

**Developing a Powerful yet Inexpensive
Computational Infrastructure
for the UT Department of Nuclear Engineering**

David D. Dixon
April 4, 2009

Developing a Powerful yet Inexpensive Computational Infrastructure for the UT Department of Nuclear Engineering

Abstract

This paper presents an approach to assembling and configuring a low cost computer cluster and the software infrastructure necessary to perform parallel calculations of interest to the UT Department of Nuclear Engineering. Emphasis is placed on simplicity, energy efficiency, reliability, and computational speed. Purchasing all-new hardware, the cost per computational node is less than \$125 in 2009, the software used is freely available, and can be expanded or duplicated with the use of this document.

This work was funded by DoE grant DE-PS07-07ID14817, “Improvement in Capabilities for the Nuclear Engineering Department of the University of Tennessee to Support GNEP Research and Development Programs,” with PI Dr. Ronald Pevey, and satisfies the first objective of the grant.

Table of Contents

1 Introduction.....	3
2 Background.....	3
2.1 Existing Cluster from UC.....	4
2.2 Design Considerations.....	5
2.2.1 Power Consumption.....	5
2.2.2 Heat Generation/Removal.....	5
2.2.3 Space.....	6
2.2.4 Cost.....	6
3 Hardware.....	6
3.1 Evaluation & Testing.....	6
3.2 Pricing/Purchasing.....	7
3.2.1 Cabinet.....	8
3.2.2 Cases.....	8
3.2.3 Power Supplies.....	9
3.2.4 Processors and Motherboards.....	9
3.2.4.1 AMD Dual Core CPUs and Motherboards.....	10
3.2.4.2 AMD Quad Core CPUs and Motherboards.....	11
3.2.4.3 Intel Quad Core CPUs and Motherboards.....	12
3.2.4.4 Newer Hardware Available.....	13
3.3 Assembly Guide.....	13
3.3.1 Installation of Components on Motherboard.....	13
3.3.2 Installation of Motherboard in Case.....	13
3.3.3 Additional Components for Head Node.....	14
3.3.4 Connecting Nodes.....	14
3.4 Heat Management and Spacing Between Nodes.....	14
3.5 Water Cooling Experiment.....	14
4 Software.....	16
4.1 Overview.....	16
4.1.1 Alternatives Considered.....	16
4.2 Operating System.....	16
4.2.1 Description of Package Manager.....	16
4.3 Diskless Remote Boot Linux (DRBL).....	16

4.4 Hardware Monitoring.....	17
4.5 Ganglia.....	17
4.6 Webmin.....	20
4.7 MPICH, LAM MPI.....	21
4.8 Application Software.....	21
4.8.1 MCNP/MonteBurns.....	21
4.8.2 VirtualBox.....	22
4.8.3 Matlab.....	22
4.8.4 Other Reactor Analysis Tools.....	22
4.8.5 ANSYS Multi-Physics.....	22
4.8.6 Intel, Portland Group, and GNU Fortran Compilers.....	22
4.8.7 C++, Perl, PHP, MySQL, Java, Apache2, Tomcat.....	22
4.9 Future Opportunities.....	23
4.10 Installation Guide.....	23
5 Appendicies.....	26
5.1 Sensors Shell Script.....	26
5.2 Modified Ganglia Graph PHP Scripts.....	28
5.3 Modules Incorporated into DRBL Deployment Script.....	29

1 Introduction

Nuclear engineers frequently use Monte Carlo methods to solve radiation transport problems. These methods are computationally intensive, but benefit from easy parallelization. The cost to purchase a pre-built cluster is often prohibitive, and can be significantly reduced using commercial off-the-shelf desktop PC components and freely available open source software. The number of components needed is kept to a minimum (e.g. using diskless nodes) which reduces cost and power consumption.

The goal in this project is to develop an inexpensive, efficient replacement for the existing cluster, with a focus on expanding the cluster's size such that it becomes a shared resource for use by students and faculty. This project is in furtherance of a GNEP readiness grant to establish a computational infrastructure for students and staff at the University of Tennessee. Simplicity in the hardware implementation is the first step, complemented by web-based and graphical administration tools. Further, it is a goal of this project to develop a guide for assembling and configuring a cluster that requires minimal understanding of software intricacies, and should minimize the extent to which software is manually configured.

The cluster has already served to execute critical benchmark experiment models in support of a Master's project by Carlos Juarez, as a parallel programming platform for Hermilo Hernandez,¹ for ANSYS Multi-Physics modeling by James Banfield, fuel cycle optimization tasks by Jack Galloway,² and for memory-intensive image processing Matlab calculations (that cannot be run on a 32-bit machine) for Brian Wood, among others. In particular, there was no capability within the department to execute the Matlab tasks or the ANSYS models prior to the installation of this computing cluster. See the application software section (4.8) for more detail of current cluster uses.

2 Background

So-called *Beowulf* computing refers to a NASA cluster bearing that name built in 1994. Its features have been replicated in many forms, but there are a few key features:

- (i) A set of independent computers using commodity hardware connected over a network
- (ii) No custom components
- (iii) Commodity operating system and software
- (iii) A set of tools for allowing parallel processing
- (iv) Appears largely as a single, powerful machine
- (v) Trivially reproducible

A Beowulf cluster typically has a single physical user interface, only one node—the “head” node—is directly accessible from other machines over a network. This head node is also the primary location of disk storage space, though other nodes often have hard drives for storage of boot information and as scratch space.

1 H. Hernandez and G. I. Maldonado, “Added Features and MPI-based Parallelization of the FORMOSA-L Lattice Loading Optimization Code” Accepted, 2009 ANS Advances in Nuclear Fuel Management IV, Hilton Head Island, SC, 2009.

2 J. Galloway, H. Hernandez, and G.I. Maldonado, “OPTIMIZATION OF AMERICIUM-LOADED LATTICES TESTED IN 3D BWR CORE-WIDE SIMULATIONS,” Proceedings of the International Conference on the Physics of Reactors, September 14-19, 2008, Interlaken, Switzerland (CD-ROM, Log#617)

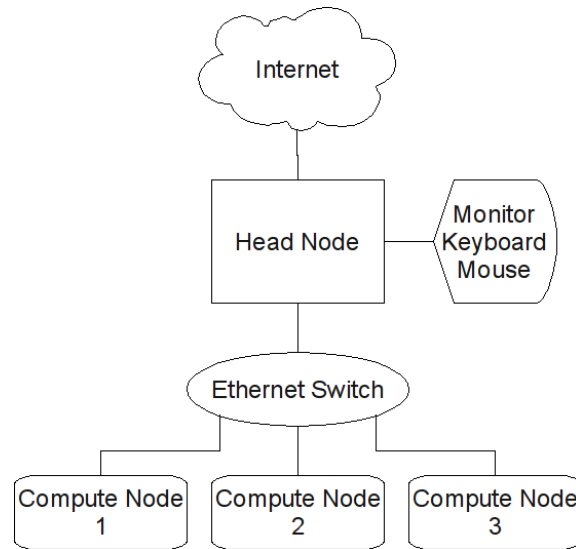


Figure 1: System Layout

2.1 Existing Cluster from the University of Cincinnati (UC)

Prior to the installation of the cluster described in this document, the department's only parallel computing resource was a 10-node cluster brought to UT in late 2007.³

The existing cluster in the department is approximately 6 years old, and has limited usefulness. The compute nodes run at speeds comparable to or slower than many desktop computers available today. Further, it runs outdated software incapable of supporting many of the applications (specifically including security tools) desired. It has only a text-based physical interface, and no web-based management tools.

This cluster also relies on *diskful* nodes. This means that every compute node has its own hard drive. While these nodes boot over the local network, they rely on system images stored locally. This may have been simpler given the available software; however, it is significantly more complex than a diskless system to administer as configuration changes must be distributed to the individual nodes.

The hard drives are also somewhat necessary for this system however, as each node has only 1GB of RAM—insufficient for many common tasks; the hard drives are used for temporary file storage and memory page swapping (extremely slow). Since hard drives rely on moving parts, they are somewhat prone to failure, and the constantly spinning disks consume significant power.

```

model name      : Intel(R) Xeon(TM) CPU 2.40GHz
stepping       : 5
cpu MHz        : 2399.380
cache size     : 512 KB
fpu            : yes
fpu_exception  : yes
cpuid level    : 2
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips      : 4784.12

```

³ Zhao, Z. and G.I. Maldonado, "Speedup of Particle Problems with a Beowulf Cluster," American Journal of Applied Sciences, 3(8): 1948-1951, 2006.

