Design and Analysis of the Alliance / University of New Mexico Roadrunner Linux SMP SuperCluster

David A. Bader [*] dbader@eece.unm.edu
Arthur B. Maccabe [†] maccabe@cs.unm.edu
Jason R. Mastaler [‡] jason@ahpcc.unm.edu
John K. McIver III jkmciver@ahpcc.unm.edu
Patricia A. Kovatch pkovatch@ahpcc.unm.edu

Albuquerque High Performance Computing Center
The University of New Mexico
Albuquerque, NM 87131
USA

Abstract

This paper will discuss high performance clustering from a series of critical topics: architectural design, system software infrastructure, and programming environment. This will be accomplished through an overview of a large scale, high performance SuperCluster (named Roadrunner) in production at The University of New Mexico (UNM) Albuquerque High Performance Computing Center (AHPCC). This SuperCluster, sponsored by the U.S. National Science Foundation (NSF) and the National Computational Science Alliance (NCSA), is based almost entirely on freely-available, vendor-independent software. For example, its operating system (Linux), job scheduler (PBS), compilers (GNU/EGCS), and parallel programming libraries (MPI). The Globus toolkit, also available for this platform, allows high performance distributed computing applications to use geographically distributed resources such as this SuperCluster. In addition to describing the design and analysis of the Roadrunner SuperCluster, we provide experimental analyses from grand challenge applications and future directions for SuperClusters.

Please see `www.alliance.unm.edu` *for further information.*

Keywords: Cluster Computing, Message Passing Interface (MPI), Symmetric Multiprocessors (SMP), Communication Primitives, Experimental Parallel Algorithmics.

1. Introduction

Over the next decade, clusters will span the entire range of high-performance computing platforms. Instead of being restricted solely to supercomputers for achieving high-performance computing, the spectrum of available platforms will spread out to Beowulf-class clusters, to SuperClusters, and on to supercomputers. Moving along this continuum represents increasing computational resources, which in turn allows the capability for solving larger classes of applications or instances of problems. Compute-bound applications on workstations often may take advantage of the extra horsepower provided by symmetric multiprocessors. Applications with even larger computational requirements and a high-degree of concurrency may use Beowulf clusters, that is, clusters that are home-built from mass-market, commercial off-the-shelf workstations and networks, for instance, PC's interconnected by Fast Ethernet. A SuperCluster may contain commodity SMP nodes interconnected with scalable, low-latency, high-bandwidth networks, and is integrated by a system vendor rather than the computational scientist. Applications with greater complexity, for example, grand challenges from the sciences, demand scalable systems with high-performance, interconnects, perhaps coupled with terascale I/O subsystems, and thus, are appropriate for these superclusters. Finally, supercomputers, the largest of the clusters, may provide the ultimate platform for the highest-fidelity models and simulations and data-intensive applications. At each step up this ladder of cluster architectures, the system cost increases ap-

[*] Also affiliated with the Electrical & Computer Engineering Department and supported in part by Department of Energy Sandia National Laboratories contract number AX-3006.

[†] Also affiliated with the Computer Science Department. This research was partially supported by the NSF CISE Research Infrastructure award CDA-9503064 and by Department of Energy Sandia National Laboratories contract number AP-1739.

[‡] Corresponding author.

proximately by an order of magnitude, but having the high-performance capability to solve the problem of interest may require an entry point at any of these levels, the Beowulf, or SuperCluster, or even supercomputer platform.

Cluster computing with Beowulf- or SuperClusters may be one or two orders of magnitude less expensive than a supercomputer yet still provide a cost-effective solution and capability that was not available on a workstation. But this cost comparison measures just the basic system price. What about the applications? As recently as a few years ago, the lack of a canonical high-performance architectural paradigm meant that migrating to a new computer system, even from the same vendor, typically required redesigning applications to efficiently use the high-performance system. Thus, the application must be redesigned with parallel algorithms and libraries that are optimized for each high-performance system. This process of porting, redesigning, optimizing, debugging, and analyzing, the application may in fact be prohibitively expensive, and certainly frustrating when the process is completed just in time to greet the next generation system! With the successful standardization of system-level interfaces, many of these issues no longer hinder our expedited use of these new, high-performance clusters.

