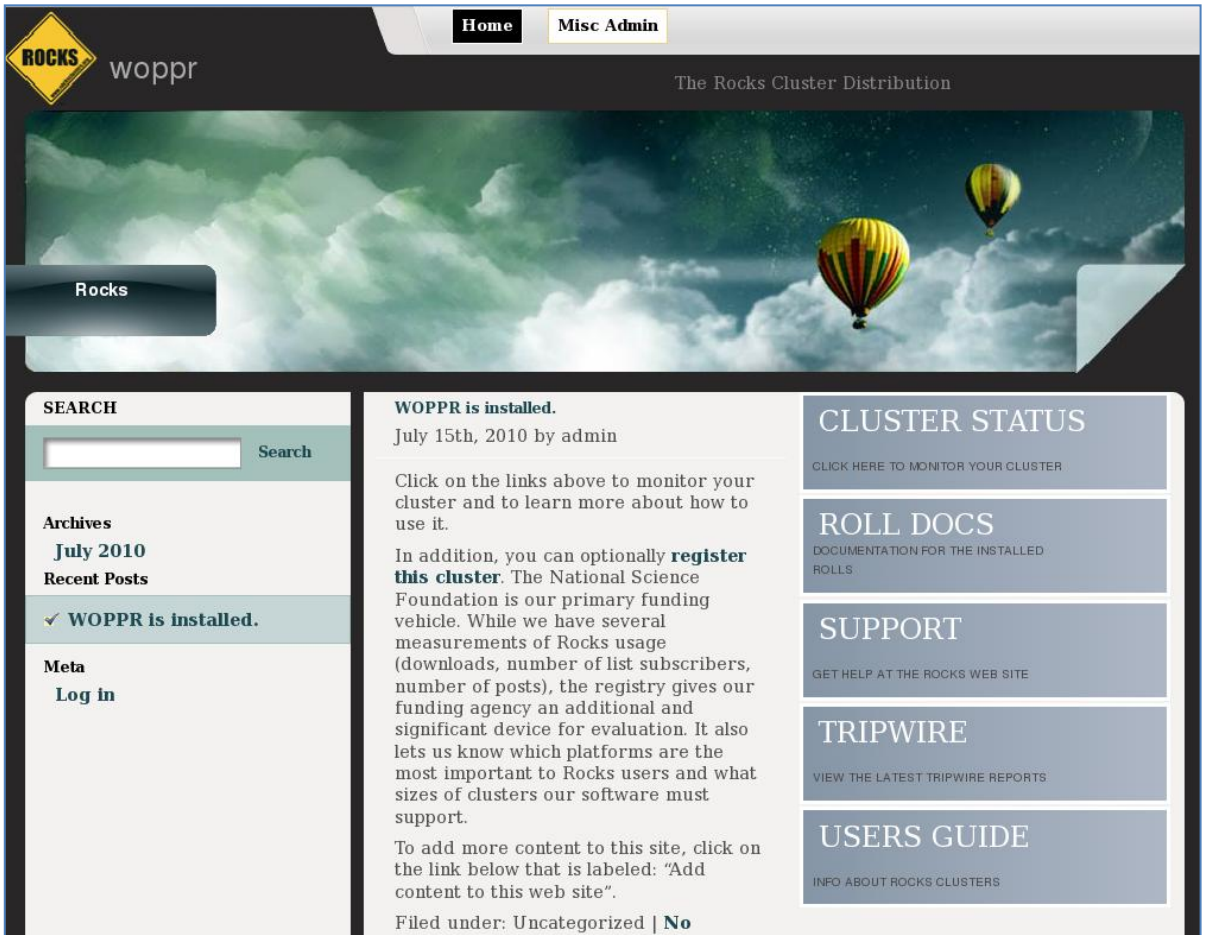
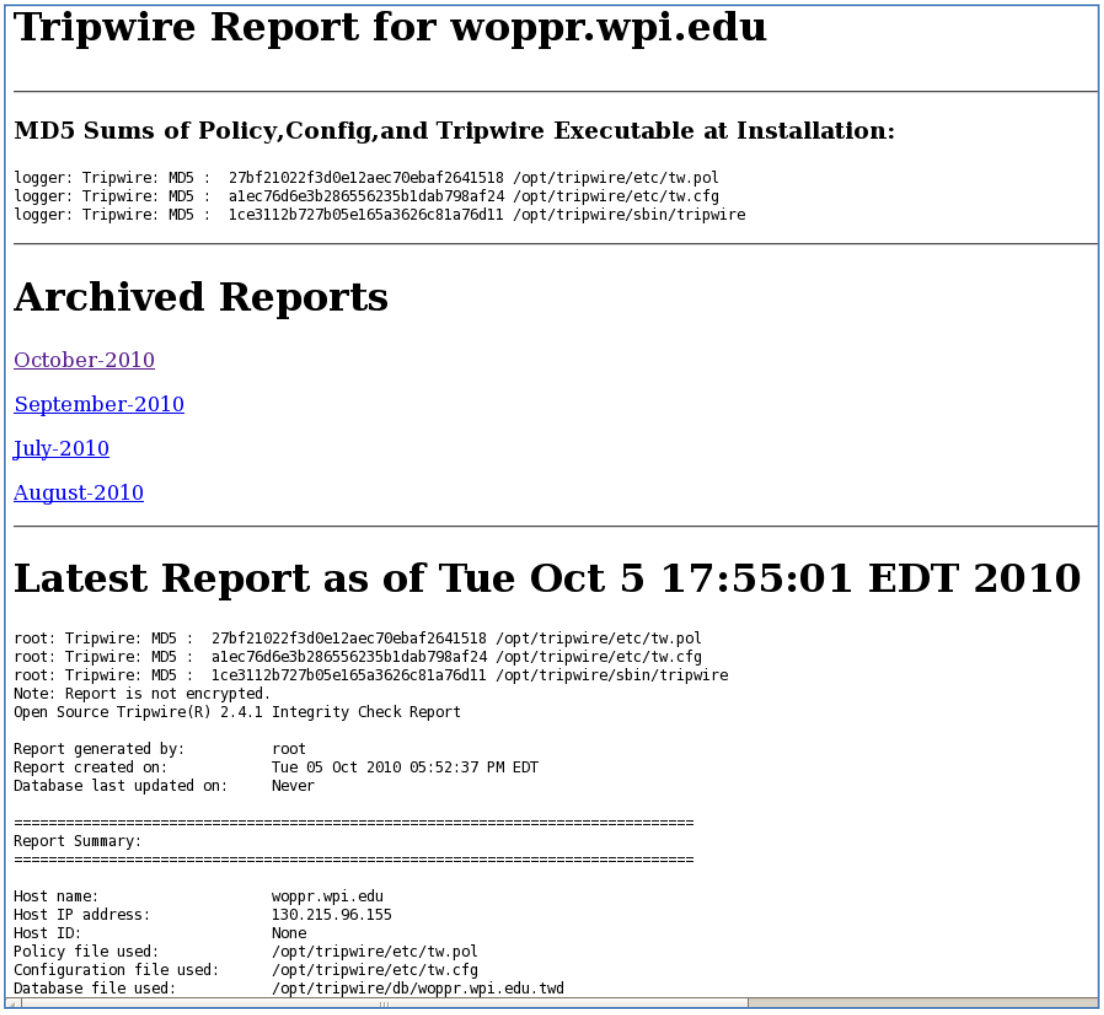


Figure 37 Main cluster webpage.

Selecting the Tripwire tab on the page above will open a Reports page with links to report archives by month as well as the most recent report. Each report is a chronological listing of any changes or policy violations and the details of each instance. An excerpt from the cluster Tripwire report page is shown below in Figure 38.



Tripwire Report for woppr.wpi.edu

MD5 Sums of Policy,Config,and Tripwire Executable at Installation:

```
logger: Tripwire: MD5 :  27bf21022f3d0e12aec70ebaf2641518 /opt/tripwire/etc/tw.pol
logger: Tripwire: MD5 :  a1ec76d6e3b286556235b1dab798af24 /opt/tripwire/etc/tw.cfg
logger: Tripwire: MD5 :  1ce3112b727b05e165a3626c81a76d11 /opt/tripwire/sbin/tripwire
```

Archived Reports

October-2010

September-2010

July-2010

August-2010

Latest Report as of Tue Oct 5 17:55:01 EDT 2010

```
root: Tripwire: MD5 :  27bf21022f3d0e12aec70ebaf2641518 /opt/tripwire/etc/tw.pol
root: Tripwire: MD5 :  a1ec76d6e3b286556235b1dab798af24 /opt/tripwire/etc/tw.cfg
root: Tripwire: MD5 :  1ce3112b727b05e165a3626c81a76d11 /opt/tripwire/sbin/tripwire
Note: Report is not encrypted.
Open Source Tripwire(R) 2.4.1 Integrity Check Report

Report generated by:          root
Report created on:            Tue 05 Oct 2010 05:52:37 PM EDT
Database last updated on:     Never

================================================================================
Report Summary:
================================================================================

Host name:                    woppr.wpi.edu
Host IP address:              130.215.96.155
Host ID:                      None
Policy file used:             /opt/tripwire/etc/tw.pol
Configuration file used:      /opt/tripwire/etc/tw.cfg
Database file used:           /opt/tripwire/db/woppr.wpi.edu.twd
```

Figure 38 Tripwire Reports page.

Chkrootkit is a program for checking systems for known rootkits. It is a command line tool that makes comparisons between the filesystem and the output of the process status command to look for directories. A check can be made by running the following command line:

```
# /opt/chkrootkit/bin/chkrootkit
```

This will result in an output similar to the following:

```
[root@woppr ~]# /opt/chkrootkit/bin/chkrootkit
ROOTDIR is `/'
Checking `amd'... not found
Checking `basename'... not infected
```

5.1.7 HPC

The HPC roll contains software that is required to run parallel applications across a cluster. It consists of the following software packages:

- OpenMPI and MPICH2
- PVM
- Benchmarks (stream, iperf, IOzone)

OpenMPI is an open source implementation of the MPI-2 Message Passing Interface standard[25]. MPICH2 is also an open source implementation of the message passing libraries and covers implementation of MPI-1 thru MPI-2.2. The Message Passing Interface (MPI) is a library of subroutines that can be called from Fortran or C programs. These subroutines are what are used to program parallel code.

PVM (Parallel Virtual Machine) is a package that permits a cluster to be used as a single large parallel computer. This is accomplished by running a daemon on all of the computers making up the cluster and starting the PVM program to create a virtual machine. The jobs that are run in this environment use the PVM library which provides the user-callable routines for message passing, coordinating tasks, etc.

The use of the PVM program was not included in the scope of this project.

5.1.8 Programming Languages

The Rocks installation includes the GNU Compiler Collection which includes front ends for C, C++, and Fortran. These libraries are free open source collections. The compilers installed on our cluster are shown in Figure 39 below.

```
[root@woppr ~]# ompi_info |grep compiler
 C compiler: gcc
 C compiler absolute: /usr/bin/gcc
 C++ compiler: g++
 C++ compiler absolute: /usr/bin/g++
Fortran77 compiler: gfortran
Fortran77 compiler abs: /usr/bin/gfortran
Fortran90 compiler: gfortran
Fortran90 compiler abs: /usr/bin/gfortran
```

Figure 39 Compilers installed on the WOPPR.

5.2 GotoBLAS2

For a Basic Linear Algebra package, we installed GotoBLAS2. This is a package of hand-coded subroutines developed and optimized by Kazushige Goto[26]. These subroutines are used to perform basic linear algebra operations such as vector and matrix multiplication.

5.2.1 GotoBLAS2 Installation

GotoBLAS2 is available from the Texas Advanced Computing Center[27]. After the untar was done placing the program in the /opt/GotoBLAS directory, the installation is started by running the script "make". This detects the Fortran compiler, the number of cores and the architecture of the processor. Since the default gcc library installed in the 64-bit Rocks is also 64-bit, the GotoBLAS installer detects it and creates a 64-bit library. After the build was complete it displayed the following system information:

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

 GotoBLAS2 build complete.

  OS              ... Linux
  Architecture    ... x86_64
  BINARY          ... 64bit
  C compiler      ... GCC  (command line : gcc)
  Fortran compiler ... GFORTRAN  (command line : gfortran)
  Library Name    ... libgoto_penrynp-r1.13.a (Multi threaded; Max
  num-threads is 4)

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

Figure 40 GotoBLAS2 build Information.

5.3 HPL

HPL is a Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. This is an industry standard benchmark for high performance computers, and is used to rank the performance of computers submitted to the Top500 list[28]. This is a list of the 500 most powerful computer systems in the world. These are general purpose computers that are in common use. HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. The HPL package requires an implementation of MPI be installed on the system as well as a Linear Algebra System.

The HPL program uses a data configuration file called HPL.dat. This file is shown in Table 15. Almost all of the variables involved in the linear calculations can be set with this data file. This was done to allow each user the opportunity to tune the process to a specific system. The details of this data file are covered in the benchmarking section.

5.3.1 HPL Installation

The HPL software is available from the netlib.org website[29]. After downloading the file has to be unzipped with "gunzip hpl.tgz" and extracted by tar "–xvf hpl.tar". This creates the hpl directory and puts the program there. This is considered the top level directory, and will be used during the configuration setup. The next step is to create a Make.<arch> file in the top directory. We checked our architecture <arch> by typing:

```
[root@bes ~]# arch
x86_64
```

This Make file contains the information on compilers, libraries, and paths. The program has many generic make files in the /setup subdirectory as shown below:

```
[ctclark@woppr hpl]$ ls setup
Make.FreeBSD_PIV_CBLAS   Make.IRIX_FBLAS            Make.Linux_PII_CBLAS
Make.Linux_PII_VSIPL     Make.PWRPC_FBLAS       Make.T3E_FBLAS
make_generic                                     Make.Linux_ATHLON_CBLAS
Make.Linux_PII_CBLAS_gm    Make.Linux_PII_VSIPL_gm    Make.SUN4SOL2_FBLAS
Make.Tru64_FBLAS
Make.HPUX_FBLAS            Make.Linux_ATHLON_FBLAS   Make.Linux_PII_FBLAS
Make.PWR2_FBLAS          Make.SUN4SOL2-g_FBLAS  Make.Tru64_FBLAS_elan
Make.I860_FBLAS                                   Make.Linux_ATHLON_VSIPL
Make.Linux_PII_FBLAS_gm  Make.PWR3_FBLAS           Make.SUN4SOL2-g_VSIPL
Make.UNKNOWN.in
```

We took the Make.Unknown.in file and edited that for our system. The changes required are highlighted below. The full configuration file is included as Appendix E.

```
# - shell --------------------------------------------------------------
#
SHELL        = /bin/sh
#
CD           = cd
CP           = cp
LN_S         = ln -s
MKDIR        = mkdir
RM           = /bin/rm -f
TOUCH        = touch
#
```

```
# ----------------------------------------------------------------------
# - Platform identifier ------------------------------------------------
# ----------------------------------------------------------------------
#
ARCH         = x86_64
#
# ----------------------------------------------------------------------
# - HPL Directory Structure / HPL library ------------------------------
# ----------------------------------------------------------------------
#
TOPdir       = /export/home/ctclark/Desktop/hpl
INCdir       = $(TOPdir)/include
BINdir       = $(TOPdir)/bin/$(ARCH)
LIBdir       = $(TOPdir)/lib/$(ARCH)
#
HPLlib       = $(LIBdir)/libhpl.a
#
# ----------------------------------------------------------------------
# - Message Passing library (MPI) --------------------------------------
# ----------------------------------------------------------------------
#
MPdir        = /opt/openmpi
MPinc        = -I$(MPdir)/include
MPlib        = $(MPdir)/lib/libmpi.so
#
# ----------------------------------------------------------------------
# - Linear Algebra library (BLAS or VSIPL) -----------------------------
# ----------------------------------------------------------------------
#
LAdir        = /opt/GotoBLAS2
LAinc        =
LAlib        = $(LAdir)/libgoto2.a
#
F2CDEFS      = -DAdd__ -DF77_INTEGER=int -DStringSunStyle
# ----------------------------------------------------------------------
# - Compilers / linkers - Optimization flags ---------------------------
# ----------------------------------------------------------------------
#
CC           = /usr/bin/gcc
CCNOOPT      = $(HPL_DEFS)
CCFLAGS      = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W -Wall
#
LINKER       = /usr/bin/gfortran
LINKFLAGS    = $(CCFLAGS)
#
ARCHIVER     = ar
ARFLAGS      = r
RANLIB       = echo
#
# ----------------------------------------------------------------------
```

Figure 41 HPL Make file configuration (affected lines).