Above is an abbreviated informational listing for one of the CPUs on this cluster. In this listing, key parameters are the CPU model, cache size, and the list of CPU flags. Clock speed is a very poor indicator of performance for comparing different CPU families, as the internal architectures are different. The CPU flags indicate specialized instruction sets that can be used with supporting codes/compilers. With the CPU model, various CPU parameters can be determined.⁴

```
Xeon 2.4 2400 MHz  
512 KiB Cache  
400 MT/s Front Side Bus  
65 W Thermal Design Power  
130 nm Feature Size
```

The cache is small compared to modern CPUs, and the feature size and thermal power are significantly higher than most CPUs currently on the market. The TDP is consistent with power measurements on the existing nodes (2-65W processors, hard drive, fans, and other losses bring the total to 175W). Power and performance are discussed further in the hardware testing and evaluation section.

2.2 Design Considerations

Incorporated into the overall design approach are several constraints and considerations. With less than \$20,000 to spend cost is important; but ultimately its biggest impact is to limit the number of computational nodes that can be purchased. Electricity is effectively the only ongoing operational cost, and without a major investment is available in a fixed quantity—maximizing node efficiency maximizes the number of nodes that can be powered. Likewise, air conditioners limit the rate at which heat can be removed from the room. Ultimately, the smallest, cheapest nodes that consume minimal power with minimal heat would be most desirable. In reality, the most efficient are not the cheapest. The goal is to extract the most performance from a system that fits within these constraints.

2.2.1 Power Consumption

Power consumption is largely driven by only a few key components: the processor itself, cooling system, and the power supply. The power supply is tasked with converting 120V alternating current (AC) power to a predefined set of direct current (DC) voltages, ranging from -12V to +12V. Rectifying and regulating the incoming supply is obviously not lossless. Recently, power supplies have been given some attention, and efficiencies have improved as a result. Active power factor correction is now available in reasonably-priced power supplies, and 80+ certifications indicate power supplies with 80% or greater efficiencies.

The processor is the biggest single power draw in the system, and power largely tracks process feature size (measured in nanometers) and clock speed. Discussed in greater detail in the evaluation section, the suite of processors examined range in power from less than 25W to more than 100W per core. Fortunately, at factory settings, the processor line most efficient happens to be the fastest.

2.2.2 Heat Generation/Removal

Practically speaking, the heat energy added to the room is the same as the electrical power supplied to the system. The room the cluster is located in has limited cooling capacity, provided via one chilled water heat exchanger and one conventional (vapor compression) window air conditioner. The capacity of both units varies with the sink temperature (water or outside air), and room temperature.

⁴ Wikipedia http://en.wikipedia.org/wiki/List_of_Intel_Xeon_microprocessors

The window air conditioner draws extra scrutiny, as its cooling coils have the potential to drop below freezing. When this happens, frost accumulates on the cooling coils, eventually restricting airflow to the point that the unit ceases to remove any heat from the room. By having a single-mode (cooling only) system, there is no defrost cycle available for the coils. The only way to control frost buildup is to ensure that the duty cycle of the compressor on the air conditioner is kept below approximately 75%. While it may seem counterintuitive, raising the thermostat setting can result in a colder room when heat loads are high, as this may be the only way to keep the compressor cycling on and off. When the compressor cycles off with the blower running, warm air from the room is drawn across the frosty coils (not being cooled from the inside during this time) and quickly melt it. Humid summers in the southeast exacerbate this problem.

2.2.3 Space

In the current configuration, space constraints will likely come into play only after power and heat removal limits are exceeded. Given, however, that the cluster is not the sole occupant of the room, opportunities to reduce the physical size of the cluster are considered when they do not adversely affect the system. Using rackmount components, the cabinet installed could support as many as 320 compute nodes (80 boxes). That close-packed configuration is likely not possible without significant design work to get heat out of the cabinet.

2.2.4 Cost

To make available as much computing power as possible, cost per unit must be minimized. Taking into account different CPU options (performance levels) complicates the analysis, and rapidly changing component prices can drive different conclusions at different times. Consider the following example: each box requires a fixed-price set of components (the box itself, power supply, RAM, and cooling fan) totaling \$150. A dual-core CPU and motherboard are available for \$150, and a quad-core set can be had for \$500. In this case, two dual-core systems are \$50 cheaper than a quad-core system with the same level of performance. However, if the price of the quad-core CPU and motherboard drops below \$450, it becomes the more cost effective system. Similar price trends occurred between phases of construction, and are detailed in the sections that follow.

3 Hardware

3.1 Evaluation & Testing

Based on a number of interests, 5 different processors were put through a series of benchmarking tests. With dozens of processors, and hundreds of motherboard and memory options, figuring out where to start can seem overwhelming to a first-time computer builder. Two metrics, performance per dollar and performance per watt, are most important; however, performance is not a clock speed, as clock speeds across different CPU architectures match differing levels of performance. Performance is how fast the computer runs your code. Actually testing systems with a representative problem takes into account memory speeds, front side bus limitations, and cache performance as appropriate. The table below presents a summary of run times, power consumption, and cost (Q1-2009) for a number of processor configurations considered. The benchmark was an MCNP k-code calculation, run in parallel across all four processors using the Message Passing Interface.

Table 1: Summary of Processors Tested

Processor	Run Time	Idle Power (W)	Full Load Power (W)	Energy Cons. (Wh)	CPU Cost
AMD Athlon X2 5200+	18:09	74	145	43.86	\$54.99
AMD Phenom X4 9850	07:12		279	33.46	\$154.99
Intel Core 2 Quad Q6700	06:28	63	127	13.69	\$204.99
Intel Core 2 Quad Q9300	05:24	45	97	8.73	\$239.99
Intel Core 2 Quad Q8200	05:48	45	91	8.8	\$169.99

Another metric of value in comparing processor efficiencies is the total power consumed in running a test case. This value of this metric is approximated by multiplying the full-load power associated with the CPU by the time it took to run the case. Energy consumption is the primary metric by which overclocking is evaluated in this study.

Notice in Table 2 that the Intel processors were successfully run at speeds significantly exceeding their factory rating. Known as overclocking, the processors operating speed can be changed within certain bounds by changing the clock speed and adjusting supply voltages. At one time attempting to overclock a processor was quite risky (hardware damage could easily occur if settings were too aggressive); however, software controls in place today make this much less likely. In most cases, if a system is not able to boot the factory settings can be restored by power cycling the board.

Table 2: Summary of Processors Tested

Processor	Run Time	Idle Power (W)	Full Load Power (W)	Energy Cons. (Wh)
Intel Core 2 Quad Q6700 @ 2.66GHz	06:28	63	127	13.69
@ 3.06GHz	06:05	64	138	13.99
@ 3.15GHz	05:48	63	140	13.53
@ 3.25GHz	05:35	64	143	13.31
Intel Core 2 Quad Q9300 @2.5GHz	05:24	45	97	8.73
@3.00GHz	04:30	47	101	7.58
@3.15GHz	04:17	47	102	7.28
@3.00GHz w/Larger Heat Sink	04:30	47	93	6.98

Notice that idle power is unaffected by changes in clock speed (no speed stepping enabled), and that full-load power is only modestly increased while yielding a significant runtime improvement. In the

last case, the heat sink included with the Q6700 processor was installed on the Q9300. This heat sink had an aluminum fin block roughly 50% taller than the other. The lower fan speeds needed as a result reduced power consumption by 8W.

3.2 Pricing/Purchasing

This section attempts to provide a description of the reasoning that led to the purchases made. Detailed information regarding part numbers and costs are included as an appendix.

3.2.1 Cabinet

The decision to utilize rackmount components carries with it a potentially significant cost. Basic 6'-high rack enclosures approach \$1,000—a fairly significant portion of the available budget. However, the surplus lot on campus turned up a nice, new double-sided rack enclosure with plexiglass doors and an exhaust blower. This cabinet was acquired for \$300.

3.2.2 Cases

The case is simply a physical mounting solution for the computer components for a single node. Within the scope of rackmount cases, the key parameter to evaluate is case height, measured in rack units (1.75" each). Taller cases provide more room for peripherals at the expense of occupying more space in the cabinet. Smaller cases can require careful selection of CPU coolers and RAM to ensure clearance. 2U cases are a good compromise, allowing sufficient room for standard components without wasting too much space. Deciding which case to purchase can be somewhat difficult, as differences between models are often not easily evaluated. However, the difference between a good choice and a bad choice lies primarily in the quality of construction (difficult to determine in pictures), aesthetics, and the ease of installing components into it. In other words, just about any case will function. All compute nodes utilize iStarUSA D-200-PFS⁵ cases, purchased for less than \$100 each (8 at \$86.99, 10 at \$94.99).



Illustration 1: Rackmount Case used for Compute Nodes

⁵ <http://www.newegg.com/Product/Product.aspx?Item=N82E16811165082>



Illustration 2: Rackmount Case used for Head Node

Since the head node would include several hard drives, the cases selected were 4U. From the same line, they are iStarUSA Storm Series D-400,⁶ and included a power supply for \$165.99.