The U.S. National Science Foundation (NSF) currently supports two large advanced computing partnerships, National Computational Science Alliance (NCSA) based at The University of Illinois, Urbana-Champaign, and the National Partnership for Advanced Computing Infrastructure (NPACI) at The University of California, San Diego. Traditionally these centers have purchased large, expensive parallel machines such as the SGI Origin and IBM SP-2. Researchers that require large computational resources can request time on these machines.

The dilemma is that these machines are so heavily utilized that jobs will often sit in the queue for weeks before they are able to run, and quite understandably, many researchers find this delay unacceptable. It is also the case that many applications do not require the complex architectural features available on these large, expensive, parallel machines. For example, the SGI Origin has the ability to provide non-uniform shared memory, but this feature may not be required, or worse, adversely affect the performance of codes that have not been fully tuned. In addition, several classes of applications tend to be computationally demanding with minimal communication. Examples include Monte Carlo simulations, computational fluid dynamics problems, and lattice computation problems which are used in computational physics.

To address these needs, scientists at The University of New Mexico and NCSA have established the first NSF-funded Linux SuperCluster at UNM using PC-class (Intel-based) hardware, a high-performance interconnection network, and the Linux operating system. This production cluster costs a fraction of the price of a commercial supercomputer, and performs well for many classes of applications.

1.1. What is a SuperCluster?

Currently there are many commodity clusters being built using mass-market commercial “off the shelf” (COTS) parts. Typically this involves purchasing or ordering the necessary components such as PC’s with Fast Ethernet NICs, and a hub or a switch to connect them together. Then comes the installation of the operating system as well as the necessary parallel processing software packages. These types of clusters are often classified as “Beowulf Systems” [3]. Our SuperCluster design on the other hand follows a different principle and does not require computational scientists also to be expert computer hobbyists. SuperClusters, while PC-based, are integrated (hardware and software) by a vendor (in our case, Alta Technologies) and may have additional supercomputing-like hardware, such as a high-performance networks and hardware monitors, that improves scalability. In this manner, a SuperCluster may support the scalable execution of a greater number of applications than a Beowulf-class machine. The Roadrunner SuperCluster arrived as a unit, and came pre-installed with Linux, a programming environment, and custom disk cloning software that allows a system administrator simply to configure a single node, and then clone it to the rest of the machine. This approach means the system as a whole is easier to get running initially and maintain into the future.

Figure 1. The Alliance/UNM Roadrunner SuperCluster.

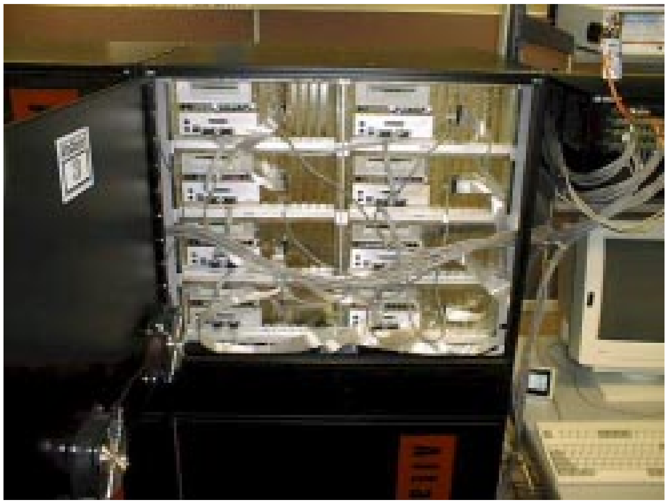

Figure 2. A Peek Inside the Roadrunner.