After editing the file, it is saved as Make.x86_64. We do the build by typing "make arch=x86_64". The next step is to edit the HPL.dat file in the top directory. The HPL.dat file is shown in Table 15. The details of the HPL.dat configuration file are discussed in Section 6.1 on HPL Configuration.

5.4 Real World Application – The Gliding Assay Code

The basis for assembling this computer cluster is to provide a tool for computationally intensive research. For the purposes of testing this, we are using the gliding microtubule assay simulations of Professor Tüzel.

The cytoskeleton is a filamentous network found in cells which maintains cell shape, aids in cell motion, and plays a key role in intracellular transport and cell division. It is made up of microtubules, actin, and intermediate filaments, which together provide shape and mechanical integrity for the cell. Recent experiments in LLC-PK1 epithelial cells suggest that in addition to their role as cargo carriers, microtubules are also deformed and transported by molecular motors. Motivated by these experiments, and the supporting in-vitro gliding assay data, we model the collective behavior of microtubules and molecular motors using coarse-grained simulations.

In the simulations, microtubules are modeled as semi-flexible polymers with rigid bond constraints embedded in a solvent. Molecular motors exert forces on the microtubules, and walk along microtubule tracks according to their known force-velocity relations, binding and unbinding stochastically. The simulations typically utilize a time step which is about 100 ns, and in order to reach realistic time scales that are of the order of tens of seconds, billions of iterations are necessary. In addition, due to the

stochastic nature of the simulations, ensemble averages over many different runs are required. It is therefore necessary to have computational architectures that enable the execution of multiple serial jobs, with little overhead. We hope to learn more about the fundamental interactions between these cytoskeletal structures and proteins, and gain insight into the intracellular mechanical stresses, and the factors determining cell shape.

A simulation snapshot showing a microtubule gliding over randomly distributed molecular motors is shown in Figure 42. The actual image shown in Figure 43 is a fluorescence image from a living epithelial cell. The bright tubular structures are the microtubules, which are labeled with a fluorescent protein (called GFP, Green Fluorescent Protein).

The code written for the purpose of modeling the motors described above is called the Assay code. The installation and results for this code will be detailed in the following sections.

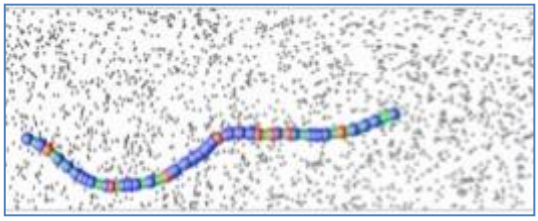


Figure 42 Microtubule simulation snapshot.

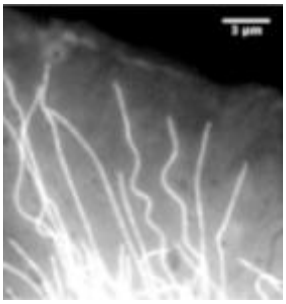


Figure 43 Microtubule image.

5.5 Intel Math Kernel Library Installation

The Intel Math Kernel Library (MKL) is required in order to run the assay code. MKL is a library of math routines for science, engineering, and financial applications. It includes BLAS, LAPACK, ScaLAPACK, and others. This is written and optimized for Intel processors[30]. After downloading and untar the software package, a simple install script performs the installation.

In order to be able to use the installed libraries across the compute nodes, the nodes need to have the libraries installed locally. This requires customization of the Rocks distribution to ensure that each time a node is rebuilt it will have the needed components included.

The 64-bit libraries that were needed from Intel were installed as part of the Math Kernel Library (MKL) installation. To be able to use them, we had to create RPMs from the library subdirectories. This was done with the following commands:

```
# cd /export/rocks/install/contrib/5.3/x86_64/RPMS
# rocks create package /opt/intel/lib intel-node-libs
```

Check the contents of the rpm

```
#rpm -qlp intel-node-libs*rpm
```

These are intel-node-libs-1.0-1.x86_64.rpm and intel-node-mkl-libs-1.0-1.x86_64.rpm. These were placed into the directory:

```
/export/rocks/install/contrib/5.3/x86_64/RPMS
```

The Rocks configuration is customized through the use of an XML file. Since this is the first customization, we had to create the file to be used.

```
# cd /export/rocks/install/site-profiles/5.3/nodes
```

```
# cp skeleton.xml extend-compute.xml
```

Inside the extend-compute.xml file, there are several lines commented for use in adding packages.

```
<!-- <package>insert 3rd package name here and uncomment the line</package> -->
```

We changed two of these lines to add the packages listed above. Only the base name of the package is added to the `extend-comute.xml` file.

```
<package> intel-node-libs </package>
<package> intel-node-mkl-libs </package>
```

The complete `extend-compute.xml` file is included in Appendix B.

After the XML file has been edited, a new Rocks distribution needs to be built. This binds the new package into a Red Hat compatible distribution that will be used on all subsequent installations.

```
# cd /export/rocks/install
# rocks create distro
```

After the distribution is built, all of the nodes need to be re-installed.

```
# rocks set host installaction compute-0-0 action=”install headless”
# shoot-node compute-0-0
```

Now that all of the installations have been accomplished and the nodes have been re-installed, the testing and benchmarking can be done.

Chapter 6: Testing and Benchmark Results

After all of the software had been installed it was time to test the system. The test plan included the HPL testing for an industry standard test between the cluster we had built, and an Apple Xserve node we had on loan for testing purposes. For specific real-world testing concerning research, we used varying runs of the Assay code mentioned earlier. Detailed explanations of both HPL and Assay are included in this section.

6.1 HPL Configuration

The HPL benchmark uses a file HPL.dat to allow user configuration of the test parameters. The file is show below in Table 15.

Table 15 HPL.dat Configuration File

Line #	
1	HPLinpack benchmark input file
2	WOPPR Testing, WPI
3	HPLaa.out output file name (if any)
4	8 device out (6=stdout,7=stderr,file)
5	2 # of problems sizes (N)
6	30000 41344 Ns
7	2 # of NBs
8	104 192 NBs
9	0 PMAP process mapping (0=Row-,1=Column-major)
10	4 # of process grids (P x Q)
11	2 1 8 4 Ps
12	4 8 1 2 Qs
13	16.0 threshold
14	1 # of panel fact
15	1 2 PFACTs (0=left, 1=Crout, 2=Right)
16	1 # of recursive stopping criterium
17	2 4 NBMINs (>= 1)
18	1 # of panels in recursion
19	2 NDIVs
20	1 # of recursive panel fact.
21	2 RFACTs (0=left, 1=Crout, 2=Right)
22	1 # of broadcast
23	0 BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
24	1 # of lookahead depth
25	0 DEPTHs (>=0)
26	2 SWAP (0=bin-exch,1=long,2=mix)
27	64 swapping threshold
28	0 L1 in (0=transposed,1=no-transposed) form
29	0 U in (0=transposed,1=no-transposed) form
30	1 Equilibration (0=no,1=yes)
31	8 memory alignment in double (> 0)

As mentioned earlier, all the parameters of the configuration file can be adjusted by the user. There has been some previous work concerning the behavior of HPL that suggests that not all parameters have an effect on performance[31]. However, there are enough variations in computing equipment to suggest that parameters that do not affect performance in one system may have an impact on a different system. This proved to be the case in our testing.

HPL measures the floating point execution rate as it solves an order N dense system of linear equations of the form A*x*=b using LU factorization. The matrix is divided into *NB x NB* blocks which are then dealt onto a *P x Q* processor grid using block-cyclic data distribution. The matrix size *N,* blocking factor *NB,* and the process grid ratio (*P x Q*) are the most important parameters with the other 15 input parameters used to fine tune the particular platform. Each of the line-items of HPL.dat is explained below.

- **Line 1 & 2**: These are ignored and can be changed to whatever the user wants.

```
HPLinpack benchmark input file
WOPPR Testing, WPI
```

- **Line 3**: This is used to specify the output file. If *Line 4* is configured to send the output to a file then the file name needs to be listed here at the beginning of *Line 3*. The remainder of the line can be used for whatever notes the user wants to include.

```
HPLtest9.out       output file name (if any)
```

- **Line 4**: This line specifies where the output will go. The line must begin with a positive integer and everything after that integer is ignored. There are 3 choices for the integer, a 6 means that the output will go to the standard output, a 7 means

that the output goes to the standard error, and any other positive integer means that the output is written to the file specified in *Line 3*.

```
8                 device out (6=stdout,7=stderr,file)
```

- **Line 5**: This line specifies the *number* of problem sizes (*N*) to be executed. This needs to be a positive integer ≤ 20. The rest of the line is ignored.

```
2                 # of problems sizes (N)
```

- **Line 6**: This line specifies the problem sizes to be run (*N*). Since *line 5* specifies *2*, the first two positive integers on *line 6* will be used and anything after that is ignored.

```
30000  41344      Ns
```

This value is the matrix size to be used (*N*). To get the best performance this should be the largest problem size that can fit in the memory available. The amount of memory that HPL uses is basically the size of the coefficient matrix. Since there is also overhead to be considered such as the operating system and other services, the general rule of thumb for HPL is to use about 80% of the maximum available memory. For an *N x N* matrix of double precision (8 byte) elements you consume *N*N*8bytes*. As an example, we have 8 nodes we will be using in the test, each with 2 GB of RAM for a total of 16GB of RAM.

$$N = \sqrt{\frac{(GB\ of\ RAM) * 1024^3 bytes/GB * (.8)}{8bytes}} \quad (1)$$

Equation 1 Matrix Size (N)

If the problem size (N) is too big, it will be swapped out and performance will degrade.

- **Line 7**: This line specifies the number of block sizes (NBs) to be run. This must be a positive integer ≤ 20 and the rest of the line following that is ignored.

```
2                # of NBs
```

- **Line 8**: This line contains the block sizes (*NB*) that the user wants to run. If Line 7 started with 2, as it did in this example, then only the first two positive integers of line 8 are used and anything following that is ignored.

```
104 192          NBs
```

As pointed out earlier, the matrix is divided into *NB x NB* blocks, which are dealt onto the processor grid. So the NB value affects the data distribution as well as the computational granularity. An NB value that is too small will increase the overhead caused by excess message passing and it will decrease data reuse thereby limiting the computational performance. On the other hand, a smaller NB will allow for a better load balance. The normal range of NB values is given as 32-256. The optimized value of NB is system specific and depends on the computation to communication performance ratio of the system. The NB is also supposed to scale. If an NB of 32 is found to work for smaller matrix sizes, then a multiple of 64 or 128 may work better for large problem sizes.

- **Line 9**: This setting specifies how MPI will map processes onto the nodes. If all of the nodes have single core processors, then this setting doesn't matter. For multi-core processor nodes, row-major mapping is recommended.

```
0          PMAP process mapping (0=Row-,1=Column-major)
```

- **Line 10**: This line specifies the number of process grids (*P x Q*) to be run. This must be a positive integer ≤ 20 and anything following this integer is ignored.

```
4                # of process grids (P x Q)
```

- **Line 11 & 12**: These two lines specify the process grids. The first integer on each line specifies the first grid (*P x Q*). The second number on each line specifies the second grid, and so on. Since Line 10 specified 4 process grids, there must be at least 4 sets of integers on lines 11 & 12. Anything after the 4th set will be ignored. The grid size must be a multiple of the number of processors being tested. HPL recommends a ratio of 1:k with k between 1 and 3 inclusive. *P* and *Q* should be approximately equal with Q being slightly larger. *But*, the grid ratio also depends on the physical interconnection network of the cluster. For a cluster connected by Ethernet, it is recommended to use as flat of a grid as possible, for example, 1 x 4, 1 x 8, 2 x 4, etc..

```
2 1 8 4          Ps
4 8 1 2          Qs
```

- **Line 13**: This line contains a real number used as a threshold for checking the residuals. HPL recommends a value of 16.0. If this was set to 0.0 then all tests would flag as failures. If it is set to a negative number, then this comparison is bypassed. This is useful during the tuning phase to save time. Even if a test result flags as failed it may still be a pass. An actual failure would be of the order of 10^6 or more.

```
16.0             threshold
```

- **Line 14 – 21:** These lines allow specific adjustments to the algorithm. Each of these adjustments will be processed in all possible combinations by HPL. These

values are adjusted and retested as needed to tweak the optimization of the cluster. One aspect of the tuning process that cannot be over emphasized is that these tuning adjustments are not independent. As one change to the algorithm is made, it could have an effect on other aspects of the algorithm. Also, as the size of the process grid and block sizes are changed, other factors such as the panel factorization or the way the recursion stop is handled may change. Therefore, each change must be tested and reevaluated each time another change is made.

```
1                   # of panel fact
1 2                 PFACTs (0=left, 1=Crout, 2=Right)
1                   # of recursive stopping criterium
2 4                 NBMINs (>= 1)
1                   # of panels in recursion
2                   NDIVs
1                   # of recursive panel fact.
2                   RFACTs (0=left, 1=Crout, 2=Right)
```

- **Lines 22 & 23:** These settings adjust how the algorithm broadcasts the current panel of columns in process rows using a ring topology adjusted here. The settings of 1, 3, and 4 are recommended.

```
1               # of broadcast
0               BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
```

- **Line 24 & 25:** These two lines control the look-ahead depth. With a setting of 0, HPL will factorize the following panel after the current panel is completely finished and the update by the current panel is finished. A setting of 1 or more means, that number specified of next panels will be factorized immediately and then the current panel will be finished. It is recommended that a value of either 1 or 0 be used unless you are more experienced.

```
1                   # of lookahead depth
0                   DEPTHs (>=0)
```

- **Line 26 & 27:** Line 26 specifies the swapping algorithm used by HPL. There are 3 choices; a binary-exchange, a spread-roll(long), and a mixture where the binary-exchange is used up to the threshold specified in Line 27 and then the spread-roll is used.

```
2                 SWAP (0=bin-exch,1=long,2=mix)
64                swapping threshold
```

- **Line 28:** Specifies whether the upper triangle of the panel of columns should be stored in no-transposed or transposed form.

```
0                 L1 in (0=transposed,1=no-transposed) form
```

- **Line 29:** Specifies whether the panel of rows U should be stored in so-transposed or transposed form.

```
0                 U  in (0=transposed,1=no-transposed) form
```

- **Line 30:** This enables/disables the equilibration phase. This option is not used unless either 1 or 2 is selected in Line 26.
- **Line 31:** The memory alignment for memory space allocated by HPL. HPL recommends either 4, 8, or 16 on modern machines.

```
8               memory alignment in double (> 0)
```

6.1.1 HPL Machinefile

When HPL is run with MPI, the executable (xhpl) uses a machinefile which names the nodes of the cluster. This is used to provide the node identification to the MPI processes. The machinefile used for the original cluster is shown in Table 16.

Table 16 Cluster Machinefile for HPL

compute-0-0
compute-0-1
compute-0-2
compute-0-3
compute-0-4
compute-0-5
compute-0-6
compute-0-7
compute-0-8
compute-0-9

6.2 Gliding Assay Code

As described in Section 5.4, the Assay code models the microtubule gliding over molecular motors. An example of the configuration file for the Assay Code is listed below in Table 17. The parameters changed in the configuration file for the purposes of this testing are the step length in line 7, and the Write motor configuration in line 22. The step time controls how many steps the time period is broken into. A smaller time means a shorter time period between data points and therefore *more* data points. The write motor configuration setting is a switch to turn on the writing of the motor configuration data to data files in a directory specified in line 2. The quantity of data generated is very large which means that with this setting turned on a very real data transfer consideration is introduced to the test time.

Table 17 Assay Code Configuration File

```
1  # Directory for data storage
2    data
   # Random number seed:  0=random seed, any other number serves
3  as seed
4  0
5  # Run length, averaging parameters
6  # tmax transient step  trelease
7    1.25E8  0E0       1.25E4    2.5E6
8  #
9  # Thermostat parameters
10 # temperature tau     viscosity (eta)
11   4.27D-21    2.0D-8  0.005
12 #
13 # Microtubule parameters
14 # Nm   Lpolymer  LoverLp     scale
15   64   8.0D-6    1.7D-2      1.0D8
16 # Motor parameters
17 # celldim  Motor density (#/micron^2)   Ratio of dead motors
18   128      10.0                        0.0
19 # Rcap    v_unload  f_stall  Kmotor  Lmotor  f_cut
20   1.0D-8  5.0D-7    5.0D-12  2.0D-4  0.0D0  2.5D-12
21 # Write motor conf. (1=yes, 0=no)
22 0
```

Table 18 contains a sample of the parameter data file generated by an Assay code run. This file only contains parameter output, the actual data files are written to the data directory and do not have any significant meaning to discuss here.