3.2.3 Power Supplies

Selecting a power supply entails looking at three parameters: form factor (the physical size of the power supply), power rating, and efficiency. Efficiencies will not be specified directly, but will be indicated by the presence or absence of the 80+ mark, indicating 80% efficiency or better. Selecting a higher power rating than necessary usually incorporates some additional losses, but brings higher levels of stability. This results from reserve capacity to supply loads on demand, as well as higher wattage cooling systems (i.e., bigger heat sinks and fans inside the power supply).

The power supplies used on all nodes are Antec EarthWatts EA380 power supplies, with active power factor correction (\$34.99 each).

3.2.4 Processors and Motherboards

With an eye toward running large MCNP calculations, two key requirements arose for the CPUs, motherboards, and memory. An inherent limitation of a 32-bit processor is its ability to address only 2^{32} memory locations, or 4GB. Typically, this space has been partitioned equally between the operating system (kernel space) and process space, meaning that an MCNP process would have a maximum of 2GB available. This is easily exceeded in complex models with modern versions of the code. While there special techniques for addressing larger memory blocks, they would require significant code modification.

Availability of 64-bit processors allows addressing up to 16TB of memory (2^{64} address locations), far more than any motherboard can physically accommodate. Only 64-bit processors were considered for this cluster. Further, desktop motherboards are typically equipped with 2 to 4 memory slots. With an interest in running memory intensive applications, and without hard drives to offer swap space, only motherboards with 4 slots were considered. This limits the number of motherboards available, but does not impact cost. For simplicity and administration, any motherboard selected should have built-in video.

⁶ <http://www.newegg.com/Product/Product.aspx?Item=N82E16856999302>

Since this computing cluster was built in stages, price changes and new product introductions have influenced the “best value” product during the course of construction. Initially (January 2008), quad core processor offerings were extremely limited and cost prohibitive. The AMD Phenom was the only credible quad-core option (Intel offerings were too expensive), and the processors available at the time suffered from a transaction lookup buffer (TLB) bug. As a result, the first set of nodes (10) purchased were AMD dual core processors.

When revised AMD Phenom processors (without the TLB bug) were released in March, they became a more cost-effective solution than the dual core units. These processors worked as expected (the cluster includes 5 of these), delivering roughly double the computational power in an identically-sized package (see section titled Water Cooling Experiment for more detail). Perhaps the biggest impact of this processor's release was to put significant price pressure on Intel's Core 2 Quad line of processors, seen as direct competition. The Core 2 Quad Q6700, released in 2007, hit the market with a \$530 price tag, but was available for less than \$300 by the end of Q1-2008. Consideration of these Intel processors would have been likely, however, the next release from Intel would prove more significant. Consistent with Intel's tick-tock product development cycle, a line of Core 2 Quad processors became available utilizing a new, smaller, feature size. These would make up the most recent batch of nodes purchased, as of September 2008. Since then, Intel has released a new micro architecture, and AMD has released a line of Phenom processors with a smaller feature size. All are discussed in the sections that follow.

Within any processor family, there are a number of clock speed ratings to choose from, with significant variations in price. The metric used to determine which clock speed to purchase was GHz per dollar, where the cost in dollars is the total for the node (including motherboard, RAM, case, etc.). This point is significant, as the processor made up only about one fourth of the total node cost.

3.2.4.1 AMD Dual Core CPUs and Motherboards

The first batch of nodes purchased (10) were equipped with AMD Athlon 64 X2 5200+ processors. These processors were rated at 65W (note that the methods for establishing thermal ratings differs between Intel and AMD), and operate 2 cores at 2.7 GHz. This processor is supported by any motherboard with an AM2 socket.

```
Athlon 64 X2 5200+
2700 MHz
2 x 512 KiB L1 Cache
1000 MHz FSB
13.5x Clock Multiplier
1.325/1.375 V
65 W TDP
Socket AM2
Brisbane Core
Released October 8, 2007
65nm Feature Size
```

These processors have performed reasonably well, and—despite a high thermal power among the processors considered—are considerably more energy efficient than the cluster being replaced.



Illustration 3: Motherboard with AMD Athlon 64 X2 5200+ CPU and Cooler Installed

3.2.4.2 AMD Quad Core CPUs and Motherboards

The second batch of nodes purchased were AMD Phenom X4 9850 Black Edition CPUs, running at 2.5 GHz. While “Black Edition” refers to an unlocked clock multiplier to attract overclockers, no appreciable overclocking was achievable with the stock heatsink. This processor is four independent processors on the same core, and with the exception of the clock speed is almost identical to two of the dual core processors.

An additional point of interest with this processor is the incorporation of copper-water heat pipes to transport heat from the processor surface to the aluminum cooling fins. Because heat pipes are ineffective at very low vapor pressures, this heat sink's throughput changes sharply at indicated core temperatures of about 80C. When operated in open air, the core temperatures are significantly higher than any other processor tested.

```
Phenom X4 9850 Black Edition
B3 2.5 GHz
4x 512 KB L1 Cache
2 MB L2 Cache
2 GHz FSB
12.5x Multiplier
1.20 - 1.30 V
125 W TDP
Socket AM2+
Released March 27, 2008
Agena Core
65 nm Feature Size
```

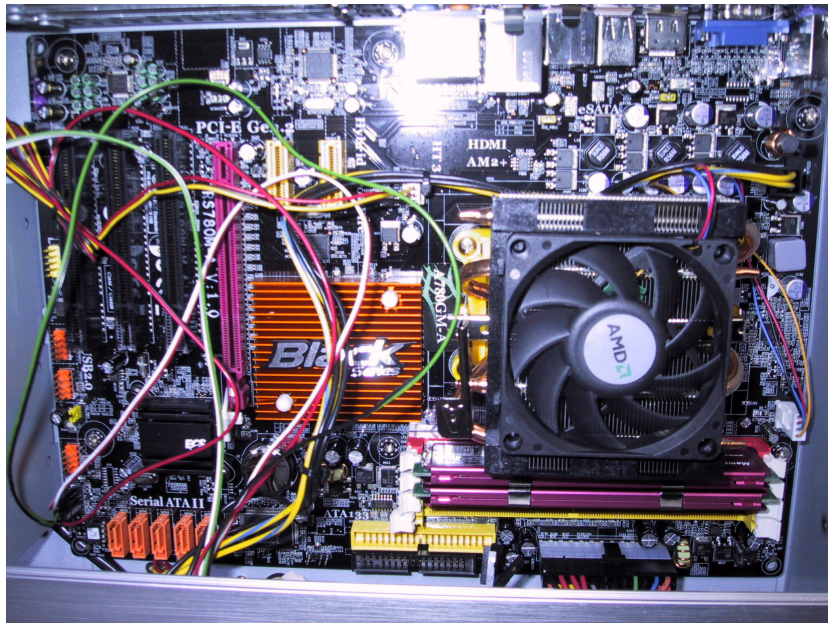


Illustration 4: AMD Phenom X4 9850 Black Processor with OEM heat sink installed in ECS A780GM-A Motherboard

3.2.4.3 Intel Quad Core CPUs and Motherboards

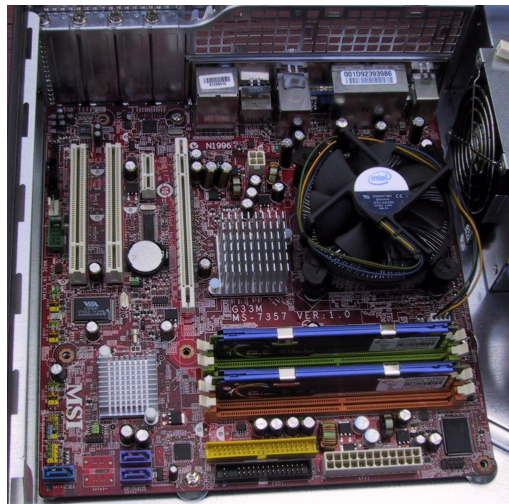


Illustration 5: Intel Core 2 Quad Q8200 CPU installed in MSI G33M motherboard

These processors and motherboards are the most recent addition to the cluster. Four nodes were added in September 2008, using a G33M motherboard produced by MSI. This combination of motherboard and CPU is in use in a 30-node cluster at Los Alamos National Lab, assembled over the summer. Its robustness under severe operating conditions has been tested, running for several weeks in a room whose temperature often exceeded 100°F for hours at a time. All of the operation under these conditions was at speeds from 3.0 to 3.2 GHz. With over 200,000 motherboard hours of operation without a failure, there is very little room to criticize the performance of this combination.