The UNM/Alliance Roadrunner SuperCluster (shown in Figures 1 and 2) is an Alta Technology Corporation 64-node AltaCluster containing 128 Intel 450 MHz Pentium II processors. The SuperCluster runs the 2.2.10 Linux operating system in SMP mode with communication between nodes provided via a high-speed Myrinet network (full-duplex 1.2 Gbps) or with Fast Ethernet (100 Mbps). The AltaCluster is also outfitted with an external monitoring network that allows supercomputing-like features; individual processor temperature monitoring, power cycling, or reseting of the individual nodes. Each node contains components similar to those in a commodity PC, for instance, a 100 MHz system bus, 512KB cache, 512 MB ECC SDRAM, and a 6.4 GB hard drive. However, the nodes do not require video cards, keyboards, mice, or displays.

The Myrinet topology (see Figure 3) consists of four octal 8-port SAN switches (M2M-OCT-SW8 Myrinet-SAN), each with 16 front-ports attached to each of 16 nodes, and eight (8) SAN back-ports attached to the other switches. The Fast Ethernet uses a 72-port Foundry switch with Gigabit Ethernet uplinks to the vBNS and Internet.

1.2. Project Goals

The motivation behind Roadrunner is "capability computing," where users intend to maximize one or more of the available computational resources and to continuously execute an application over a long period of time (on the order of days to weeks). We are attempting to solve grand challenge class problems that require large amounts of resources and scalable systems unavailable in Beowulf PC clusters. What the Roadrunner project demonstrates is a well-packaged, scalable cluster computer system that gives the best computational power per dollar.

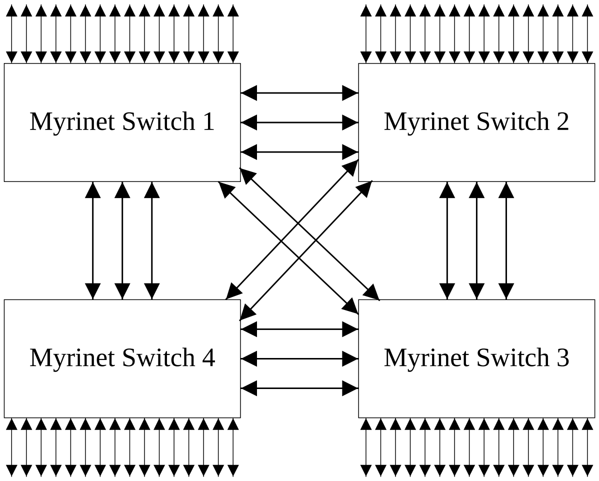

Figure 3. An example Myrinet topology for 64 hosts using four Octal 8-port SAN switches.

2. Linux Operating System

2.1. Motivation for Linux

Linux was chosen as a base for the SuperCluster for several reasons. The scientific community has a history of success with freely-available applications and libraries and has found that development is improved when source code is openly shared. Application codes are generally more reliable because of the multitude of researchers studying, optimizing, and debugging the algorithms and code base. This is very important in high performance computing. When researchers encounter performance problems in a commercial computing environment, the vendor must be relied upon to fix the problem which may impede progress.

Free operating systems have recently matured to the point where they can be relied upon in a production environment. Even traditional HPC vendors like IBM and SGI are now supporting Linux. We feel that this is clearly the direction in which the world of high performance computing is moving.

The particular version of Linux we are running on the SuperCluster is the Redhat Linux distribution version 5.2, with a custom compiled SMP kernel 2.2.10. It is important to note that our system would work equally well under another Linux distribution or another free OS altogether such as FreeBSD or NetBSD.

There are several other reasons for using Linux: low to no cost, SMP support performs well in practice (continuously improving), and wealth of commercial drivers and applications that are easily ported or written for Linux. Ease of development is another very important factor in choosing

Linux. Because of the availability of Linux, scientists may develop algorithms, code, and applications on their own locally available machines and then run on Roadrunner with little or no porting effort. Nearly all of the software used on Roadrunner is freeware making it very easy for users to setup a similar system for development at their home site.

2.2. Linux Customizations

Although the Roadrunner is based upon the 5.2 release of Redhat Linux, several customizations were needed for interoperability with system software as well as increased performance.