Table 18 Assay Code Output Data File

```
   Parameters:
   ==========
1  Linear system size (micron) = 16.25
2  # of cells per dimension =128
3  L/L_p =  0.0170          L (micron) =  8.00E+00
4  # of beads =  64 # of motors=  634
5  Capture radius (nm) =  10.00 Eq. stalk length (nm) =   0.00
6  Stall force (pN) =   5.00    Unloaded velocity (micron/sec)=   0.50
7  Spring constant = 0.0002   F_cut (pN)=   2.50
8  Dead motor %= 0.000
9  Motor density (#/micron^2)= 10.0000

10 Equilibration completed
11 End: total=   4840.000     user=   4687.000     system=   153.0000
```

- **Line 1**: This is the side length of the simulation box (shown in Figure 42).
- **Line 2**: The nearest distance to check for motor binding.
- **Line 3**: Length divided by persistence length of a microtubule. Persistence length L_p is a measure of how much a given filament, whether microtubule or not, persists in a given direction. L(micron) is the length of a microtubule.
- **Line 4**: Discretization of the microtubule. There are 64 nodes on the tube itself.
- **Line 5**: The capture radius is the distance in which a motor will bind. The stalk length is the rest length of the spring.
- **Line 6**: Stall force is the force that will stop a motor. The unloaded velocity is the fastest speed a motor can go.
- **Line 7**: This is the compliance of the motor linkage modeled by Hooke's law. F_{cut} is the force that goes into the detachment rate of a motor.
- **Line 8**: The percentage of dead motors on a surface.
- **Line 9**: The motor density.
- **Line 11**: The user value is the time in seconds that the CPU took to process the job; the system value is the time in seconds that the job devoted to data transfer; the total is the total time in seconds for the job to run.

6.3 WOPPR Cluster Testing and Results (full 10 nodes)

The original cluster we built consisted of 10 nodes and the frontend. As shown in Appendix A, all of the nodes except two had 3.4 GHz processors and 2 Mb of L2 cache. Before starting the testing, the theoretical peak performance (R_{peak}) for the system had to be determined. R_{peak} is calculated by multiplying the total number of processor cores, the processor clock frequency, and the theoretical number of double precision floating point (FP) results that the processor can process per clock tick.

$$R_{peak} = \#_{cores} * CPU_{freq} * FP \quad (2)$$

Equation 2 Theoretical Peak Performance

The Pentium 4 has 128-bit FP MUL and FP ADD units, both of which can accept either a packed or scalar operation every other cycle. Both the ADD and MUL execution units (EUs) are located on the same port which can dispatch just one of either the ADD or MUL packed or scalar operations per cycle. Therefore, peak FP operations throughput is one 64-bit FP MUL + one 64-bit FP ADD per cycle (4 SP or 2 DP FLOPS). But, to achieve it, packed 128-bit instructions must be used. If the code is not vectorized, then just one scalar (either ADD or MUL) FP operation can be dispatched per clock tick on the P4.

$$\boldsymbol{R_{peak}} = 1_{core} * 3.4GHz * 2_{FPops} = \mathbf{6.8}\ \boldsymbol{Gflops/core}\ \mathbf{DP} \quad (3)$$

Equation 3 WOPPR Theoretical Peak for Single Node

$$\textbf{Cluster } \mathbf{R_{peak}} = 6.8\ Gflops/core * 8\ cores$$

$$= 54.4\ Gflops + 2_{core} * 3.2GHz * 2_{FP} = \mathbf{67.2}\ \boldsymbol{Gflops}\ \mathbf{DP} \quad (4)$$

Equation 4 WOPPR Theoretical Peak for 10 Nodes

The first round of testing of the original cluster used the settings listed in table 19 below and was performed on all 10 nodes. The settings below were fixed for the initial testing while the matrix size, block size, and process-grid size and configurations were changed to obtain initial empirical data. Figure 44 below provides a key to simplify viewing the data result codes.

Table 19 Phase 1, Test 1 Parameters

16.0	threshold
1	# of panel fact
2	PFACTs (0=left, 1=Crout, 2=Right)
1	# of recursive stopping criterium
4	NBMINs (>= 1)
1	# of panels in recursion
2	NDIVs
1	# of recursive panel fact.
1	RFACTs (0=left, 1=Crout, 2=Right)
1	# of broadcast
1	BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1	# of lookahead depth
1	DEPTHs (>=0)
2	SWAP (0=bin-exch,1=long,2=mix)
64	swapping threshold
0	L1 in (0=transposed,1=no-transposed) form
0	U in (0=transposed,1=no-transposed) form
1	Equilibration (0=no,1=yes)
8	memory alignment in double (> 0)

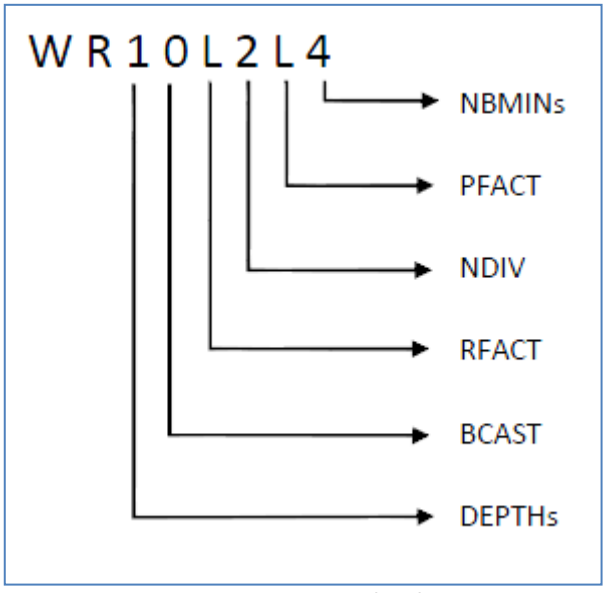


Figure 44 HPL Output data key.

6.3.1 WOPPR Cluster, First Tests

Parameters: The process-grid was a 2x5 matrix, of matrix size N = 46336, and varying NB from 32 – 128.

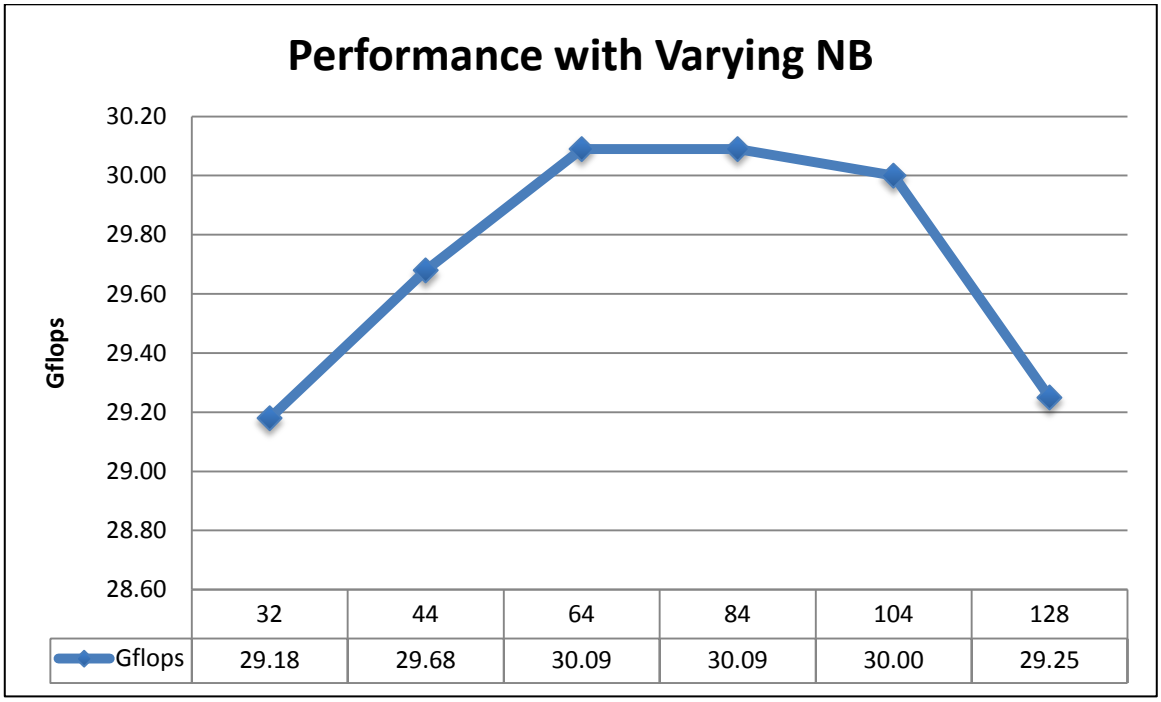


Figure 45 Original cluster, phase 1, test 1.

Another test was run at NB=64 to verify the high R_{max} value. R_{max} = 30.15 Gflops

6.3.1.1 WOPPR Cluster, Single Core Test

A single core test was run on the cluster to validate the calculated R_{peak} of Equation 3. The test was run with a process grid of *1x1*, *NB* = 64, and *N* = 1640. The resulting node performance (R_{max}) was 4.84 Gflops.

Computer efficiency (*compEff*) is the ratio of cluster performance to the theoretical peak performance.

$$compEff = R_{max}/R_{peak} * 100 \quad (5)$$

Equation 5 Computer Efficiency

Single node computer efficiency was:

$$compEff = {}^{4.84}/_{6.8} * 100 = 71.2\% \text{ Single Node} \quad (6)$$

Equation 6 WOPPR Single Node Computer Efficiency

6.3.3 Original Cluster, Parameter Tuning

After obtaining some initial results, we gathered some data in order to determine the impact of the major contributing parameters, process grid size, process grid geometry, and blocking size (NB).

First we ran a test with multiple N values on a fixed matrix of 2 x 5 and NB = 128. This verified that the largest obtainable R_{max} was found at the maximum N value. These results are shown in Figure 46.

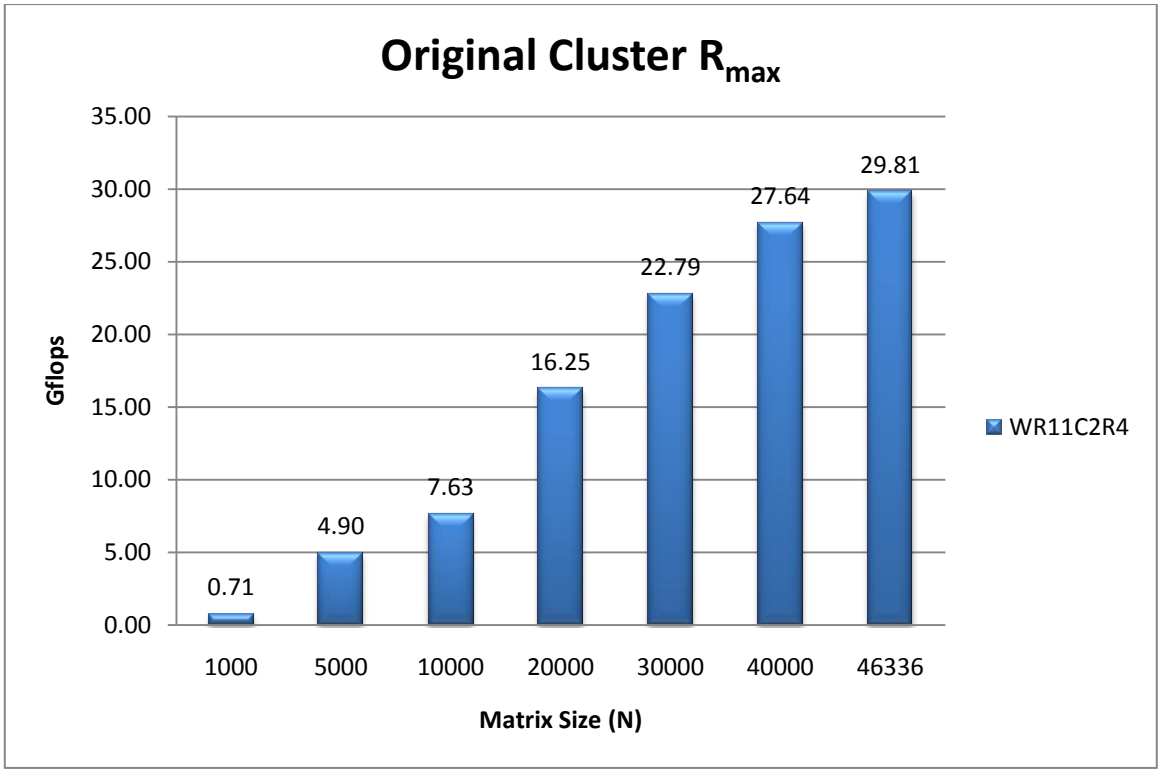


Figure 46 Original cluster, multiple N, NB=128, 2x5 matrix.

We then ran two sets of tests to vary the NB values with N fixed at 46336 and matrices of 2 x 5 and 3 x 3. Results are shown in Figures 47 and 48.

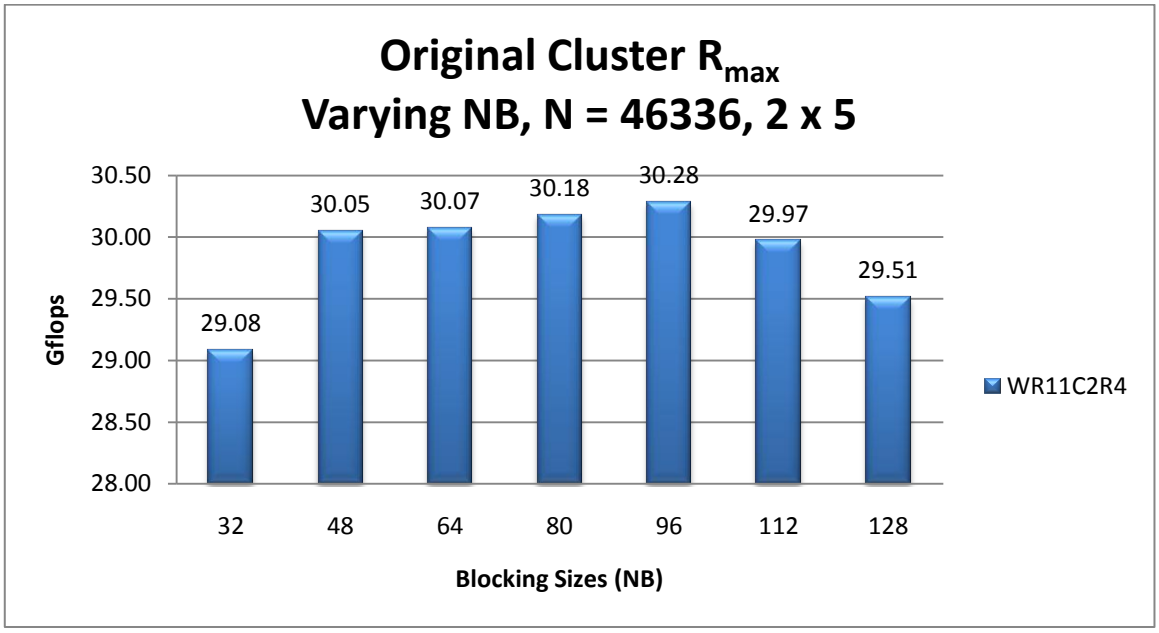


Figure 47 Original cluster, multiple NB, N=46336, 2x5

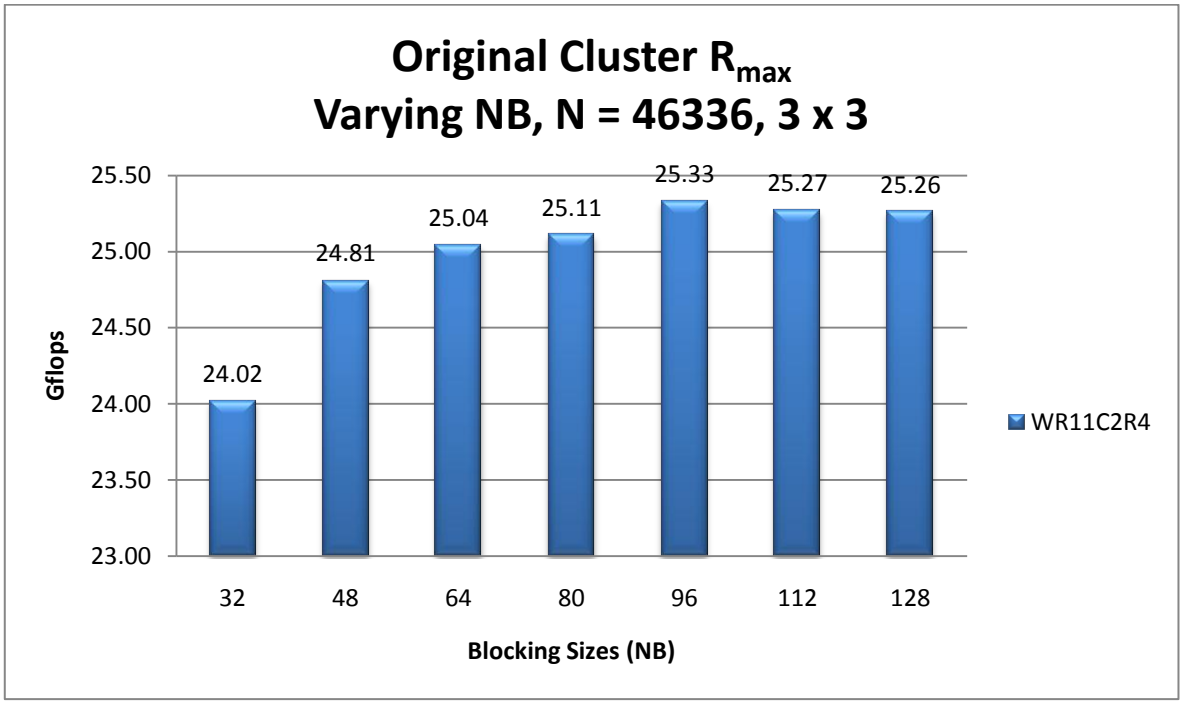


Figure 48 Original cluster, multiple NB, N=46336, 3x3

Since these processors have hyperthreading, we changed the machinefile used with HPL.dat to reflect the two processes per core. The modified file is shown in Table 20.