3.2.4.4 Newer Hardware Available

New hardware makes its way to market on a regular basis, and two significant CPU products have been released since the last batch of nodes were purchased for the cluster. Consistent with the tick-tock design cycle at Intel, the newest release is a new microarchitecture—i7—produced using the established 45-nm process. This architecture eliminates the front side bus, and includes hyperthreading on each of the four processor cores. The most significant performance improvement should be evident in multithreaded processes with large data requirements.

AMD's latest Phenom processors, designated Phenom II, now utilize a 45-nm process, and should see energy efficiency improvements of the same scale as with the Intel processors. All other factors equal, this remains a slower processor.

3.3 Assembly Guide

For a first time computer builder, the arrival of perhaps dozens of components might be overwhelming. However, assembly is no different than assembling any other computer—with the exception of the head node, the number of components to be installed is lower than with the average desktop PC; remember that there are no hard drives, no independent video cards, DVD drives, etc. This section describes the basic assembly process.

3.3.1 Installation of Components on Motherboard

NOTE: These components are sensitive to damage from static discharge. Normal handling is usually not a problem; however, the components should not be handled more than necessary, and periodic discharge by touching a grounded surface is advised.

Because space inside many computer cases is limited, it is usually easiest to install the CPU, CPU cooler, and RAM into a loose motherboard. CPUs for many years have made use of zero insertion force (ZIF) sockets that require no pressure to make contact. Instead, pins or contacts on the CPU are clamped in to the mounting socket using a locking lever. The CPU is installed into the motherboard by lifting the locking lever (and a metal clamp in the case of the CPUs shown here), and setting the CPU in place. If it seems as if the CPU is not falling into place, double check that it is oriented correctly—DO NOT use any force. Once in place, lower the clamp and locking lever.

The majority of the heat produced comes from the small area (<1 sq. inch) of the CPU, so it is necessary to provide a heat sink and fan to cool it. As previously discussed, retail boxed CPUs include a fan and heat sink, and are shown below. Set the heat sink and fan assembly on top of the CPU, align the mounting clips and press down on them. This may require considerable force; take care not to flex the motherboard. Then connect the fan power lead to the matching connector on the motherboard.

RAM is installed by first opening the latches (pushing outward) at either end of the slots. Align the memory, noting that the connector is keyed so the RAM can be inserted only one way, and push down. The latches should be pushed into notches at the end of the RAM assemblies. It does not matter on most motherboards which slots are populated.

3.3.2 Installation of Motherboard in Case

Physically mounting the motherboard in the case should be relatively straightforward. Connect two motherboard power cables (24-pin ATX connector, square 4-pin 12V cable), and refer to motherboard documentation for connecting front panel connectors (power switch, reset, and power LED are most important).

3.3.3 Additional Components for Head Node

The head node needs to store the operating system and user files, and will need an extra network interface if it is to be accessible over a network. The simplest means to disk storage is to install a single (or multiple) hard drives directly connected to the motherboard. With single drive capacities exceeding 1.5 TB, and costing less than \$0.07/GB, this is effective for small clusters. Two cables need to be connected to the hard drive, one for power and another for data. For connecting to a network, simply install an additional network card in one of the expansion slots.

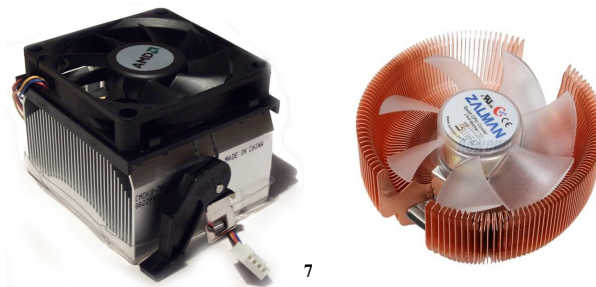
3.3.4 Connecting Nodes

Each node will need to be connected to power, and to the private network switch serving the cluster (refer to figure 1 if necessary). The head node will also need to connect to power; one of its network interfaces will connect to the switch and the other to the public network. Supply power to the switch, and connect monitor, keyboard, and mouse, and the hardware is ready to be powered on.

3.4 Heat Management and Spacing Between Nodes

The first attempt to mount 10 dual-core AMD machines in a rack enclosure resulted in overheating when fully utilized. Automatic shutdown logic built into the BIOS protects the hardware, but it would be necessary to provide additional cooling to these machines. Part of the problem was determined to be the close proximity of the case lid to the CPU cooling fan, which severely restricted flow. Two solutions were immediately viable: leave open rack spaces between nodes, allowing airflow through the case's top vents or replace the OEM CPU coolers with a lower profile heat sink and fan.

Empty rack spaces reduce the total number of nodes that can be accommodated, and new CPU coolers could have cost as much as \$50 per node. A hybrid solution was tested successfully, where a 1U space was left open between every other node, and the nodes without ventilation space received new copper coolers approximately 5 inches in diameter (roughly double the OEM cooler's size). The figures below show the difference between the two coolers.



Notice that the OEM cooler (left) allows airflow in only one direction, while the replacement cooler has its cooling fins oriented radially. Further, the fins on the replacement cooler are copper instead of aluminum, and extend to the full height of the fan.

For \$175 and only 8 3/4" of rack space, the enclosures can exhaust the full power of the dual-core CPUs.

3.5 Water Cooling Experiment

The availability of revised quad-core processors from AMD in March 2008 presented an opportunity to improve the computational density. The higher thermal power however, further complicates the heat

7 http://www.aztec-online.net/images/amd_heatsink.jpg

removal problem, as there is now twice the power per unit volume (there is no appreciable difference in the processors' efficiency).

One fairly common solution for a single machine is to replace conventional heat sinks with water blocks, and cool the system with water. The approach here was to use a finned copper water block, seen below, to mate with individual CPUs, and connect to a pumped water loop rejecting heat through an automotive radiator. Using flexible tubing between components, and a rigid PVC manifold, 5 quad-core processors were mounted in a custom-built 8U rackmount chassis. A test run was performed running all 5 processors (625W) that demonstrated more-than-adequate cooling.



Illustration 6: Water Block for Cooling CPU

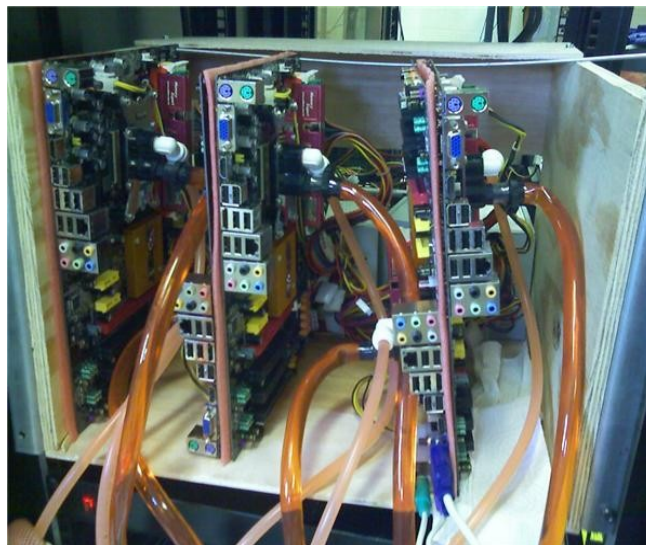


Illustration 7: AMD Quad-Core Motherboards Connected to Water Cooling System

4 Software

4.1 Overview

A number of very useful step-by-step guides are available to aid in the software configuration for a computer cluster. The operating system used here is Ubuntu, a free version of Linux distributed with good hardware support and an easy installation process. Further, its package manager makes installing and configuring most software very easy. The operating system is installed on the head node only. Diskless Remote Boot Linux (DRBL) is then used to support remote booting (i.e. over the network) of the compute nodes. DRBL is installed through Ubuntu's Synaptic Package Manager, and a script walks the user through the configuration process.

4.1.1 Alternatives Considered

Two other clustering packages, often more popular, were considered. The first, ROCKS, is currently in use on the *spacenuke* cluster at Los Alamos National Laboratory. It offers support for diskless nodes, however, it does not support the ethernet hardware built in to many current motherboards. This is a result of the use of Red Hat Enterprise Linux-class kernels, which follow a much slower development pace than others.

The second alternative, OSCAR, would have allowed the use of a newer version of the Linux kernel. This cluster software package was installed and tested, but lacked sufficient support for diskless nodes. The feature technically existed, but did not work correctly and had seen no development in approximately two years.

4.2 Operating System

The operating system selected is Ubuntu's Long Term Support Server Edition. Ubuntu follows a 6-month release cycle, with a long term support version every 12 months. Release numbers match the month and year of release, so the version installed in 2008 on the UT cluster is 8.04, released April, 2008. This operating system is installed through the use of a burned CD image, and supported all of the hardware on the UT cluster out-of-the-box. This, combined with an excellent software package manager and robust server edition made Ubuntu the operating system of choice.