One of Roadrunner's extra operating system components is the 'GM' Myrinet messaging layer. GM is distributed by Myricom in either source or binary format. We have used the binary distribution thus far, and tend to upgrade our GM fairly soon after each release to gain additional performance and stability. Myricom releases their binary distributions of GM for Linux based upon a given kernel revision. This drives what Linux kernel Roadrunner has installed (2.2.10 at current writing). Since Redhat 5.2 is shipped with kernel version 2.0.36, we must maintain a custom kernel for Roadrunner. GM is shipped as a module that will run on an SMP system, so our kernel must be built with support for loadable modules. The GM driver is then dynamically loaded at boot time.

We maintain a list of customizations that must be made to Linux and related system software for it to scale up to tens, hundreds, or even thousands of nodes. For instance, one of the supported interactive shells on Roadrunner is '`tcsh`'. Because some parallel jobs generate a long command-line argument list (for instance, including each host machine name), we found that jobs were failing due to "Word too long" messages from the shell. With more than 1010 characters on the command line, the jobs were failing. This problem was due to a limitation in `tcsh` in which words could be no longer than 1024 characters. The define "BUFSIZE" which limits word size was set to 1024 by default. Our solution was to compile a new shell with the BUFSIZE increased 4-fold since the machine list for our full 64 nodes required over 1400 characters.

3. Systems Software

There are a number of core components that we have identified for running a production SuperCluster like the Roadrunner. As discussed in previous sections, the first is a robust UNIX-like operating system. After that the only add-ons are programs that allow one to use the cluster as a single, parallel resource. These include MPI libraries, a parallel job scheduler, application development software, and programming environment.

3.1. Message Passing Interface (MPI): MPICH

MPI provides a portable, parallel message-passing interface. On Roadrunner we are using MPICH, a freely available, portable implementation of the full MPI specification developed by Argonne National Labs [8]. We chose MPICH over the commercially available MPI implementations for several reasons. MPICH has been very successful, and is essentially the de-facto standard for freely-available MPI implementations largely because it is the reference implementation of the MPI standard. Commercial MPI implementations also tend to be operating system dependent. MPICH on the other hand is available in source code form and is highly portable to all the popular UNIX operating systems including Linux.

Two general distributions of MPICH are available on Roadrunner. First, the standard Argonne release of MPICH is available for those using the Fast Ethernet interface. Also available is MPICH over GM which uses the Myrinet interface. MPICH-GM is distributed by Myricom. Within each of these distributions two builds of MPICH are available based upon the back-end compilers. One set uses the freely available GNU project compilers, and the second uses the commercial Portland Group suite of compilers. Thus, the user has four flavors of MPICH to choose from on Roadrunner. All MPICH distributions on Roadrunner are configured to use the Secure-Shell (SSH) remote login program for initial job launch. As we will discuss later, this provides a more secure alternative to the '`rsh`' default without compromising performance.

3.2. Parallel Job Scheduling

It is also very helpful in a clustered environment to deploy some type of parallel job scheduling software. Users running a parallel application know beforehand that there is a finite amount of resources such as memory and CPU available for their application, so they naturally want to use all of these resources. In this capability computing environment, it is not sufficient to rely solely on a friendly user policy. Many of the codes on Roadrunner are very time consuming, and the risk of job conflict is therefore great. This risk is compounded further as the cluster size increases, and the more users it accommodates. Parallel job scheduling software eliminates these types of conflicts and at the same time enforces the particular site usage policy.

For parallel job scheduling, Roadrunner uses the Portable Batch System, or PBS, which was developed at NASA Ames Research Center [9]. PBS is a job scheduler that supports management of both interactive and batch style jobs. The software is configured and compiled using popular GNU autoconf mechanisms, which makes it quite portable including to Linux. PBS is freely distributed in source-code form under a software license that permits a site to use PBS locally, but does not allow one to redistribute it. PBS has some nice features including built-in support for MPI, process accounting, and an optional GUI for job submission, tracking, and administration of jobs.

3.3. Application Development: EGCS and Portland Group Suite

Lastly, parallel application development requires parallel compilers and debuggers for clusters. Supported standard languages on Roadrunner include C, C++, Fortran 77 and Fortran 90.