Table 20 Modified Machinefile for Hyperthreading

compute-0-0
compute-0-0
compute-0-1
compute-0-1
compute-0-2
compute-0-2
compute-0-3
compute-0-3
compute-0-4
compute-0-4
compute-0-5
compute-0-5
compute-0-6
compute-0-6
compute-0-7
compute-0-7
compute-0-8
compute-0-8
compute-0-9
compute-0-9

Two tests were run at maximum N = 46336, NB = 64, and a matrix of 4 x 5. Comparing the results shown in Figure 49 with those shown earlier in Figures 46 & 47 clearly shows that with hyperthreading turned on, we experienced a drop in R_{max} of ~13%. This is due to the processors having to share the 2GB of local memory on each node between the two processes being run by each core.

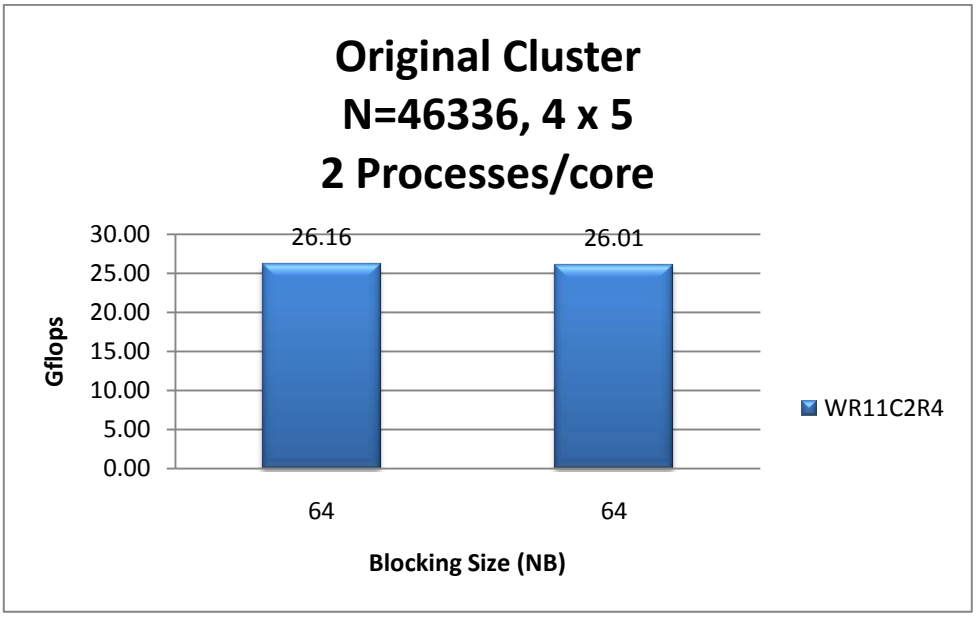


Figure 49 Original cluster, 4x5 matrix using Hyperthreading

6.3.4 WOPPR Cluster Results (10 nodes)

As stated earlier in the discussion of HPL, the major factors affecting the benchmark results are the matrix size (N), the Blocking Size (NB), and the matrix configuration (P x Q). For our initial testing of the cluster the best results were obtained with the following major parameters:

- N = 46336, an N value that used 80% of the available memory in the cluster.
- NB = 96
- (P x Q) of (2 x 5)

These settings resulted in an R_{max} of **30.28 Gflops**. This is a compEff of **45.1%.** All testing up to this point was done with the remaining tuning parameters at WR11C2R4.

The second phase of the testing is the Assay code tests. These tests create very large amounts of data (in the 100s of GB) and the original cluster frontend had a hard drive capacity of 80GB. The decision to replace the frontend was made instead of upgrading the hard drive. The specifications of the new frontend are included in Table 9 and Appendix A.

6.4 WOPPR 8-node Testing for Comparison with Apple Xserve

At the same time that we were replacing the frontend on the cluster, we were also able to obtain an Apple Xserve for testing. The Xserve had two Xeon quad core processors giving it 8 cores. To allow as much of a direct comparison as possible we removed WOPPR nodes 4 and 6 from the machinefile leaving 8 nodes (8 cores). The two nodes removed were the 3.2GHz processors.

A similar test plan was followed between the WOPPR using the new frontend and 8 nodes and the Apple Xserve that also has 8 cores. The testing and results are detailed in the following sections.

6.4.1 WOPPR Determining the Effect of Matrix Size

Starting with an N value of 41344 (approximating 80% memory usage) and a matrix distribution of 2 x 4, we ran testing to verify the overall effects of matrix sizes on the R_{max}. The results are shown in Figure 50. The R_{max} = 19.97.

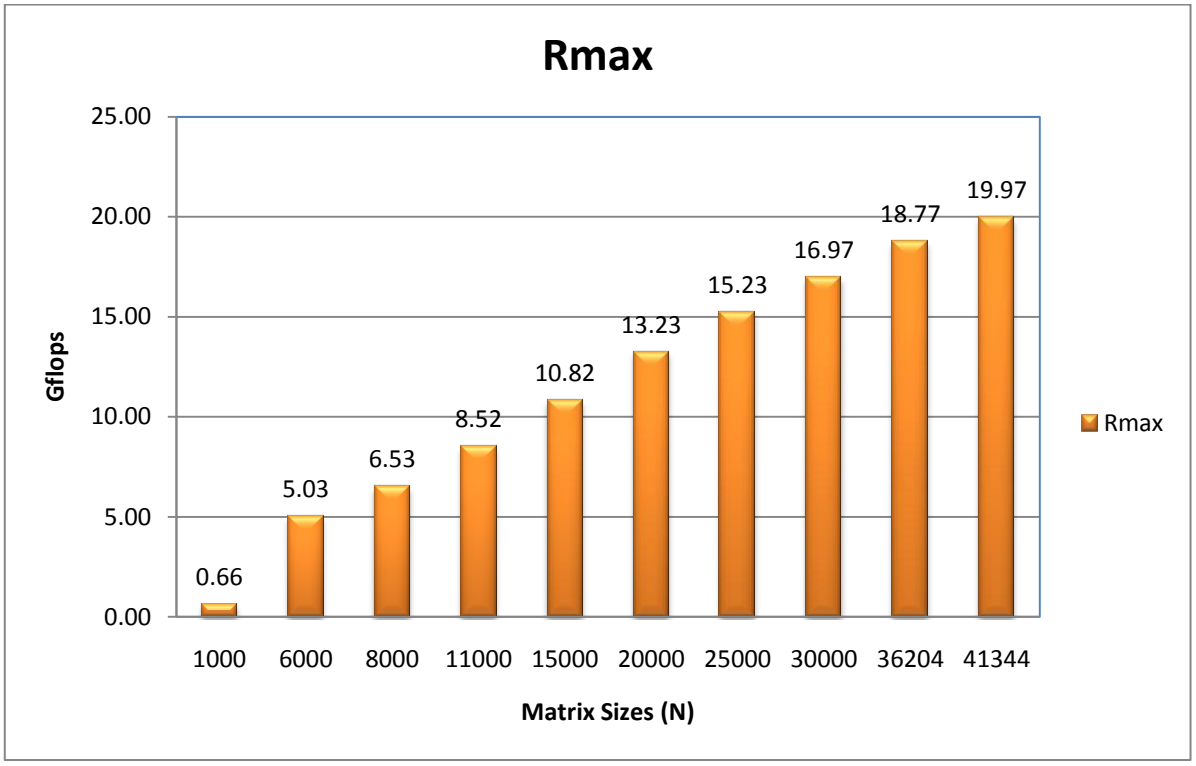


Figure 50 WOPPR determining the effect of N size.

6.4.2 WOPPR Parameter Tuning

The first step in the parameter tuning for the WOPPR was to take the maximum N = 41344 and run tests across a range of NB values to determine the maximum NB effect. The results are shown below in Figure 51. The optimum NB value is 208.

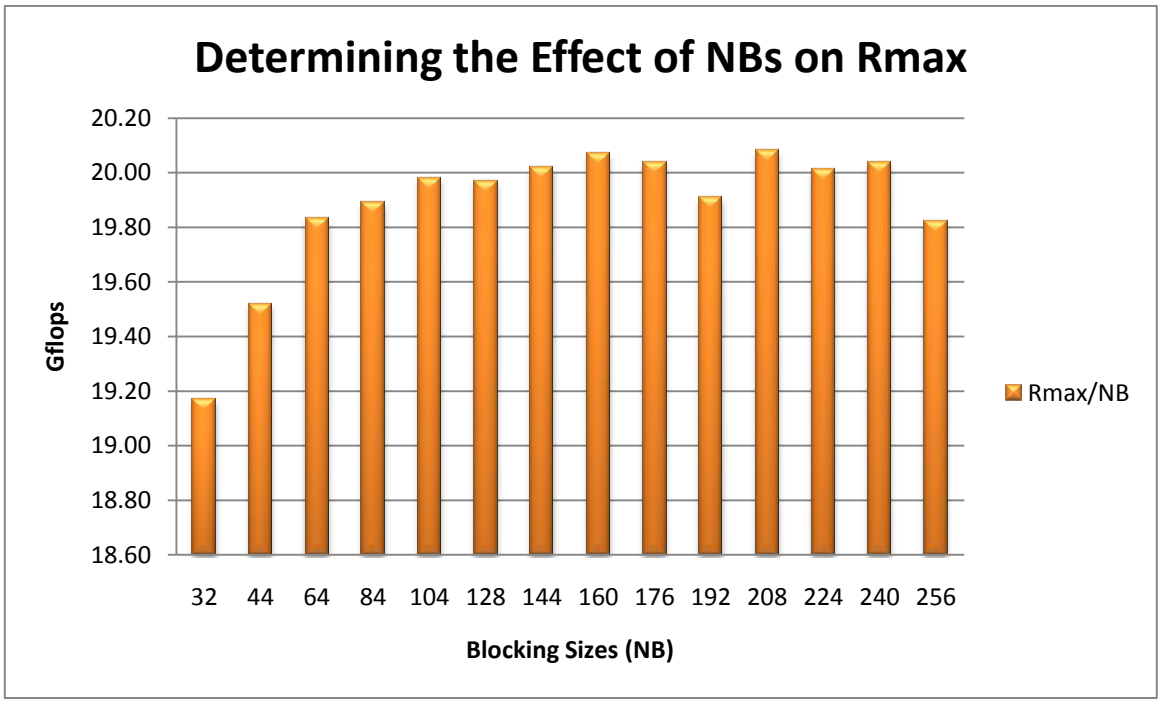


Figure 51 WOPPR determining the optimum NB value.

Using the values N = 41344 and NB = 208, the fine parameters were set up for a recursive test. The results are shown in Figure 52. The resolution of data points produced by HPL at the matrix size used was only 2 decimal places. At this resolution 7 of the nine results were identical.

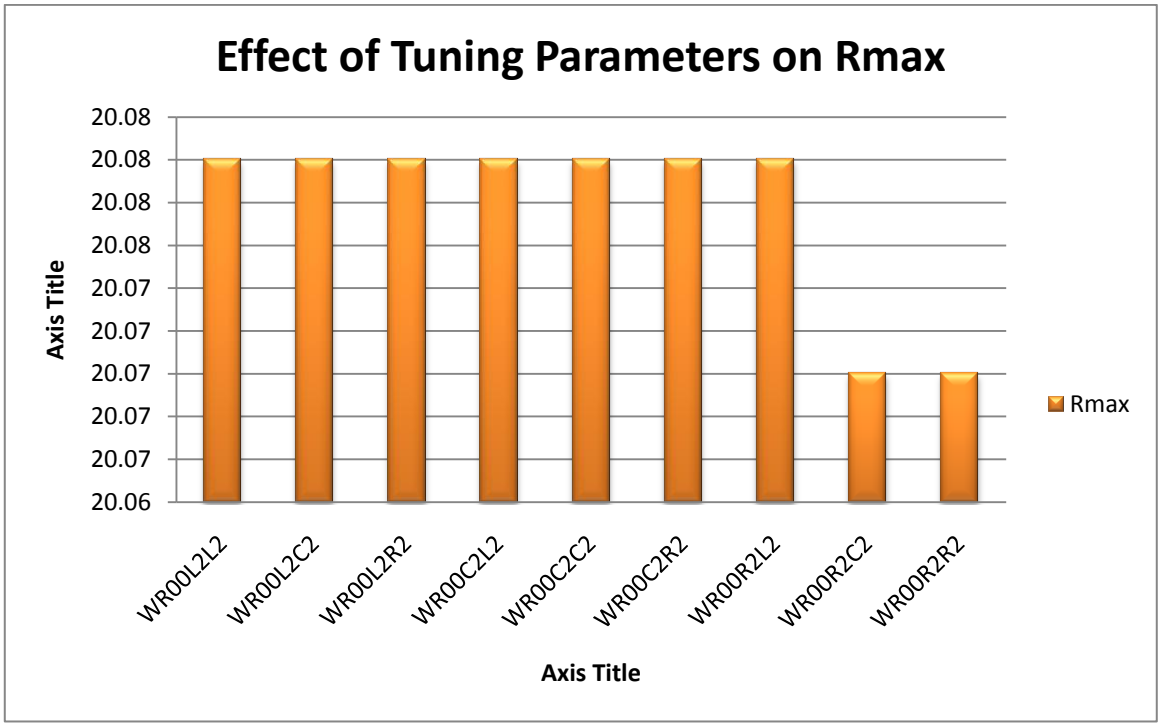


Figure 52 WOPPR parameter tuning.

The theoretical peak performance for the cluster is 54.4 Gflops as calculated below.

$$\boldsymbol{R_{peak}} = 1_{core} * 3.4GHz * 2_{FPops} = \mathbf{6.8}\ \boldsymbol{Gflops/core}$$

$$\textbf{Cluster } \mathbf{R_{peak}} = 6.8\ Gflops/core * 8\ cores = \ \mathbf{54.4}\ \boldsymbol{Gflops}$$

The measured performance (R_{max}) is 20.08 Gflops. Calculating the computer efficiency we get:

$$compEff = \ {}^{20.08}/_{54.4} * 100 = \ 36.9\%$$

6.4.3 WOPPR Assay Testing

The first set of Assay code testing runs was performed with hyperthreading turned off (through Sun Grid Engine) and the motor writing configuration turned off. Having the motor writing turned off removes the overhead of the network and allows comparison to the CPU dependent results only. The results are shown in Figure 53.