4.2.1 Description of Package Manager

Ubuntu, like the Debian distribution it is based on, uses *dpkg* for installing and uninstalling software at the command line. Ubuntu, however, includes the Synaptic Package Manager for graphical listings of software available. With this interface, Synaptic allows installation, re-installation, or removal of software by simply checking a box next to the listing. Synaptic also shows any differences between the currently installed version of a software package and the newest version available.

4.3 Diskless Remote Boot Linux (DRBL)

Diskless Remote Boot Linux is the package—installed using the Synaptic Package Manager—that allows configuration of boot images for the compute nodes. It also performs a number of other tasks necessary in setting up a cluster:

- Manages dynamic host control protocol (DHCP) server settings

- Configures firewall
- Modifies hosts file to identify new compute nodes
- Collects MAC addresses to match boot settings with physical nodes
- Can be used to configure Clonezilla images

4.4 Hardware Monitoring

With limitations imposed by the facilities available, it is desirable to remotely monitor the “health” of the hardware in operation. Fortunately, it is possible to query thermocouples manufactured as part of the CPUs and motherboards installed. Accessing this sensor data requires loading a kernel chip driver for the specific hardware present, then making a call to *sensors*, which is part of the *lm-sensors* package. Determining which driver to load is aided by *sensors-detect*, which attempts to try each of the available modules and determine which ones are present.

On calling *sensors*, a simple text output is returned with any voltage readings, temperatures, and fan speeds available. As this does not provide for historical data, or monitoring multiple machines at once, its integration with Ganglia (see next section) will be essential.

4.5 Ganglia

Ganglia is a scalable cluster monitoring software package capable of monitoring a number of predefined metrics (e.g. CPU load, memory usage) as well as custom metrics (e.g. temperatures, fan speeds). Each node runs a monitoring daemon, which broadcasts data to all other nodes on the network's subnet. When the meta-daemon running on the head node queries the cluster for data, only one node need respond with a message containing data for all of the nodes. This is an attempt to provide for scaling to extremely large computing clusters.

The data collected by the meta-daemon is stored in a round-robin database. In such a database, when new data is received it flushes out the oldest data to make room. In other words, after an initial period of time where the database filled, it will not grow without bound. This data is readily queried by a web application, running PHP scripts, that provides data as shown in the figures that follow.

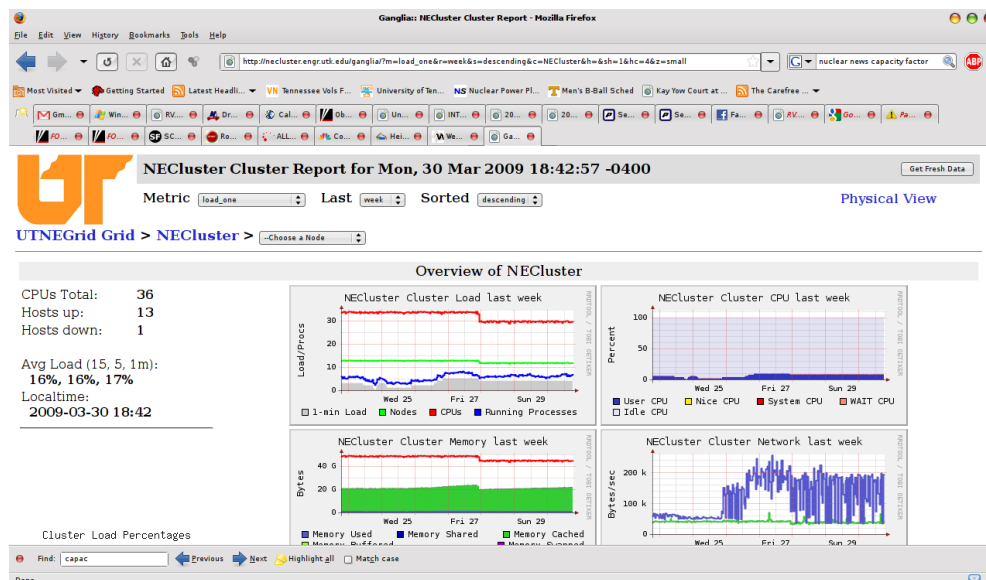


Illustration 8: Main Ganglia Cluster Status Page

One of the custom tools added to the Ganglia status page is a system-average temperature plot. This has proven useful in diagnosing room air conditioning problems, as a temperature rise appears without a matching increase in system load. The modifications included making additional queries to the round-robin database, and performing mathematical sums/averages within the scope of a web page (i.e. no external code is used, all graphs are generated on the user's request). The figure below shows a sample plot of week-long average temperature history.

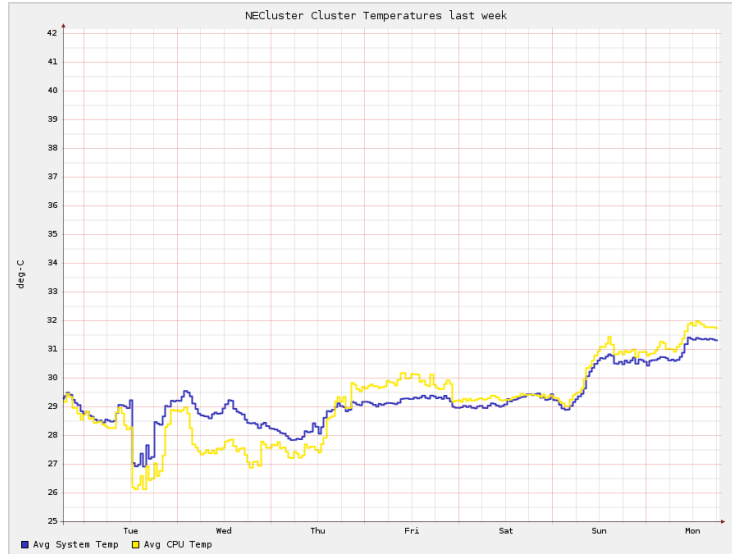


Illustration 9: Custom System-Average Temperature Monitoring Plot

Generally speaking, Ganglia is a very valuable tool for monitoring cluster utilization and health. Its most important purpose, given that the cluster does not make use of a job scheduler, is to allow users to quickly determine which nodes are available for job submission. Without the historical information, short drops in CPU usage could lead a user to believe that a machine is available. Such drops are very common running MCNP in parallel k-code, as a data collection rendezvous occurs after each cycle.

The default and custom metrics already described are supplemented by an additional script that submits metric data for each active user, showing that user's memory and CPU usage on each node. This allows tracing of resource utilization problems, and in the event of a crash documents who was active at the time it occurred. This script is called by the hardware monitoring script, documented in section 5.1.

Node detail pages are shown in the next three figures, showing the node status, utilization by user, and hardware health.