Our system uses a mix of compilers. For C, C++, and Fortran 77, we use EGCS, the experimental version of the GNU compilers optimized for Intel x86 processors. On top of these we have the MPI libraries to provide parallelism. However, applications also may be written in Fortran 90, and currently there is little free software support for Fortran 90 like there is for the other languages. To address this, we have installed a suite of commercial high performance FORTRAN compilers from the Portland Group. This suite also includes C and C++ compilers and a debugging tool.

4. Usage Policy, Access, & Security

As mentioned previously, system software such as a job scheduler allows the enforcement of the particular site usage policy. This section provides details on how this policy may be enforced on clusters through systems software. Scalability issues will also be discussed.

4.1. Access Control Methodology

As with many supercomputing sites, our local usage policy dictates that only one user (or group) be allowed to use a particular compute node at a time. Again, for quality of service, we do not want to rely only upon a friendly user policy where users compete unfairly for system resources. Thus, we need a system where users are unable to circumvent the job scheduler either maliciously or accidentally. That is, users should not able to login or use resources on the compute nodes without first being granted an allocation through PBS. This prevents users from directly launching parallel programs on the nodes and potentially interfering with other running jobs.

In its default state, each compute node has a persistent '`/etc/nologin`' file in place which disables all user logins except for the root (administrative) user. Support for the '`nologin`' file is standard under Linux and many other UNIX flavors. With our implementation, the '`/etc/pbsuser`' file overrides '`/etc/nologin`' if it exists. Thus, any user listed in the `pbsuser` file on a node is permitted access to that node. This file is automatically created when a job is allocated nodes through PBS and then removed when the job terminates.

4.2. Access Control Implementation

Node access control on a SuperCluster may be enforced using the job scheduler (PBS), the Secure-Shell (SSH) secure `rsh`/`rlogin` replacement, and the '`masshosts`' tool.

On Roadrunner, user access to nodes can only take place through SSH. SSH is configured with a particular option ('`--with-nologin-allow`') that allows it to check an arbitrary file (`/etc/pbsuser` in our case) for users who are allowed to override the `nologin` file and access the node.

The `pbsuser` file is manipulated by the PBS job scheduler. PBS allows the system administrator to create a 'prologue' and an 'epilogue' script that handle job pre- and post- processing, respectively. These scripts are run as root (that is, with full system privileges), must be executable, and can be written in the programming language of choice. Our prologue is responsible for creating an `/etc/pbsuser` file on all scheduled nodes containing the username of the scheduled user. This allows that user to override `/etc/nologin` and run commands on those nodes. Executed when the job terminates, our epilogue script deletes the `/etc/pbsuser` file on all scheduled nodes which locks the user back out.

Masshosts is a freely available Perl script written by John Mechalas [7] that can be used to execute arbitrary commands on groups of hosts in an efficient and flexible manner. Masshosts is called from both prologue and epilogue and is responsible for running the remote SSH commands on the compute nodes to manipulate the `pbsuser` file. The advantage of using masshosts rather than a sequential command loop is speed. The script has an option that allows the system administrator to specify the number of remote SSH connections to run in parallel. For example, with masshosts set to run 16 parallel sessions, it takes under 10 seconds to finish prologue and epilogue processing on Roadrunner's full 64 nodes.

4.3. Thoughts on Future Limitations & Scalability

This access control method has proven effective and reliable in our current (64 compute node) environment. It is also a portable implementation since it uses pre-existing system software mechanisms and requires no source code modifications of these programs. However, as the number of compute nodes grow on a SuperCluster, the question of whether the existing implementation will continue to perform adequately arises. One possible limitation is in the pbsuser file creation/deletion step carried out by masshosts. With 512 compute nodes for example, the amount of time needed by masshosts to finish execution of remote commands on all nodes may well exceed what is reasonable by user and administrator expectations.

A more scalable solution might be to house the pbsuser file on a read-write network filesystem that is shared across all compute nodes rather than on the local filesystems of each node. Currently the '--with-nologin-allow' SSH option only supports lookups from a static filename specified at compilation time. However, since SSH is open-source, it would be feasible to modify the program to perform nologin-allow lookups on a shared, network file named dynamically based on a local hostname. For example, SSH on node 15 would look to file '/shared/pbsuser-node15', node 422 to '/shared/pbsuser-node422', etc. for usernames that should be granted local access. Then, the PBS prologue and epilogue scripts could create and delete one pbsuser file per node on the shared filesystem instead of connecting to each node with masshosts to accomplish this. This would undoubtedly provide a faster and more efficient implementation.