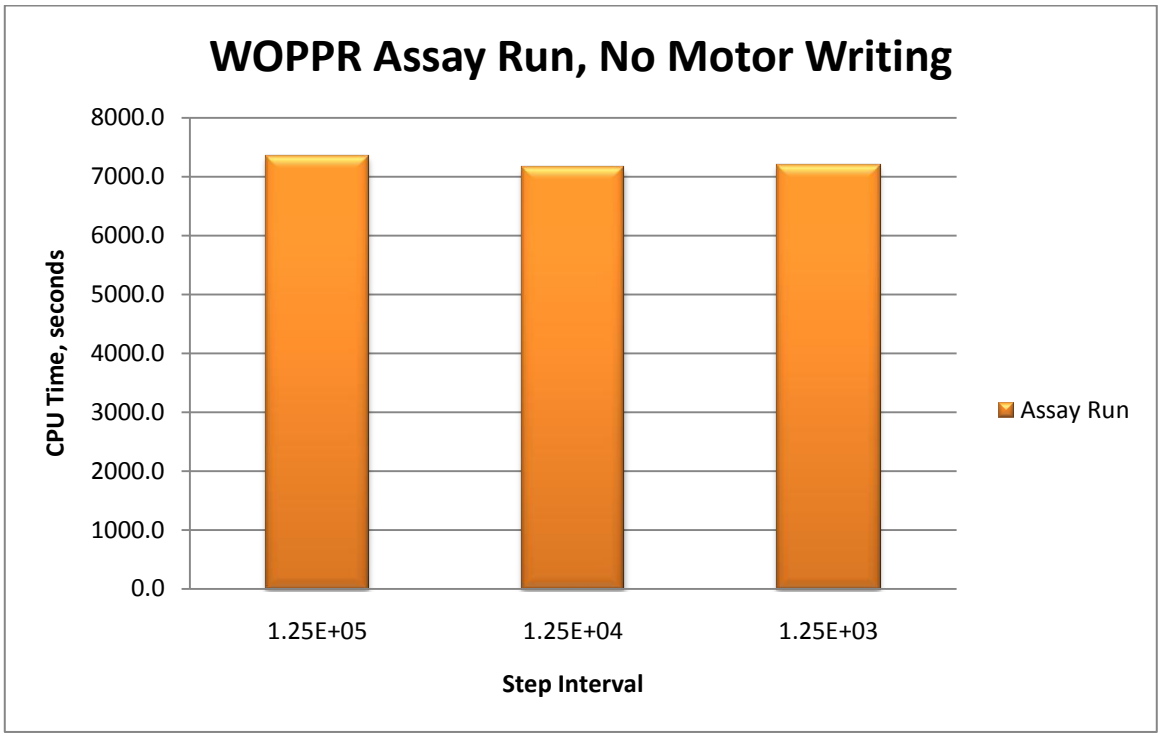


Figure 53 WOPPR Assay run, 8 nodes, no motor writing.

The following tests were done with the motor writing turned off and then with motor writing turned off. The graph in Figure 54 shows the results for the single thread tests including the total time values which incorporate the network transfer times.

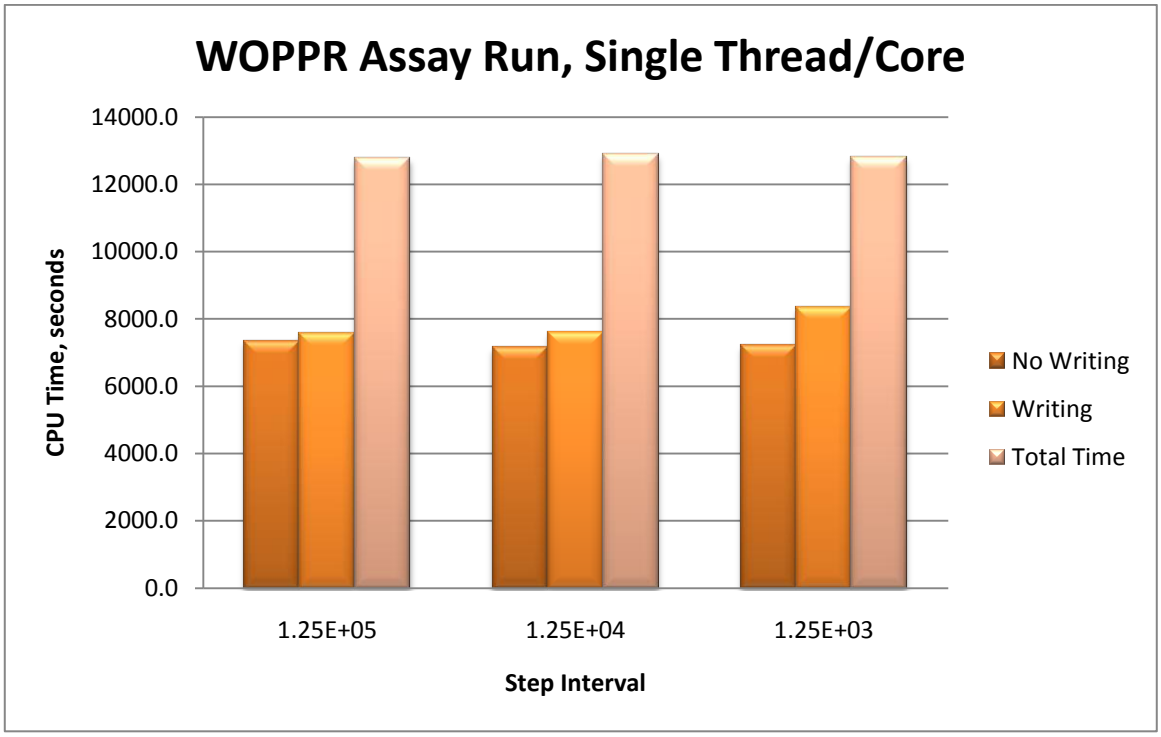


Figure 54 WOPPR Assay comparison with and without motor writing.

Hyperthreading was re-enabled in Sun Grid Engine for the following tests. The motor writing configuration was turned off for the first group and turned on for the second group. The combined chart is provided in Figure 55 for easy comparison of the data. The results also show the total test time for the runs using two threads per core. The network transfer time is included.

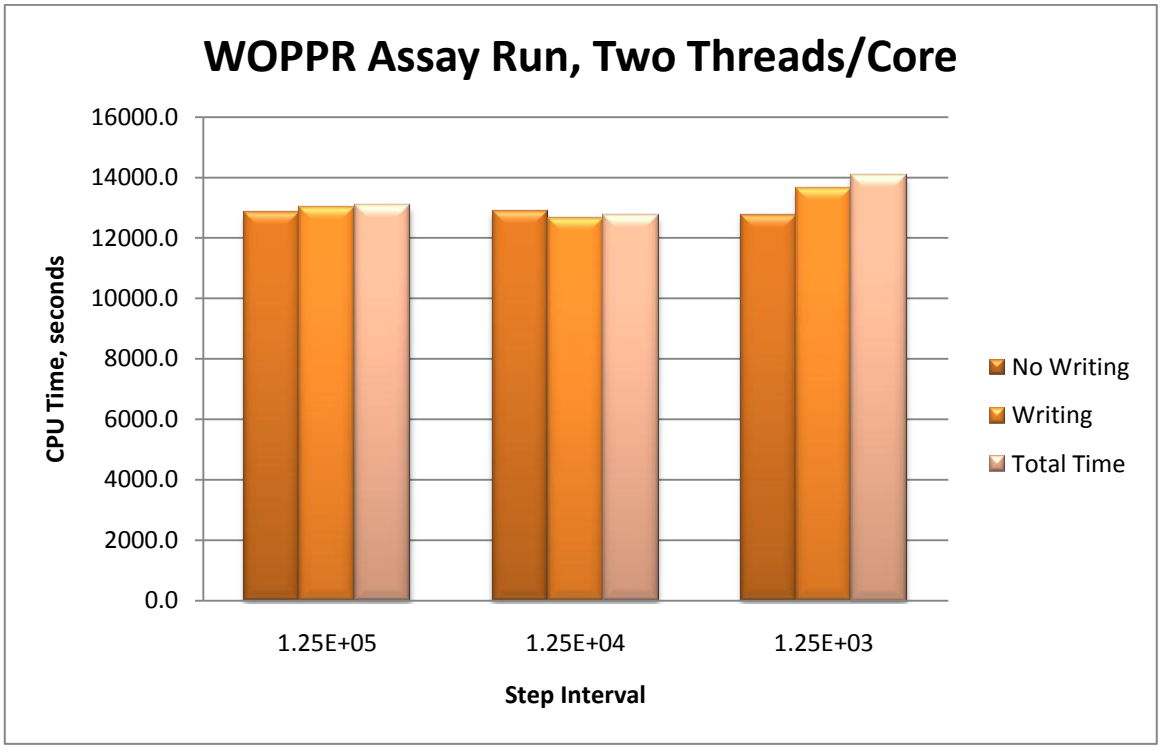


Figure 55 WOPPR Assay comparison with two threads/core.

6.4.4 WOPPR Testing Results

Using 8 nodes of the WOPPR Cluster, we achieved a peak performance (R_{max}) of 20.08 Gflops and computer efficiency (compEff) of 36.9%. This differs from the original cluster by ~17%. While the difference between the original frontend CPU and the CPU of the new frontend is ~17% this is needs more investigation to prove a direct correlation. One aspect of the continued work should be to evaluate the changes caused by recompiling the benchmark software on different systems.

6.5 Apple Xserve Testing and Results

The Xserve specifications are listed in Table 21 below. It is configured as a standalone server node running on OS X. This means that it does not have a frontend to control it but rather runs its own server software on itself. This configuration has some benefits and some drawbacks. The biggest benefits are that there are no interconnections between nodes to create bottlenecks for data transfer if all of the processes have the benefit of using the QPI interconnects for message passing and the processors share memory across the interconnects. The biggest drawback to this configuration is that the server is standalone which means that it has the overhead of running all the normal services, the server software, and has to handle the processing of jobs. Overall the benefits outweigh the drawbacks, as clearly shown in the data collected.

Table 21 Apple Xserve Specifications

Apple Xserve	
Processor	Quad Core Intel Xeon 5520
Clock Speed	2.26 GHz
Bus Speed	5.86 GT/s (QPI Interconnects)
L2 Cache	256Kb/core (2MB)
Memory	12GB
Memory Speed	1066MHz

The Xeon 5520 is a Nehalem design CPU which means that its FP peak performance was doubled from that of the P4 architecture by adding 128-bit FP ADD and FP MUL EUs on different ports working with 1 cycle throughput. This gives it a peak FP throughput for vectorized code of 2 64-bit MUL and 2 64-bit ADD operations per cycle (8 SP or 4 DP Flops). Calculating the peak performance for this Xserve server give us:

$$\boldsymbol{R_{peak}} = 1_{core} * 2.26GHz * 4_{FPops} = \mathbf{9.04}\ \boldsymbol{Gflops/core}$$

Server R_{peak} = $9.04\ Gflops/core * 8\ cores = \mathbf{72.32\ Gflops}$

6.5.1 Xserve - Determining the Effect of Matrix Size

The Xserve machine was on loan to us for testing purposes. Due to the limited time that we had available, we started the initial data runs incorporating a wide range of parameters to test. The parameters specified are shown in Table 22 below. With multiple NBMINs, PFACTs, and RFACTs in effect, the testing recursively tested all combinations of the parameters. The maximum (80%) setting for N is:

$$N = \sqrt{\frac{12 * 1024^3 * (.8)}{8}} = 35895$$

The results shown in Figure 56 are the R_{max} for the various matrix sizes tested without regards to the finer tuning parameters.

Table 22 Xserve Initial 2x4 Testing Parameters

NB	:	128			
PMAP	:	Row-major	process	mapping	
P	:	2			
Q	:	4			
PFACT	:	Left	Crout	Right	
NBMIN	:	2	4		
NDIV	:	2			
RFACT	:	Left	Crout	Right	
BCAST	:	1ring			
DEPTH	:	0			
SWAP	:	Mix	(threshold	=	64)
L1	:	transposed	form		
U	:	transposed	form		
EQUIL	:	yes			
ALIGN	:	8	double	precision	words

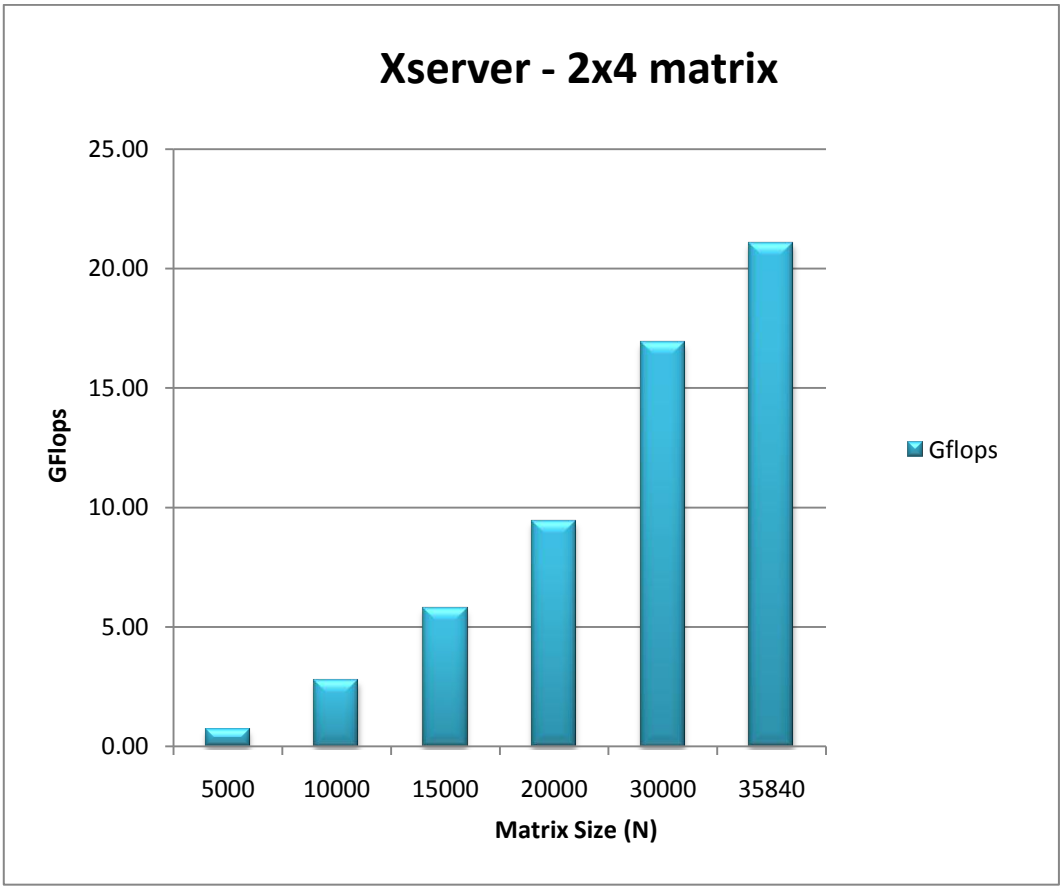


Figure 56 Apple Xserve Gflops for 2x4 matrix.

As seen in Figure 51, the 80% maximum matrix size (N) is clearly the most efficient setting.

6.5.2 Xserve Parameter Tuning

The effect of varying the finer tuning parameters was noticeably obvious for this machine. Starting at the lower N values a definite pattern showed itself alternating between values of NBMIN. These NBMIN values determine when the recursive panel factorization will stop. When the current panel being factorized has less than or equal columns to the NBMIN value then the recursion stops. As shown in Figure 57 as the NBMIN value alternates between 2 and 4 the results vary by more approximately 5%.

However, when the matrix size reaches its maximum value (80% of total memory) the effect is almost totally damped.