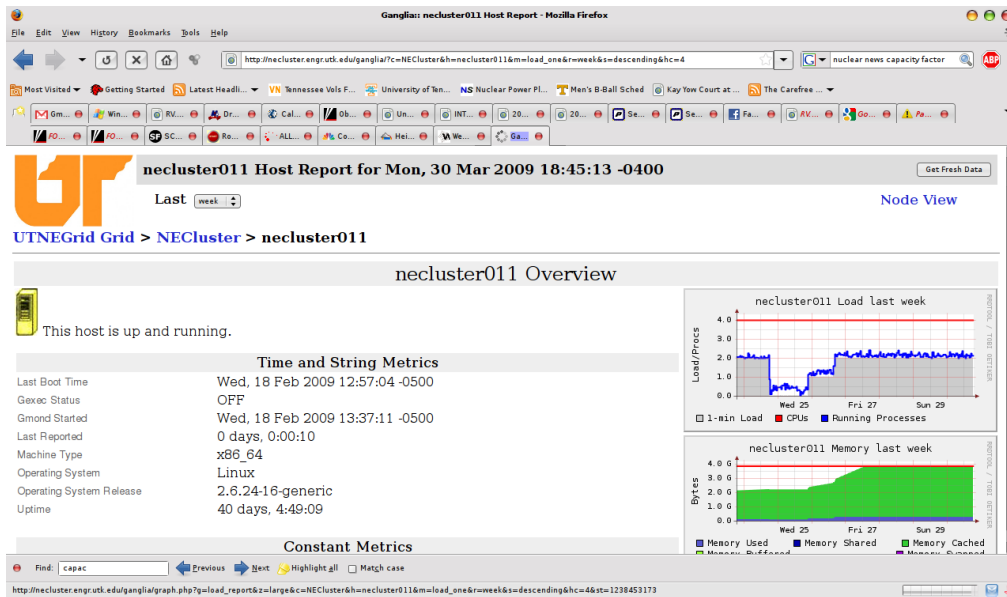


Illustration 10: Ganglia Node Detail Page



Illustration 11: Ganglia Node Detail Page - User CPU Utilization

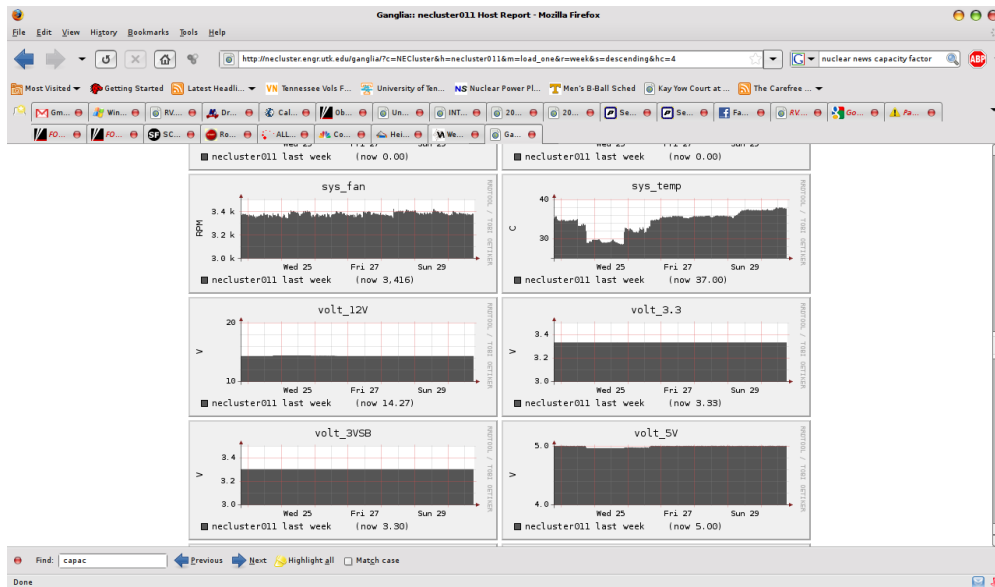


Illustration 12: Ganglia Node Detail Page - Hardware Health (System Voltages, Temperatures, Fan Speed)

4.6 Webmin

Many system administration tasks are typically not very user friendly in Linux environments. Most are accomplished through the use of command line activity and configuration file manipulation. One tool, installed on the UT cluster, that greatly simplifies administration is Webmin. Webmin is a web-based system administration tool that provides graphical, point-and-click access to many common tools while preserving the ability to access configuration files directly through the interface. Additionally, changes to configuration files outside of Webmin will be read back into Webmin's graphical interface automatically. The following two figures show the main Webmin status page, and the user account administration page.

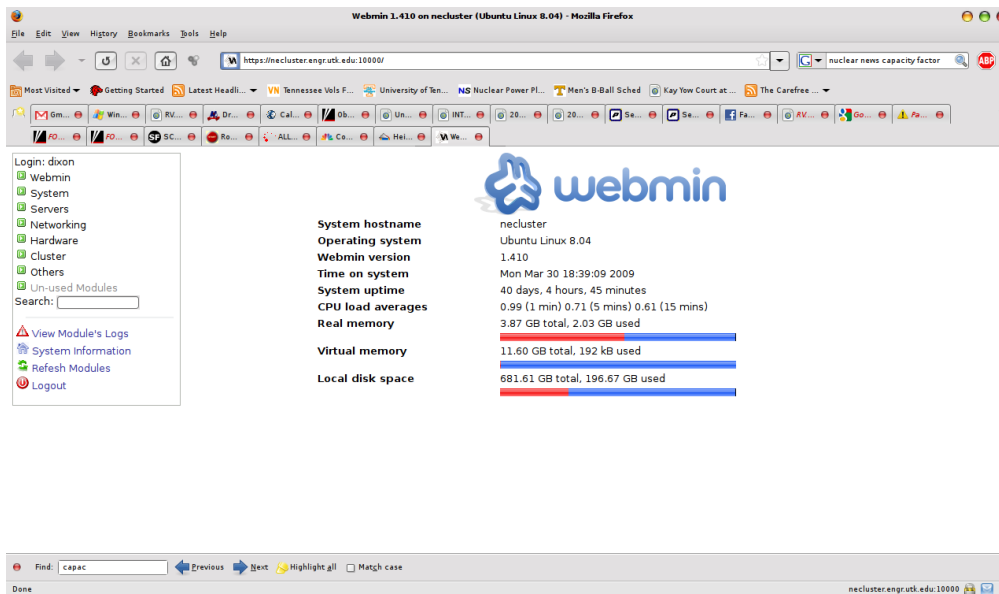


Illustration 13: Webmin System Status Page

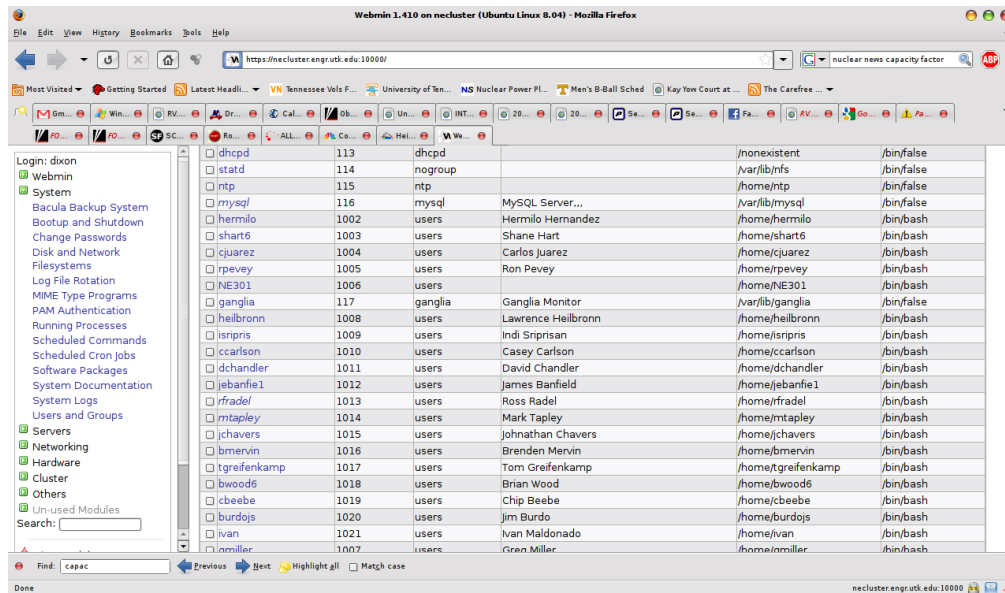


Illustration 14: Webmin User Accounts List

4.7 MPICH, LAM MPI

The Message Passing Interface (MPI) is probably one of the most popular protocols for exchanging process data across multiple machines for code execution in parallel. Two implementations of MPI—MPICH and LAM MPI—are currently installed on the UT cluster. MPICH is most commonly used by MCNP users, while LAM MPI is the default for some of the code written by Hermilo Hernandez.

The presence of MPI is perhaps the most important threshold for enabling parallel code execution on the UT cluster. The MPI implementations are configured to be enabled at boot for the head node and all of the compute nodes over the Ethernet links between them.

4.8 Application Software

This section provides basic descriptions of software made available to nuclear engineering students and faculty with the installation of this computer cluster.

4.8.1 MCNP/MonteBurns

A number of MCNP executables are available in the /opt/exe directory on the cluster. These include a number of versions of MCNP4C, MCNP5, and MCNPX, with a number of compilation options (e.g. with and without plotting capability). One option not available elsewhere is a parallel-capable version of MCNPX that reduces the amount of memory allocated to the amount necessary to support neutron, photon, and electron calculations. This results in the ability to perform even larger calculations (at least in terms of geometry), and executes approximately 10% faster than the standard version in most cases.

Monte Carlo depletion tools are also available in the form of MonteBurns, which is an external coupling of MCNP (X or 5) and ORIGEN, and MCNPX 2.6, which couples MCNPX and CINDER internally.

4.8.2 VirtualBox

VirtualBox is a virtual machine software platform that allows users to create their own virtual machines for tasks where specific operating system support is necessary or where experimentation with operating system parameters is necessary without wanting to disturb the existing cluster infrastructure. Uses of VirtualBox would include executing Windows-only software, Linux software that requires a distribution other than Debian/Ubuntu, to include ANSYS Multi-Physics. VirtualBox is freely available software supported by Sun Microsystems.