5. Roadrunner on the GRID

5.1. What is a Computational Grid?

A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities [6]. This infrastructure is analogous to the electric power grid; it allows a large-scale sharing of resources through a common "plug-in" interface with little dependence or thought as to the supplier of power. Although computers are faster than ever, more computational power is needed to solve grand challenge problems. The Grid can help solve these problems by making significant resources available at a reasonable cost. In addition, the Grid may connect geographically distributed resources such as SuperClusters and supercomputers, scientific instruments, as well as data repositories, visualization suites, and people.

5.2. Globus & GUSTO Testbed

One mechanism for allowing communication between the nodes of a computational grid is Globus (www.globus.org). The Globus Infrastructure is a high performance, distributed computing toolkit that allows ready access to geographically distributed resources as described above. The Roadrunner SuperCluster is fully integrated with Globus and GUSTO (Globus Ubiquitous Supercomputing Testbed Organization), and allows scientists from all over the world to use all of the Globus resources for computation. GUSTO combines 40 sites with over 2.5 TeraFlops of computing power in the largest demonstration of a computational grid.

6. Experimental Results & Applications Performance

Next we analyze the system performance of a SuperCluster using both low-level benchmarks and application kernels. Many benchmarks have been run on Roadrunner to test everything from computational power of the nodes to communication across the interconnection networks. The most important aspect of Roadrunner's performance is not the embarrassingly parallel applications, for instance, Monte Carlo simulations, but those that require large amounts of computational power and inter-processor communication. First we provide experimental results from the communication subsystem, followed by empirical results for two important grand challenge class applications for computational physics and 3-dimensional numerical relativity simulations.

6.1. Various Benchmarks and their Results

6.1.1 Myrinet vs. Ethernet performance comparisons

The time to send and receive a message is directly affected by how much bandwidth is available at the time of transmission, and by the effectiveness of the routing method. In this context, we compare Myrinet performance with that of Fast Ethernet. In order to test Roadrunner's communication capabilities, we used a standard ping-pong test for both latency and bandwidth over an increasing, user defined message size.

In Figure 4, we see both Fast Ethernet and Myrinet time performance against application-level messages ranging in size from zero to one megabyte. Myrinet has a latency of approximately 28.0 microseconds while Fast Ethernet's average is about 178 microseconds as seen in Figure 5. Not only does Myrinet have an order of magnitude less latency than Fast Ethernet, but it is continuously faster for all sizes of packets.

In Figure 4, the time period required to send and receive a message between two nodes is displayed. Myrinet has blatant repeatability and consistency, combined with excellent speed (unlike Ethernet's erratic performance).

In Figure 6, Fast Ethernet and Myrinet bandwidth performance is compared again in message sizes ranging from zero to one megabyte. Myrinet has a peak achievable bandwidth of about 69.1 MB/s while Fast Ethernets' peak is around 8.09 MB/s (Fast Ethernet is switched). Unlike Ethernet's use of the operating system's TCP stack, Myrinet

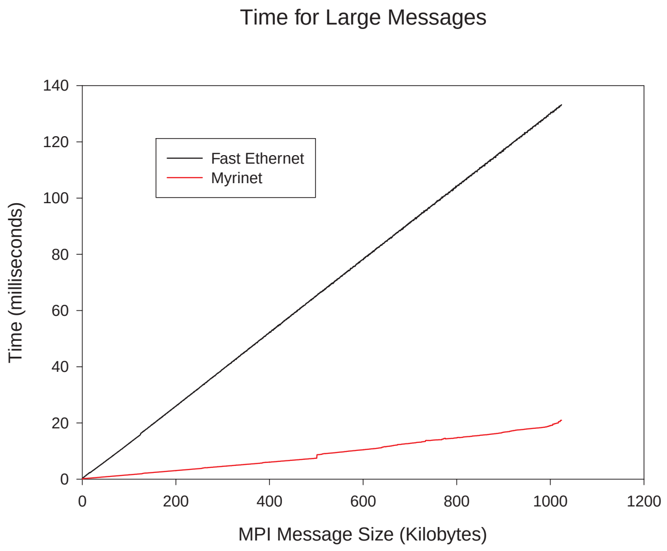

Figure 4. Comparison of Fast Ethernet and Myrinet Performance for Large Messages.