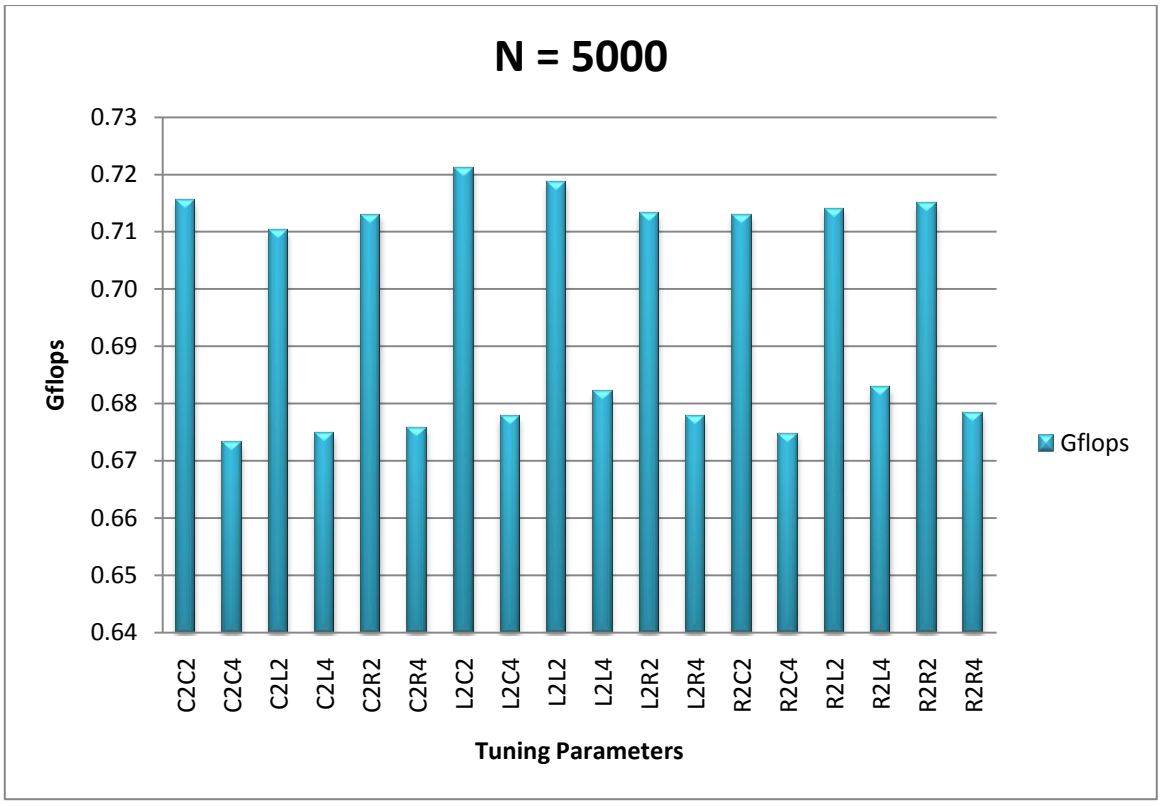


Figure 57 Apple Xserve tuning, NBMIN effect, N=5000.

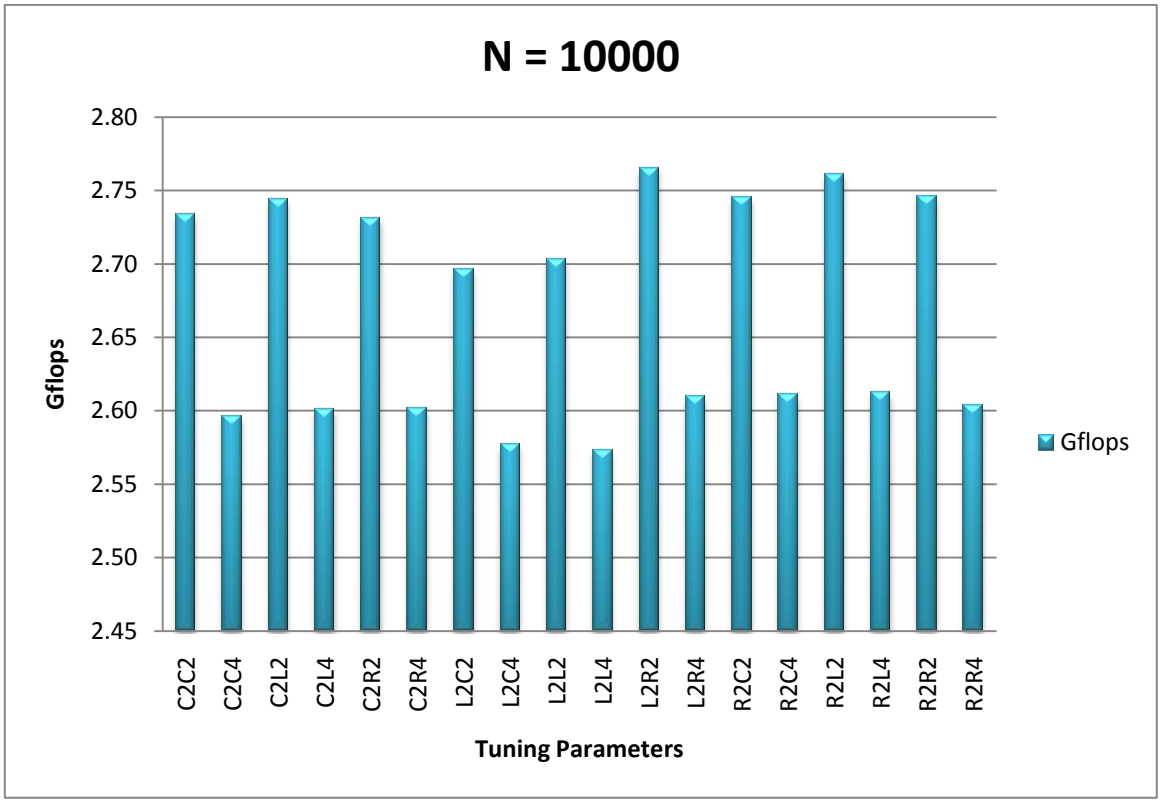


Figure 58 Apple Xserve tuning, NBMIN effect, N=10000

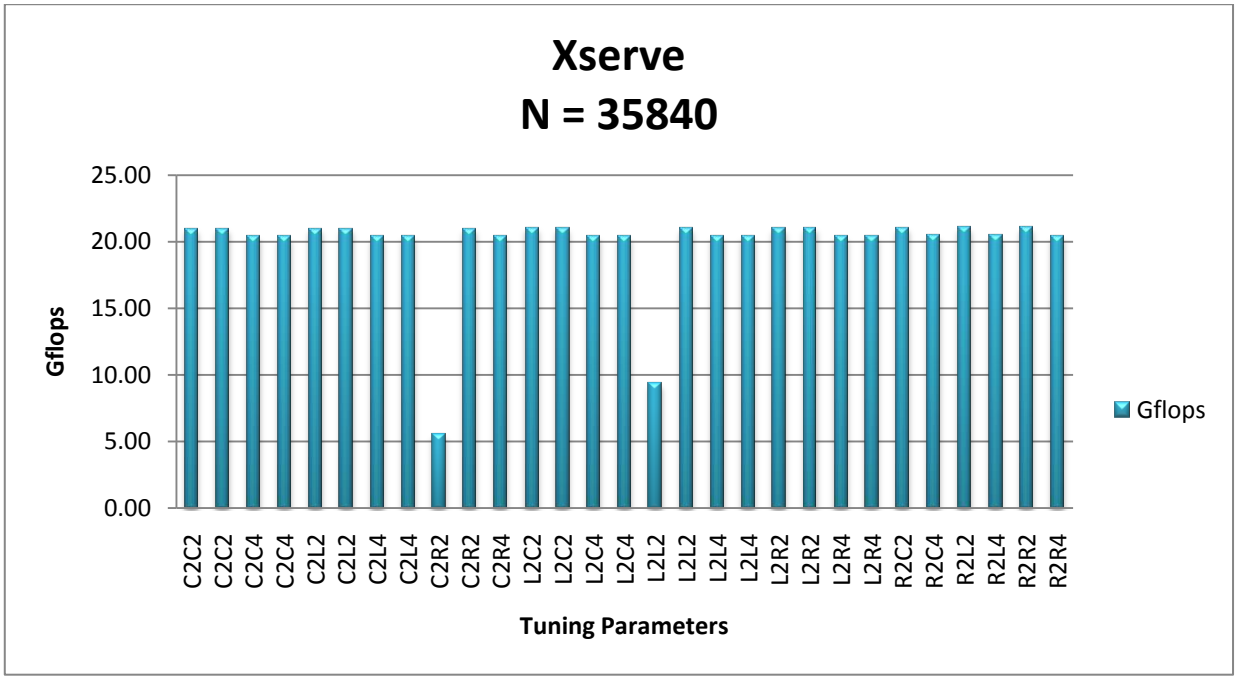


Figure 59 Apple Xserve tuning, NBMIN effect is damped.

After evaluation of all the tuning parameter combinations, the RFACT/PFACT and NDIV/NBMINs contributing to the highest R_{max} is a combination of L2L2. Most of the data points making up Figure 51 have the L2L2 combination.

The computer efficiency for the server tested on the 2 x 4 matrix is:

$$compEff = {}^{21.07}/_{72.32} * 100 = 29.1\%$$

We continued on with our test plan to see if we could identify a reason for such a low efficiency. The next set of tests used a fixed NB of 128 and the maximum N of 35840 while varying the matrix (P x Q) configuration through 2 x 4, 1 x 4, and 4 x 1.

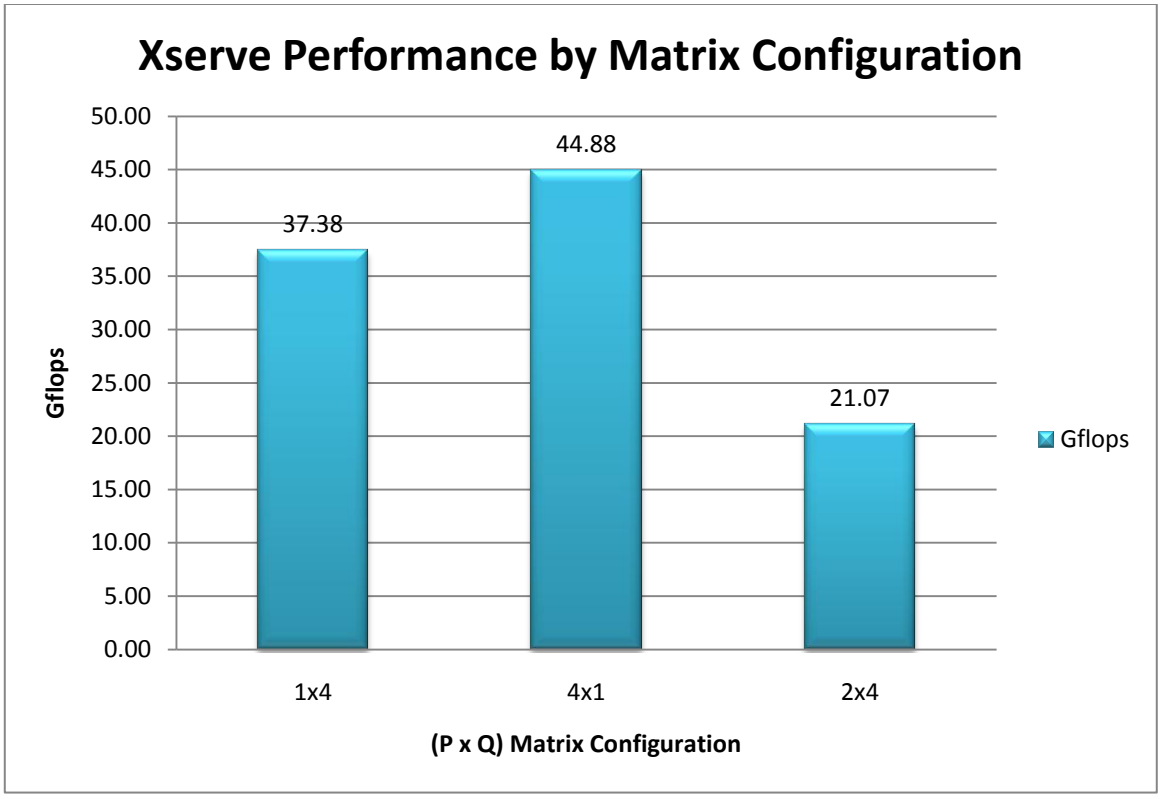


Figure 60 Apple Xserve performance by matrix configuration.

Figure 60 shows a remarkable leap in R_{max} for both the 1 x 4 and the 4 x 1 configurations. This is possibly caused by having all of the processors and memory on the same motherboard with all message passing occurring through the interconnects.

The HPL algorithm distributes data onto a P-by-Q grid of processes. Each CPU in the Xserve has 4 cores so a PxQ distribution of 1 x 4 or 4 x 1 would only span one CPU in one case and only 2 cores of each CPU in the other case. In the first instance, the efficiency would increase due to the fact that communications are not needed between CPUs and in the second case, efficiency would increase due to two processes per core having full use of the QPI links and the full amount of RAM

The results for the 1x4 and 4x1 matrix distributions are higher than theoretical limits for the specified matrix sizes. This could be due to the problems associated with configuring HPL to run on a single server node. Since the server node is no longer available for testing, no validation can be completed within the time limits of the project. Continuing work should include testing of a single server node configuration to validate HPL response. The computer efficiency measured now is:

$$compEff = {}^{44.88}/_{72.32} * 100 = 62.1\%$$

The Xeon processors have hyperthreading giving the server a total of 16 process threads (2 threads/core). Since this server is self contained and not a frontend/node configuration, there is no machinefile to edit. The threads are handled by using openMP directives to set the number of threads available. The results for the 16-thread tests are shown below in figure 61. The 1 x 4 matrix has been included for direct comparison. The difference in efficiency is caused by the memory sharing that takes place in the HPL process. This creates a bottleneck to the throughput.

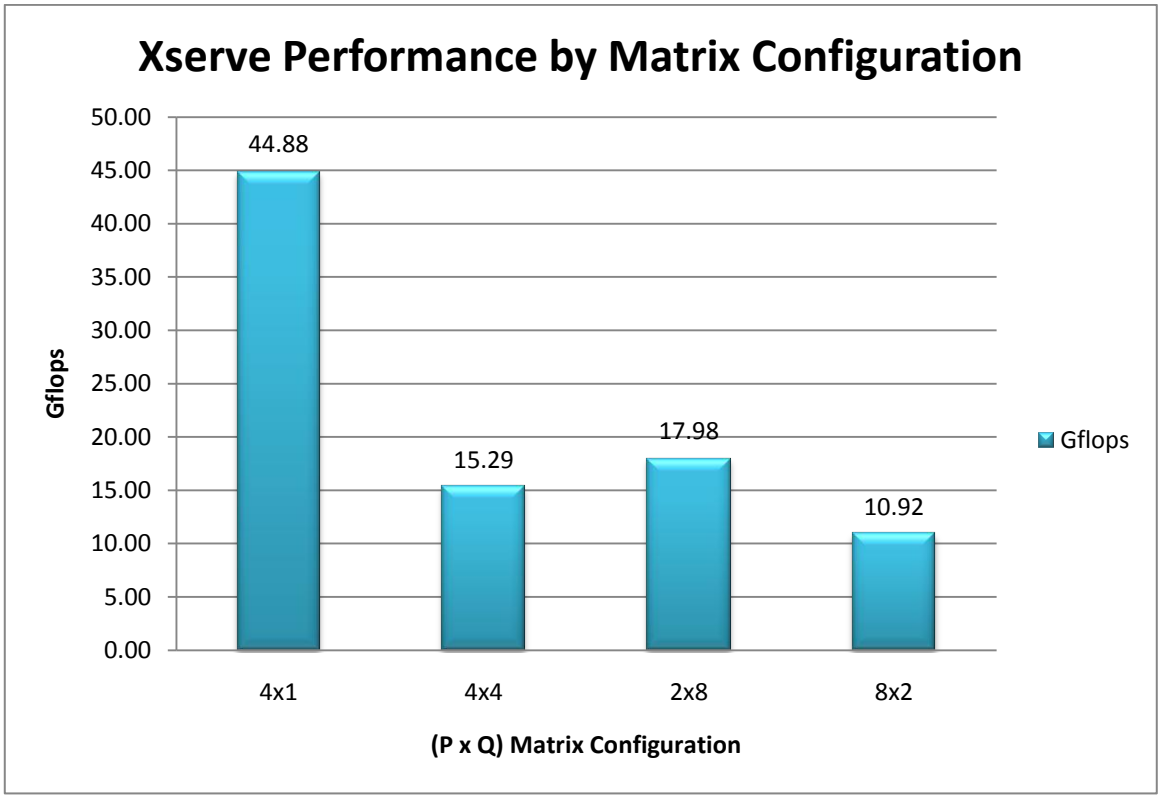


Figure 61 Apple Xserve 16-thread performance by matrix configuration.