4.8.3 Matlab

Matlab has been installed using a University-provided license supporting 64-bit execution with up to 8GB of memory (and more with the use of swap space—this is not limited by addressing capability). Brian Wood is currently making use of this capability for processing CT images. His use of the cluster began when it became impossible to load certain sets of data on a 32-bit machine, and has realized the significantly increased execution speed provided by the cluster.

4.8.4 Other Reactor Analysis Tools

The CoreMaster 2 suite of reactor analysis tools developed by Westinghouse are available on the cluster and are currently supporting advanced fuel cycle development activities as part of a PhD for Jack Galloway. This package includes PHOENIX and POLCA among others.

Also available are versions of NESTLE and SCALE, previously unavailable within the department on a high-performance computer.

4.8.5 ANSYS Multi-Physics

A license for ANSYS Multi-Physics for use by James Banfield in support of HTGR fuel block stress analysis funded by Los Alamos National Laboratory at the end of 2008. This license is renewable annually, and is available to others within the department while the license is active.

4.8.6 Intel, Portland Group, and GNU Fortran Compilers

Licenses for the Intel and Portland Group optimizing FORTRAN compilers have been purchased for this cluster. These compilers, combined with the MPI packages, provide the ability to develop parallel computer codes. Hermilo Hernandez has been making use of this capability.

Also available is the freeware GNU compiler suite, which includes FORTRAN 77 and 95 compilers. This is not, however, an optimizing compiler.

4.8.7 C++, Perl, PHP, MySQL, Java, Apache2, Tomcat

Other code and web development tools available on the cluster include GNU C++, Perl and PHP for scripting, the MySQL database platform, Java Virtual Machine and the Java Software Development Kit, and the Apache2 web server with Java servlets enabled through Tomcat. These tools are well documented elsewhere, and are listed here only to indicate their presence on the cluster. With the exception of C++, Perl, and PHP, these tools were not available on the old UC cluster.

4.9 Future Opportunities

- More nodes

Processors are continually improving, and are increasingly more affordable. The platform is almost infinitely expandable, with options to allow separate subnets to increase network bandwidth.

- Smaller form factors

The energy efficiency improvements reported in this document have resulted in motherboard designs based on the mini-ITX form factor that support the fastest quad-core processors available today. Previously, this form factor (about 6" square, compared to the 9" square motherboards currently in use) was reserved for very low power (thermal and computational ability) applications because of the difficulty in managing the heat generated. Consequently, there are a few case designs in the works that will allow installation of two or more motherboards in a single 1U rackmount case.

- Development of a Job Scheduler and Automated Power Management system

Currently available job schedulers are not really designed for the types of computing tasks typical of nuclear engineers. Most are somewhat difficult to set up, and less than user friendly. Development of a simple scheduler, with an eye towards MPI applications, could be combined with a system that powers nodes on and off as they are needed. Note that the capability of powering nodes on and off remotely is already available, but control of that functionality is reserved for system administrators.

4.10 Installation Guide

This installation guide is intended to be as brief as possible, while still providing a step-by-step process for configuring the necessary cluster software. More detail about individual packages can be found through online sources.

```
Ubuntu-Server install
Make sure to select LAMP server and OpenSSH server
Recommend use of LVM
```

```
Uncomment community entries in /etc/apt/sources.list, add the following
line at the end of the file:
```

```
deb http://free.nchc.org.tw/drbl-core drbl stable testing
```

Alternate for above step: Under the system menu, go to Administration, then Software Sources. Make sure all checkboxes are checked under Ubuntu Software, then click Add under Third Party Software and add the link above.

```
Add key for DRBL: (note the need to run apt-key as su is omitted from
drbl installation guide)
```

```
wget http://drbl.nchc.org.tw/GPG-KEY-DRBL; sudo apt-key add GPG-KEY-DRBL
```

```
sudo apt-get update
sudo apt-get upgrade
```

```
sudo apt-get install ubuntu-desktop build-essential librrd2-dev libapr1-
dev libconfuse-dev libexpat1-dev python-dev rrdtool apache2 php5-mysql
libapache2-mod-php5 localepurge vim lm-sensors drbl libx11 libx11-dev
```

Alternate: Select these packages using the Synaptic package manager found under System-->Administration.

In the localepurge dialog, select only the three formats containing en-US

Set root password: `sudo passwd root`

Reboot (type `sudo reboot`)

Optionally, edit `/etc/vim/vimrc` and remove the `"` in front of syntax on

Login using the username and password created during the ubuntu setup process.

Left-click network icon in upper right corner. Click unlock, enter password. For network interface(s) that will provide service to compute nodes, edit properties as follows:

Disable roaming mode

Configuration=Static IP address

IP address= 192.168.0.254

Subnet mask=255.255.255.0

Gateway address=blank

If using more than 1 interface, increment third octet (number) in IP address. Recommend matching 3rd octet with interface number (e.g. eth0 --> 192.168.0.254)

Go to <http://drbl.sourceforge.net/one4all/>

Open a terminal window

```
sudo /opt/drbl/sbin/drblsrv -i
```

Do you want to install those network installation boot images NO

Do you want to use the serial console output for clients NO

Do you want to upgrade operating system NO

While you're waiting for the previous command to finish, run `sensors-detect` to determine which modules it looks like it needs to be added to get temperature, fan speed, and voltage data.

Modify `/opt/drbl/sbin/drblpush`: search for "add any modules" to get to the section where you can add chip drivers to be loaded by the compute nodes, and add the modules recommended by `sensors-detect` before the END MODULE line.

For CPU frequency scaling, add
`powernow_k8` or `acpi`, depending on CPU type
`cpufreq_ondemand`
`freq_table`

Change `/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor` to `ondemand`

Install Ganglia

Download the `ganglia-monitor-core` package from <http://ganglia.info>

Follow instructions at <http://www.zimbio.com/pilot?ID=xEp3SD99KI4&ZURL=%2FUbuntu%2BLinux%2Fnews%2FxEp3SD99KI4%2FInstalling%2Bganglia%2B3%2B1%2B1%2BUbuntu%2BLinux%2B8%2B04&URL=http%3A%2F%2Fdx13.co.uk>

<http://www.farticles.com/2008/09/12/Installing-ganglia-311-on-Ubuntu-Linux-804-Hardy-Heron.html>

or http://docs.google.com/View?docid=dgmmft5s_45hr7hmggr

Follow instructions for server running `gmond` and `gmetad`

Note that in the command to copy the web files both the copy and move must be run as `su`

Copy `/etc/init.d/ganglia-monitor` and `/etc/init.d/gmetad` scripts from `necluster` OR install `ganglia` packages using `synaptic` *before* installing from source

Copy /opt/drbl/conf/client-extra-service.example to
/opt/drbl/conf/client-extra-service, and add ganglia-monitor and cron to
the list.

Copy sensors.sh to /usr/bin (or include its full path in crontab)

Edit /etc/crontab to include the following line:

```
* * * * * root sensors.sh
```

MCNP

To compile MCNP with plotting, you may need 32-bit versions of X11
libraries. The easiest way to do this is with the getlibs package,
available here: <http://ubuntuforums.org/showthread.php?t=474790>

Install it, then run `sudo getlibs -l libX11.a`

On the plot menu, set the following:

- | | |
|----------------------------------|--------------|
| 3. Graphics Library Path (entry) | /usr/lib |
| 4. Graphics Library Name (entry) | libX11.a |
| 5. Graphics Include Path (entry) | /usr/include |