6.5.3 Xserve Assay Testing

The Xserve server was used to make 3 test runs with different step time values and motor configuration writing turned off. The results are shown in Figure 63.

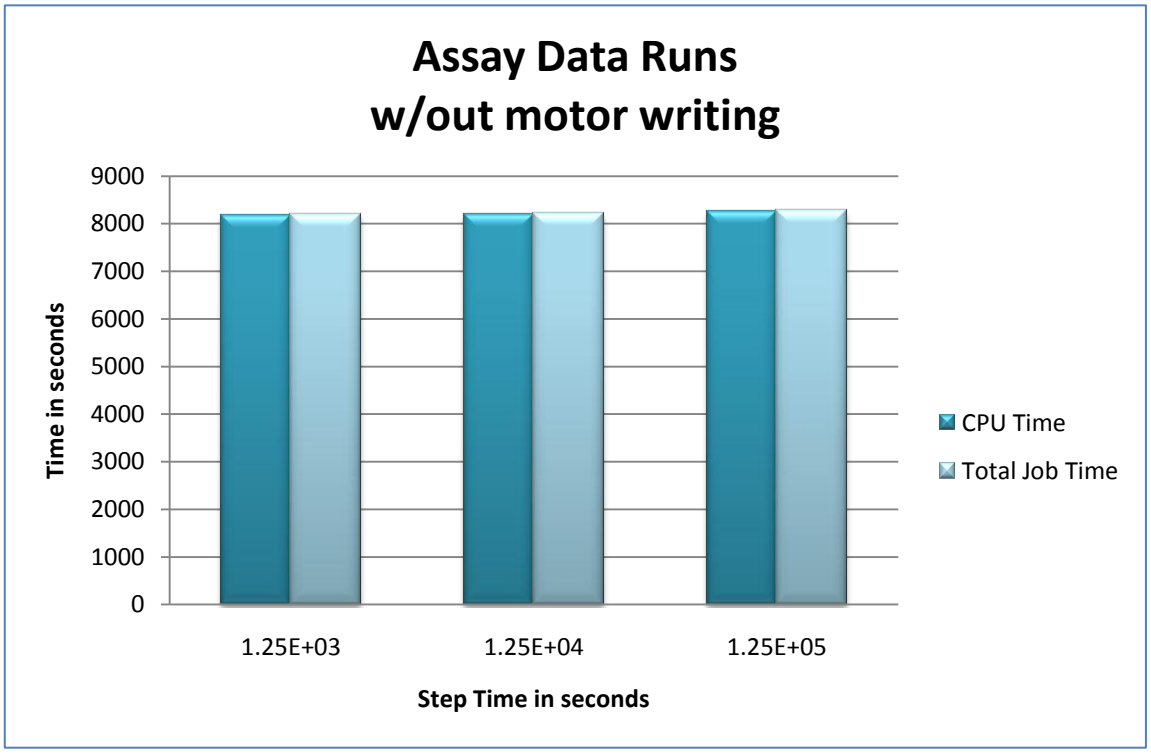


Figure 62 Apple Xserve Assay testing results.

6.5.4 Xserve Results Summary

The Xserve has a calculated R_{peak} of 72.32 Gflops. During the HPL benchmark testing we were able to tune the tests to obtain an R_{max} = 44.88 Gflops and a compEff = 62.1%.

6.6 Conclusion

After completing the testing of the WOPPR and the Xserve we had three groups of data for comparison; 1) The WOPPR Cluster with the original frontend, 2) The WOPPR Cluster with the new frontend, and 3) The Xserve. The theoretical R_{peak} and the measured R_{max} are shown in Table 23. The summary of data from the Assay code is provided below in Table 24.

The expected difference between the testing done with WOPPR on the original frontend using all 10 nodes and the new frontend using only 8 nodes is ~19%. The actual difference between frontend configurations was ~33.7%. The difference of 14.7% was verified with additional data collection. The research concerning this difference should be conducted in continuing work on this project. Some possible reasons for the discrepancy are:

- Differences in compiling HPL with different CPU architectures.
- Differences in compiling GotoBLAS2 with different CPU architectures.
- Differences in compiling MPICH2 with different CPU architectures.

The real-world testing results involving the Assay code are shown in Table 24 below. As shown, the WOPPR cluster provided better results than the Xserve when operating in single thread mode due to the higher clock speeds of the node processors. However, when run with hyperthreading enabled (2 processes per core) the added overhead involved with shared memory on the WOPPR increased the data run by more than 80%.

Table 23 Testing Summary for HPL Data

HPL Data	Rpeak (Gflops)	Rmax (Gflops)/ compEff(%)
WOPPR (original frontend)	67.2	30.28/45.1%
WOPPR (new frontend)	54.4	20.08/36.9%
Xserve	72.3	44.88/62.1%

Table 24 Testing Summary for Gliding-Assay Data

Assay Data	Peak (minimum time in seconds)	Average (time in seconds)
WOPPR (no hyperthreading)	7,149	7,236
WOPPR (hyperthreading)	12,754	12,823
Xserve	8,183	8,220

Chapter 7: Conclusion and Recommendations

A cluster computer (the WOPPR) was successfully built from 'outdated' reuse computers being recycled at WPI. The computational speed of the cluster was proven to be faster than a mainstream server node when operating in single thread mode on single process multiple data programming for a computational physics application. This project has demonstrated that there are viable and productive uses for clusters such as the WOPPR.

The decision to use a cluster built from reuse computers would have to be made on a case-by-case basis depending on variables such as the type of programming to be used, the space available, the power available, and the number of process threads required.

The cost of using reuse computers is limited to the peripheral items that would be added on when building the cluster, such as a new network switches, cables, etc. , and the electrical power to run the cluster. In comparison with the purchase of a new cluster computer, the initial costs for a reuse cluster are far more economical.

When planning a cluster, the type of programming to be run is an important consideration. An older technology such as the Pentium 4's used in the WOPPR are ideal for single instruction multiple data processing where multiple cases of the same instruction need to run simultaneously. For a cluster such as this, the main limiting factor would most likely be the space considerations in the area that the cluster is built. As shown in Figure 63, over 90% of registered rocks clusters have less than 170 processors [32]. The top 7 outliers are not included in Figure 63.

The Rocks Cluster operating system was chosen for the WOPPR since it is a robust platform for running clusters and provides the tools necessary for running parallel as well as serial code. The included Sun Grid Engine is a popular and functional front-end management system for submitting and monitoring cluster jobs.

While this project was successful in setting up, configuring, and testing a cluster capable of continued use in computational research, there are some areas of future research that were either beyond the scope of the project or outside of the time allotted for the project. These items are recommended for continued work.

The first recommendation is to test and evaluate the changes caused by recompiling the HPL benchmark software on different CPU architectures, with attention to CPU speed, front side bus speed, L2 cache, and the speed of the RAM.

The second recommendation is to test and quantify the effects on HPL results, of recompiling the linear algebra system and message passing software on different CPU architectures

The final recommendation is to conduct further testing of the HPL benchmarks on single server nodes with multiple processors capable of hyperthreading to determine the behavior of the benchmark software. This should include varying matrix distribution configurations.

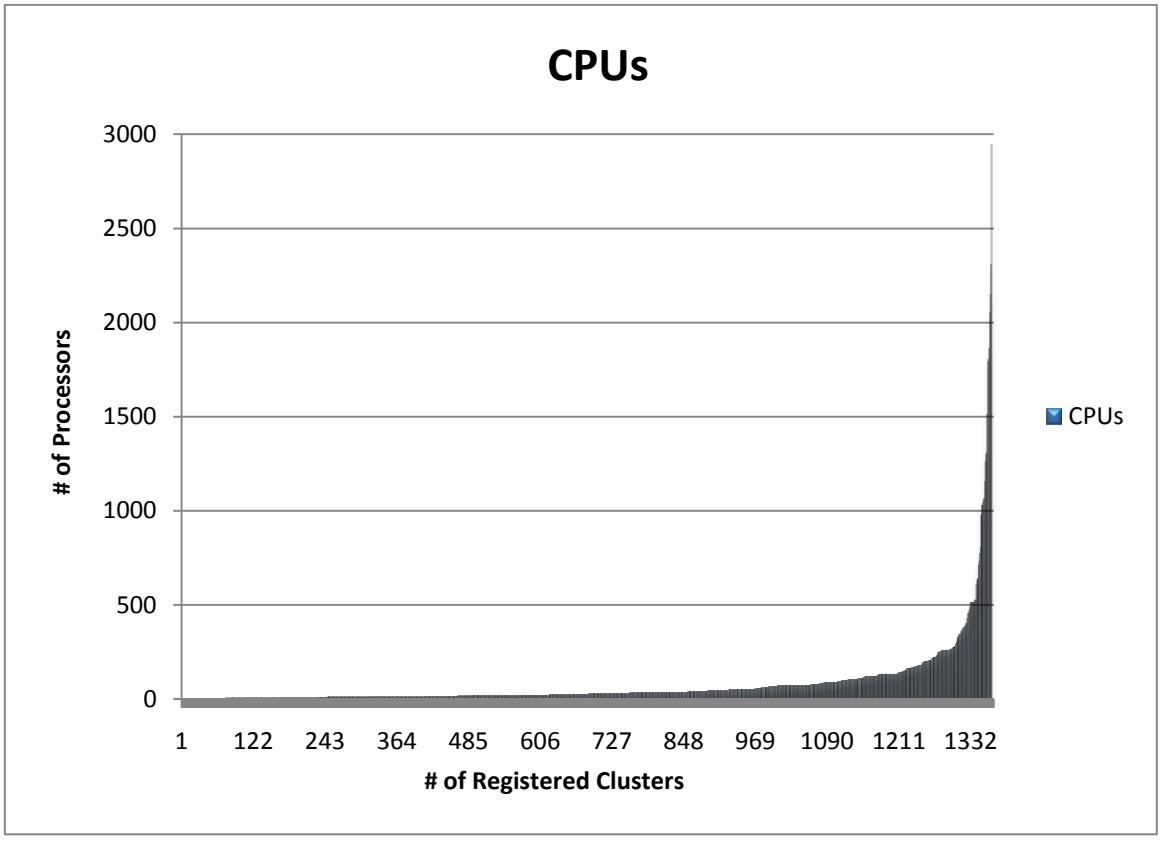


Figure 63 Registered Rocks Cluster Sizes

References

[1] History of Cluster Computing, http://cunday.blogspot.com/2009/01/history-of-cluster-computing.html, (accessed 7/7/2010)
[2] Beowulf Project Overview, http://www.beowulf.org/overview/history.html, (accessed 7/20/2010)
[3] Top 500 Supercomputers, http://www.top500.org/lists, (accessed 7/7/2010)
[4] Intel Corporation, http://h41201.www4.hp.com/tradein/html/910/dk/da/intel_case_study.pdf
[5] Gartner Says More than 1 Billion PCs In Use Worldwide and Headed to 2 Billion Units by 2014, http://www.gartner.com/it/page.jsp?id=703807 , (accessed 7/20/2010)
[6] Gartner press releases, April 15, 2009, July 16, 2009, October 14, 2009, January 13, 2010, http://www.gartner.com/it/page.jsp?id=939015, http://www.gartner.com/it/page.jsp?id=1076912, http://www.gartner.com/it/page.jsp?id=1207613, http://www.gartner.com/it/page.jsp?id=1279215, (accessed 7/29/2010)
[7] Statistics on the Management of Used and End-of-Life Electronics, July 2008, http://www.epa.gov/epawaste/conserve/materials/ecycling/manage.htm, (accessed on 7/29/2010)
[8] Basel Conference Addresses Electronic Wastes Challenge, Press release from United Nations Environment Programme, http://www.unep.org/documents.multilingual/default.asp?DocumentID=485&ArticleID=5431&l=en, (accessed on 7/29/2010)
[9] Moor's Law, http://www.intel.com/technology/mooreslaw/index.htm, (accessed 10/6/2010)
[10] U.S. Environmental Protection Agency, Wastes-Resource Conservation-Common Wastes and Materials-eCycling website, http://www.epa.gov/osw/conserve/materials/ecycling/donate.htm#local, (accessed 8/9/2010)
[11] XServe Performance, http://www.apple.com/xserve/performance.html, (accessed on 1/26/2010)
[12] U.S Energy Information Administration Independent Statistics and Analysis http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html
[13]G. F. Pfister, *In Search of Clusters*, *2nd Edition*, Prentice Hall, 1998.

[14] Prime95, http://files.extremeoverclocking.com/file.php?f=103
[15] OCCT, http://www.ocbase.com/perestroika_en/index.php
[16] http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0438741, (accessed 9/30/2010)
[17] http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0721623, (accessed 9/30/2010)
[18] http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=1032778, (accessed 9/30/2010)
[19] Tools and Techniques for Easily Deploying Manageable Linux Clusters, http://www.rocksclusters.org/rocks-doc/papers/ieee-cluster-2001/paper.pdf, (accessed 1/20/2010)
[20] Rocks Cluster homepage, http://www.rocksclusters.org/wordpress/?page_id=114, (accessed 10/4/2010)
[21] Rocks Cluster downloads, http://www.rocksclusters.org/wordpress/?page_id=114, (accessed 10/5/2010)
[22] The ganglia distributed monitoring system: design, implementation, and experience. Massie, Chun, and Culler, February 2003
[23] Open Source Tripwire, http://sourceforge.net/projects/tripwire/, (accessed 10/5/2010)
[24] Chkrootkit, http://www.chkrootkit.org/, (accessed 10/5/2010)
[25] MPI-2 Message Passing Interface Forum, http://www.mpi-forum.org/, (accessed 9/16/2010)
[26] Kazushige Goto, Author of GotoBLAS, http://en.wikipedia.org/wiki/Kazushige_Goto, (accessed 10/6/2010)
[27] Texas Advanced Computing Center, http://www.tacc.utexas.edu/tacc-projects/gotoblas2/downloads/, (accessed 2/15/2010)
[28] Top500 SuperComuters, http://www.top500.org/, (accessed 10/5/2010)
[29] Netlib.org HPL software page, http://www.netlib.org/benchmark/hpl/software.html, (accessed 10/6/2010)

[30] Intel Math Kernel Library, http://software.intel.com/en-us/intel-mkl/, (accessed 10/6/2010)
[31] K. Singh, E. Ipek, S. A. McKee, B.R. de Supinski, M. Schulz, and R. Caruana. Predicting parallel application performance via machine learning approaches. Concurrency And Computation: Practice and Experience, 19(17): 2219-2235, 2007.
[32] The Rocks Cluster Register, http://www.rocksclusters.org/rocks-register/index.php?sortby=CPUs&sortorder=down, (accessed 10/19/2010)

APPENDIX

APPENDIX A: Computer Specifications

Node	woppr-0-0	woppr-0-1	woppr-0-2	woppr-0-3	woppr-0-4	woppr -0-5
Bios Version	A07	A07	A07	A11	A05	A07
Bios Date	3/31/2006	3/31/2006	3/31/2006	11/30/2006	10/13/2005	3/31/2006
Service Tag	CDXF0B1	FCXF0B1	GDXF0B1	7GXF0B1	BH0CY81	JCXF0B1
Processor	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640
Clk Spd	3.4 GHz	3.4	3.4	3.4	3.2	3.4
Bus Spd	800 MHz	800 MHz	800 MHz	800 MHz	800 MHz	800 MHz
L2 Cache	2 Mb	2 Mb	2 Mb	2 Mb	2 Mb	2 Mb
Proc ID	0F43	0F43	0F43	0F43	0F43	0F43
Memory	2Gb	2Gb	2Gb	2Gb	2Gb	2Gb
Mem Spd	533 MHz	533 MHz	533 MHz	533 MHz	533 MHz	533 MHz
Dim 1	1024MB	1024MB	1024MB	1024MB	1024MB	1024MB
Dim 2	1024MB	1024MB	1024MB	1024MB	1024MB	1024MB
Node	**Original Frontend**	**New Frontend**	**woppr-0-6**	**woppr-0-7**	**woppr-0-8**	**woppr-0-9**
Bios Version		401	A01	A07	A07	A07
Bios Date		12/30/2009	5/24/2005	3/31/2006	3/31/2006	3/31/2006
Service Tag	BW75181		GFTFW71	DFXF0B1	2FXF0B1	1XWF0B1
Processor	Pentium 4 Prescott Dt, 640	Core 2 Quad Q8400	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640	Pentium 4 Prescott Dt, 640
Clk Spd	3.2 GHz	2.66GHz	3.2	3.4	3.4	3.4
Bus Spd	800 MHz	1333 MHz	800 MHz	800 MHz	800 MHz	800 MHz
L2 Cache	2 MB	4096 KB	1Mb	2 Mb	2 Mb	2 Mb
Proc ID	0F43	1067A	0F41	0F43	0F43	0F43
Memory	2 Gb	4 Gb	2Gb	2Gb	2Gb	2Gb
Mem Spd	533 MHz		533 MHz	533 MHz	533 MHz	533 MHz
Dim 1	512 MB		1024MB	1024MB	1024MB	1024MB
Dim 2	512 MB		1024MB	1024MB	1024MB	1024MB
Dim 3	512 MB					
Dim 4	512 MB					

APPENDIX B: Extend-Compute.xml File

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<description>

        A skeleton XML node file. This file is a template and is intended
        as an example of how to customize your Rocks cluster. Kickstart XML
        nodes such as this describe packages and "post installation" shell
        scripts for your cluster.

        XML files in the site-nodes/ directory should be named either
        "extend-[name].xml" or "replace-[name].xml", where [name] is
        the name of an existing xml node.

        If your node is prefixed with replace, its instructions will be used
        instead of the official node's. If it is named extend, its directives
        will be concatenated to the end of the official node.

</description>


<changelog>
</changelog>

<main>
        <!-- kickstart 'main' commands go here -->
</main>

<pre>
        <!-- partitioning commands go here -->
</pre>


<!-- There may be as many packages as needed here. Just make sure you only
     uncomment as many package lines as you need. Any empty <package></package>
     tags are going to confuse rocks and kill the installation procedure
-->
<package> intel-node-libs </package>
<package> intel-node-mkl-libs </package>
<!-- <package> insert 3rd package name here and uncomment the line</package> --
>


<post>
        <!-- Insert your post installation script here. This
        code will be executed on the destination node after the
        packages have been installed. Typically configuration files
        are built and services setup in this section. -->

        <!-- WARNING: Watch out for special XML chars like ampersand,
        greater/less than and quotes. A stray ampersand will cause the
        kickstart file building process to fail, thus, you won't be able
        to reinstall any nodes. It is recommended that after you create an
        XML node file, that you run:

                xmllint -noout file.xml
        -->

        <eval shell="python">
```

```
                <!-- This is python code that will be executed on the
                frontend node during kickstart file generation. You may contact
                the database, make network queries, etc.  These sections are
                generally used to help build more complex configuration
                files.  The 'shell' attribute is optional and may point to any
                language interpreter such as "bash", "perl", "ruby", etc.
                By default shell="bash".  -->

        </eval>

</post>

</kickstart>
```

APPENDIX C: SMC EZNET-16SW Network Switch

SMC Networks

Unmanaged Switch

EZNET-16SW / EZNET-24SW

EZNET™ 16 and 24-Port 10/100Mbps Standalone Unmanaged Layer 2 Switches

OVERVIEW

This EZNET series offers 16- and 24-port 10/100Mbps standalone, unmanaged switches. Designed specifically for home and small office applications, these switches feature auto-negotiation, auto-MDIX ports for easy network expansion and provide full bandwidth capability on each port. With fully auto-negotiating 10/100 ports, the speed and duplex modes of attached devices are instantly detected, allowing the switches to auto-configure for optimal performance.

These switches offer flexibility for any business while ensuring feature-rich and value-added support. Every port can automatically adjust and link to either a PC station or another switch, no need for crossover cables. Rackmountable or desktop, these switches are ideal for any office or addition to existing network workgroups.

FEATURES	BENEFITS
3.2 or 4.8Gbps aggregate bandwidth; speeds up to 200Mbps per port	Plug-and-play—no complicated setup procedures or cabling requirements
Filtering & forwarding all full-wire speed on all ports	Integrates Ethernet and Fast Ethernet networks
Buffered store-and-forward transmission method prevents propagation of bad packets	At-a-glance LEDs show port status and monitor network activity
Back pressure and flow control prevent packet loss under heavy load.	Auto-negotiation - chooses optimal speed and operating mode for attached devices
Sturdy steel construction; rackmountable	Address learning probes the network and automatically configures the switch

www.smc.com

SMC Networks

TECHNICAL SPECIFICATIONS — EZNET-16SW / EZNET-24SW

PORTS
- EZNET-24SW
  - 24 RJ-45 ports, auto-sensing
- EZNET-16SW
  - 16 RJ-45 ports, auto-sensing

LED
- Power- one
- Link/Activity - Per port
- 10/100 - Per port

NETWORK INTERFACE
- 10Base-T RJ-45; Category 3,4,5 UTP cable
- 100Base-tx RJ-45; Category 5 UTP cable

POWER CONSUMPTION
- EZNET-24SW - 40 Watts
- EZNET- 16SW - 13.2 Watts

POWER REQUIREMENTS
- 100 to 240V
- 50 to 60 Hz

HEAT DISSIPATION
- 137 BTU/hr max@100-240 VAC

DIMENSIONS
- 10.7 x 6.5 x 1.8 in
- 27.2 x 16.5 x 4.6 cm

WEIGHT
- EZNET-24SW
  - 3.3 Lbs / 1.5 kg
- EZNET-16SW
  - 3.2 Lbs / 1.45 kg

HUMIDITY
- 10% 90% (non-condensing)

TEMPERATURE
- Operating: 32° to 104° F / 0° to 40° C
- Storage: - 40° to 158° F / - 40° to 70° C

WARRANTY
- Limited Lifetime

MAC ADDRESS TABLE
- EZNET-16SW - 2k
- EZNET-24SW - 8k

MEMORY BUFFER
- EZNET-16SW - 768KBytes
- EZNET-24SW - 2560KBytes

STANDARDS
- IEEE802.3
- IEEE802.3u
- ISO/IEC 8802-3

IMMUNITY
- IEC 1000-4-2,3,4,6

EMI
- FCC Class B
- CE Mark
- CISPR Class B

Contact

North America
38 Tesla
Irvine, CA 92618
1-800-SMC-4YOU
24/7 Technical Support

Europe/ Africa
Fructuos Gelabert 6-8
08970 Sant Joan Despí
Barcelona, Spain

Check www.smc.com for your local country contact information

www.smc.com

APPENDIX D: Tripp-Lite Rackmount Surge Suppressor, Model IBAR12-20ULTRA

OVERVIEW	
Intended Application	Computer Networks
SYSTEM OVERVIEW	
Voltage compatibility (VAC)	120
OUTPUT	
Frequency compatibility	50 / 60 Hz
Output watts	2400
Output volt amp capacity (amps)	20
Outlet quantity / type	12(2 front NEMA 5-15R/ 10 rear NEMA 5-20R)
INPUT	
Recommended Electrical Service	120V (110-125V)
Input connection type	NEMA 5-20P input plug
Input cord length (ft.)	15
LEDS ALARMS & SWITCHES	
Switches	Red illuminated power switch controls power to all outlets
Diagnostic LEDs	PROTECTION PRESENT (green), LINE FAULT (red) and LINE OK (green)
Locking switch cover	Transparent locking switch cover prevents accidental shutoff
SURGE / NOISE SUPPRESSION	
AC suppression joule rating	3840
AC suppression response time	NM = 0 ns. CM = <1 ns
Protection modes	Includes full normal mode (H-N) and common mode (N-G / H-G) line surge suppression
Clamping voltage (RMS)	140
AC suppression surge current rating	96,000 amps
AC suppression components used	Metal oxide varistors, toroidal balanced chokes and VHF capacitors
Safe thermal fusing	Prevents unsafe conditions during extreme extended overvoltages and catastrophic occurences
UL1499 let through rating	330V - UL Verified
EMI / RFI filtering	40-80 dB
Immunity	Conforms to IEE 587 / ANSI C62.41
DATALINE SURGE SUPPRESSION	
Cable (Coax) Protection	No
Network (Ethernet) Protection	No
PHYSICAL	
Shipping weight (lbs)	6.3
Unit Dimensions (HWD/in)	1.75 x 17.5 x 4
Shipping Dimensions (HWD/in)	2.25 x 20.5 x 9.25
Unit weight (lbs)	6.3
Material of construction	Metal
Mounting accessories included	Included mounting flanges are pre-installated for 1U rackmount installation and can be re-configured to support a variety of mounting options
Housing Color	Black
Receptacle Color	Black
AC line cord color	Black
Style	Rack Mount
Form factors supported	1U rackmount, 0U vertical rackmount, wallmount and under-counter mounting supported - detachable flanges can be re-configured to support a variety of additional mounting options
SPECIAL FEATURES	
Right-Angle Plug	Yes
CERTIFICATIONS	
UL1449 3rd Edition (AC Suppression)	UL1449 3rd EDITION
UL1283 (EMI Filter)	UL1283
cUL / CSA (Canada)	cUL
Approvals	Exceeds IEEE 587 category A&B specifications

APPENDIX E: HPL Make File Configuration

```
[ctclark@woppr hpl]$ cat Make.x86_64
#
#  -- High Performance Computing Linpack Benchmark (HPL)
#     HPL - 1.0a - January 20, 2004
#     Antoine P. Petitet
#     University of Tennessee, Knoxville
#     Innovative Computing Laboratories
#     (C) Copyright 2000-2004 All Rights Reserved
#
#  -- Copyright notice and Licensing terms:
#
#  Redistribution  and  use in  source and binary forms, with or without
#  modification, are  permitted provided  that the following  conditions
#  are met:
#
#  1. Redistributions  of  source  code  must retain the above copyright
#  notice, this list of conditions and the following disclaimer.
#
#  2. Redistributions in binary form must reproduce  the above copyright
#  notice, this list of conditions,  and the following disclaimer in the
#  documentation and/or other materials provided with the distribution.
#
#  3. All  advertising  materials  mentioning  features  or  use of this
#  software must display the following acknowledgement:
#  This  product  includes  software  developed  at  the  University  of
#  Tennessee, Knoxville, Innovative Computing Laboratories.
#
#  4. The name of the  University,  the name of the  Laboratory,  or the
#  names  of  its  contributors  may  not  be used to endorse or promote
#  products  derived   from   this  software  without  specific  written
#  permission.
#
#  -- Disclaimer:
#
#  THIS  SOFTWARE  IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
#  ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES,  INCLUDING,  BUT NOT
#  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
#  A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE UNIVERSITY
#  OR  CONTRIBUTORS  BE  LIABLE FOR ANY  DIRECT,  INDIRECT,  INCIDENTAL,
#  SPECIAL,  EXEMPLARY,  OR  CONSEQUENTIAL DAMAGES  (INCLUDING,  BUT NOT
#  LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
#  DATA OR PROFITS; OR BUSINESS INTERRUPTION)  HOWEVER CAUSED AND ON ANY
#  THEORY OF LIABILITY, WHETHER IN CONTRACT,  STRICT LIABILITY,  OR TORT
#  (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
#  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
# ######################################################################
#
# ----------------------------------------------------------------------
# - shell --------------------------------------------------------------
# ----------------------------------------------------------------------
#
SHELL        = /bin/sh
#
CD           = cd
CP           = cp
LN_S         = ln -s
MKDIR        = mkdir
RM           = /bin/rm -f
TOUCH        = touch
#
# ----------------------------------------------------------------------
```

```
# - Platform identifier ------------------------------------------------
# ----------------------------------------------------------------------
#
ARCH         = x86_64
#
# ----------------------------------------------------------------------
# - HPL Directory Structure / HPL library ------------------------------
# ----------------------------------------------------------------------
#
TOPdir       = /export/home/ctclark/Desktop/hpl
INCdir       = $(TOPdir)/include
BINdir       = $(TOPdir)/bin/$(ARCH)
LIBdir       = $(TOPdir)/lib/$(ARCH)
#
HPLlib       = $(LIBdir)/libhpl.a
#
# ----------------------------------------------------------------------
# - Message Passing library (MPI) --------------------------------------
# ----------------------------------------------------------------------
# MPinc tells the  C  compiler where to find the Message Passing library
# header files,  MPlib  is defined  to be the name of  the library to be
# used. The variable MPdir is only used for defining MPinc and MPlib.
#
MPdir        = /opt/openmpi
MPinc        = -I$(MPdir)/include
MPlib        = $(MPdir)/lib/libmpi.so
#
# ----------------------------------------------------------------------
# - Linear Algebra library (BLAS or VSIPL) -----------------------------
# ----------------------------------------------------------------------
# LAinc tells the  C  compiler where to find the Linear Algebra  library
# header files,  LAlib  is defined  to be the name of  the library to be
# used. The variable LAdir is only used for defining LAinc and LAlib.
#
LAdir        = /opt/GotoBLAS2
LAinc        =
LAlib        = $(LAdir)/libgoto2.a
#
# ----------------------------------------------------------------------
# - F77 / C interface --------------------------------------------------
# ----------------------------------------------------------------------
# You can skip this section  if and only if  you are not planning to use
# a  BLAS  library featuring a Fortran 77 interface.  Otherwise,  it  is
# necessary  to  fill out the  F2CDEFS  variable  with  the  appropriate
# options.  **One and only one**  option should be chosen in **each** of
# the 3 following categories:
#
# 1) name space (How C calls a Fortran 77 routine)
#
# -DAdd_              : all lower case and a suffixed underscore  (Suns,
#                       Intel, ...),                           [default]
# -DNoChange          : all lower case (IBM RS6000),
# -DUpCase            : all upper case (Cray),
# -DAdd__             : the FORTRAN compiler in use is f2c.
#
# 2) C and Fortran 77 integer mapping
#
# -DF77_INTEGER=int   : Fortran 77 INTEGER is a C int,         [default]
# -DF77_INTEGER=long  : Fortran 77 INTEGER is a C long,
# -DF77_INTEGER=short : Fortran 77 INTEGER is a C short.
#
# 3) Fortran 77 string handling
#
```

```
# -DStringSunStyle    : The string address is passed at the string loca-
#                       tion on the stack, and the string length is then
#                       passed as  an  F77_INTEGER  after  all  explicit
#                       stack arguments,                       [default]
# -DStringStructPtr   : The address  of  a  structure  is  passed  by  a
#                       Fortran 77  string,  and the structure is of the
#                       form: struct {char *cp; F77_INTEGER len;},
# -DStringStructVal   : A structure is passed by value for each  Fortran
#                       77 string,  and  the  structure  is  of the form:
#                       struct {char *cp; F77_INTEGER len;},
# -DStringCrayStyle   : Special option for  Cray  machines,  which  uses
#                       Cray  fcd  (fortran  character  descriptor)  for
#                       interoperation.
#
F2CDEFS       = -DAdd__ -DF77_INTEGER=int -DStringSunStyle
#
# ----------------------------------------------------------------------
# - HPL includes / libraries / specifics -------------------------------
# ----------------------------------------------------------------------
#
HPL_INCLUDES = -I$(INCdir) -I$(INCdir)/$(ARCH) $(LAinc) $(MPinc)
HPL_LIBS     = $(HPLlib) $(LAlib) $(MPlib)
#
# - Compile time options -----------------------------------------------
#
# -DHPL_COPY_L           force the copy of the panel L before bcast;
# -DHPL_CALL_CBLAS       call the cblas interface;
# -DHPL_CALL_VSIPL       call the vsip  library;
# -DHPL_DETAILED_TIMING  enable detailed timers;
#
# By default HPL will:
#    *) not copy L before broadcast,
#    *) call the BLAS Fortran 77 interface,
#    *) not display detailed timing information.
#
HPL_OPTS     =
#
# ----------------------------------------------------------------------
#
HPL_DEFS     = $(F2CDEFS) $(HPL_OPTS) $(HPL_INCLUDES)
#
# ----------------------------------------------------------------------
# - Compilers / linkers - Optimization flags ---------------------------
# ----------------------------------------------------------------------
#
CC           = /usr/bin/gcc
CCNOOPT      = $(HPL_DEFS)
CCFLAGS      = $(HPL_DEFS) -fomit-frame-pointer -O3 -funroll-loops -W -Wall
#
LINKER       = /usr/bin/gfortran
LINKFLAGS    = $(CCFLAGS)
#
ARCHIVER     = ar
ARFLAGS      = r
RANLIB       = echo
#
# ----------------------------------------------------------------------
```