5 Appendicies

5.1 Sensors Shell Script

```
#!/bin/bash

# /etc/sysconfig/ganglia is used to specify INTERFACE
CONFIG=/etc/sysconfig/ganglia
[ -f $CONFIG ] && . $CONFIG

#default to eth0
if [ -z "$MCAST_IF" ]; then
    MCAST_IF=eth0
fi

GMETRIC_BIN=/usr/bin/gmetric
# establish a base commandline
GMETRIC="$GMETRIC_BIN"

# Grab a random value between 0-30.
value=$RANDOM
while [ $value -gt 30 ] ; do
    value=$RANDOM
done

# Sleep for that time.
sleep $value

SENSORS=sensors
if [ `${SENSORS} | head -n 1` == f71882fg-isa-0a10 ]; then
# send cpu temps if gmond is running
#for i in `seq 1 1000000`;
#do
    # send cpu temperatures
    for temp in `${SENSORS} | grep "hyst = +84" | cut -b 14-18`; do
        gmetric -tfloat -ncpu_temp -uC -v$temp -d360
        echo cpu_temp $temp
    done
    for temp in `${SENSORS} | grep "hyst = +81" | cut -b 14-18`; do
        gmetric -tfloat -nsys_temp -uC -v$temp -d360
        echo sys_temp $temp
    done
    for temp in `${SENSORS} | grep "System" | grep "RPM" | cut -b 13-17`; do
        gmetric -tfloat -nsys_fan -uRPM -v$temp -d360
        echo sys_fan $temp
    done
    for temp in `${SENSORS} | grep "CPU" | grep "RPM" | cut -b 13-17`;
do
        gmetric -tfloat -ncpu_fan -uRPM -v$temp -d360
        echo cpu_fan $temp
    done
    /usr/bin/svoltages.sh
#    sleep 60
#done
elif [ `${SENSORS} | head -n 1` == k8temp-pci-00c3 ]; then
    for temp in `${SENSORS} | grep "temp2" | cut -b 14-18`; do
        gmetric -tfloat -ncpu_temp -uC -v$temp -d360
        echo cpu_temp $temp
    done
    for temp in `${SENSORS} | grep "temp1" | cut -b 14-18`; do
        gmetric -tfloat -nsys_temp -uC -v$temp -d360
        echo sys_temp $temp
    done
fi
```

```

done
for temp in ` ${SENSORS} | grep "fan2" | grep "RPM" | cut -b 13-17`;
do
    gmetric -tfloat -nsys_fan -uRPM -v$temp -d360
    echo sys_fan $temp
done
for temp in ` ${SENSORS} | grep "fan1" | grep "RPM" | cut -b 13-17`;
do
    gmetric -tfloat -ncpu_fan -uRPM -v$temp -d360
    echo cpu_fan $temp
done
# /home/dixon/svoltages_AMD.sh
elif [ ` ${SENSORS} | head -n 1` == it8716-isa-0e80 ]; then
for temp in ` ${SENSORS} | grep "temp2" | cut -b 14-18`; do
    gmetric -tfloat -ncpu_temp -uC -v$temp -d360
    echo cpu_temp $temp
done
for temp in ` ${SENSORS} | grep "temp1" | cut -b 14-18`; do
    gmetric -tfloat -nsys_temp -uC -v$temp -d360
    echo sys_temp $temp
done
for temp in ` ${SENSORS} | grep "fan2" | grep "RPM" | cut -b 13-17`;
do
    gmetric -tfloat -nsys_fan -uRPM -v$temp -d360
    echo sys_fan $temp
done
for temp in ` ${SENSORS} | grep "fan1" | grep "RPM" | cut -b 13-17`;
do
    gmetric -tfloat -ncpu_fan -uRPM -v$temp -d360
    echo cpu_fan $temp
done
# /home/dixon/svoltages_AMD.sh
else
    echo Sensors Output Not Regognized by sensors.sh
fi
/usr/bin/user_stats.pl

```

svoltages.sh

```
#!/bin/bash
```

```
CONFIG=/etc/sysconfig/ganglia
[ -f $CONFIG ] && . $CONFIG
```

```
#default to eth0
if [ -z "$MCAST_IF" ]; then
    MCAST_IF=eth0
fi
```

```
GMETRIC_BIN=/usr/bin/gmetric
# establish a base commandline
GMETRIC="$GMETRIC_BIN"
```

```
SENSORS=sensors
```

```

# send voltages
for temp in ` ${SENSORS} | grep "3.3V" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_3.3 -uV -v$temp -d120
    echo `hostname` volt_3.3 $temp
done
for temp in ` ${SENSORS} | grep "Vcore" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_Vcore -uV -v$temp -d120
    echo `hostname` volt_Vcore $temp
done
for temp in ` ${SENSORS} | grep "Vdimm" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_Vdimm -uV -v$temp -d120
    echo `hostname` volt_Vdimm $temp

```

```

done
for temp in `${SENSORS} | grep "Vchip" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_Vchip -uV -v$temp -d120
    echo `hostname` volt_Vchip $temp
done
for temp in `${SENSORS} | grep "+5V" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_5V -uV -v$temp -d120
    echo `hostname` volt_+5V $temp
done
for temp in `${SENSORS} | grep "12V" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_12V -uV -v$temp -d120
    echo `hostname` volt_+12V $temp
done
for temp in `${SENSORS} | grep "5VSB" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_5VSB -uV -v$temp -d120
    echo `hostname` volt_5VSB $temp
done
for temp in `${SENSORS} | grep "3VSB" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_3VSB -uV -v$temp -d120
    echo `hostname` volt_3VSB $temp
done
for temp in `${SENSORS} | grep "Battery" | cut -b 13-18`; do
    gmetric -tfloat -nvolt_Battery -uV -v$temp -d120
    echo `hostname` volt_Battery $temp
done

```

5.2 Modified Ganglia Graph PHP Scripts

This Ganglia graph script adds generation of the cluster-average temperature plot that appears as part of the header on the main Ganglia page.

```

<?php

/* Pass in by reference! */
function graph_cputemp_report ( &$rrdtool_graph ) {

    global $context,
           $cpu_idle_color,
           $cpu_nice_color,
           $cpu_system_color,
           $cpu_user_color,
           $cpu_wio_color,
           $hostname,
           $range,
           $rrd_dir,
           $size;

    $rrdtool_graph['height'] += $size == 'medium' ? 14 : 0 ;
    $title = 'Temperatures';
    if ($context != 'host') {
        $rrdtool_graph['title'] = $title;
    } else {
        $rrdtool_graph['title'] = "$hostname $title last $range";
    }

    # $rrdtool_graph['upper-limit'] = '100';
    # $rrdtool_graph['lower-limit'] = '0';
    $rrdtool_graph['vertical-label'] = 'deg-C';
    # $rrdtool_graph['extras'] = '--rigid';

    if($context != "host" ) {

```

```

        /* If we are not in a host context, then we need to calculate
the average */
        $series =
            "DEF:'num_nodes'='$
{rrd_dir}/cpu_user.rrd':'sum':AVERAGE "
            ."DEF:'sys_temp'='$
{rrd_dir}/sys_temp.rrd':'sum':AVERAGE "
            ."CDEF:'csys_temp'=sys_temp,num_nodes,/ "
            ."DEF:'cpu_temp'='$
{rrd_dir}/cpu_temp.rrd':'sum':AVERAGE "
            ."CDEF:'ccpu_temp'=cpu_temp,num_nodes,/ "
            #
            ."DEF:'cpu_maxtem'='$
{rrd_dir}/cpu_temp.rrd':'sum':AVERAGE "
            #
            ."CDEF:'ccpu_maxtem'=cpu_maxtem,num_nodes,/ "
            #
            ."DEF:'cpu_idle'='$
{rrd_dir}/cpu_idle.rrd':'sum':AVERAGE "
            #
            ."CDEF:'ccpu_idle'=cpu_idle,num_nodes,/ "
            ."LINE2:'csys_temp'#$cpu_user_color:'Avg System Temp'
"
            #
            ."LINE2:'ccpu_temp'#$cpu_nice_color:'Avg CPU Temp' ";
            #
            ."LINE2:'ccpu_maxtem'#$cpu_system_color:'Max CPU
Temp' ";
            #
            $series .= "STACK:'ccpu_idle'#$cpu_idle_color:'Idle
CPU' ";

    }
    // Context is not "host"
    else {

        $series ="DEF:'cpu_user'='${rrd_dir}/cpu_user.rrd':'sum':AVERAGE
"
        #
        ."DEF:'cpu_nice'='${rrd_dir}/cpu_nice.rrd':'sum':AVERAGE "
        ."DEF:'cpu_system'='${rrd_dir}/cpu_system.rrd':'sum':AVERAGE "
        ."DEF:'cpu_idle'='${rrd_dir}/cpu_idle.rrd':'sum':AVERAGE "
        ."AREA:'cpu_user'#$cpu_user_color:'User CPU' "
        ."STACK:'cpu_nice'#$cpu_nice_color:'Nice CPU' "
        ."STACK:'cpu_system'#$cpu_system_color:'System CPU' ";

        if (file_exists("${rrd_dir}/cpu_wio.rrd")) {
            $series .= "DEF:'cpu_wio'='$
{rrd_dir}/cpu_wio.rrd':'sum':AVERAGE ";
            $series .= "STACK:'cpu_wio'#$cpu_wio_color:'WAIT CPU' ";
        }

        $series .= "STACK:'cpu_idle'#$cpu_idle_color:'Idle CPU' ";
    }

    $rrdtool_graph['series'] = $series;

    return $rrdtool_graph;
}

?>

```

5.3 Modules Incorporated into DRBL Deployment Script

The DRBL deployment script, `drblpush`, contains a section to add a list of kernel modules to be loaded by compute nodes. A list is provided below of the modules currently requested. The first three provide access to hardware monitoring sensors for the different motherboards in use, while the remainder provide the CPU frequency scaling that significantly reduces the idle power consumption of the cluster.

```

k8temp
it87

```

f71882fg
powernow_k8
acpi
cpufreq_ondemand
freq_table