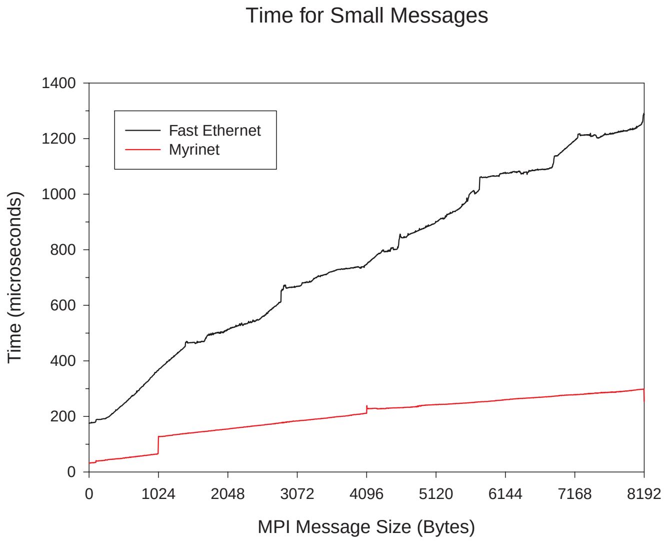

Figure 5. Comparison of Fast Ethernet and Myrinet Performance for Small Messages.

drivers bypass the operating system and use a *zero-copy* mechanism that copies a message directly from source to destination application buffer without the need for unnecessary local copying of the message. It is also interesting to note that Myrinet is using different communication algorithms based on the size of the message being transfered. This is indicated by the saw tooth markings of the Myrinet performance curve in Figure 6.

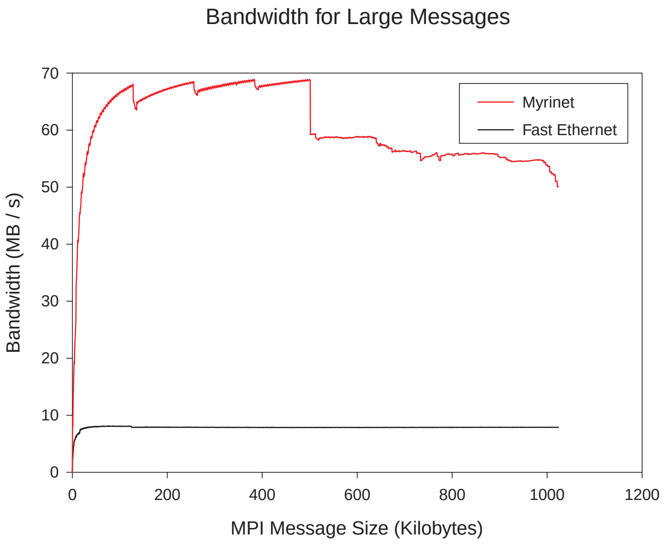

Figure 6. Comparison of Fast Ethernet and Myrinet Bandwidth.

6.1.2 MILC

The MIMD Lattice Computation (MILC) [4] is a body of high performance research software for modeling SU(3) and SU(2) lattice gauge theory on several different (MIMD) parallel computers in current use. The MILC benchmark problem used in this paper is a conjugate gradient algorithm for Kogut-Susskind quarks. Note that L refers to the problem size, and that $L = 4$ means that there is a 4^4 piece of the domain on each node.

In Figures 7 - 9, this MILC benchmark was run on two Linux clusters. The Indiana University (IU) Physics Linux cluster is a Beowulf-class 32-node Fast Ethernet cluster with a single 350 MHz Pentium II processor, 64 MB RAM and 4.3 GB disk per node. Looking at the single node benchmarks (see Figure 7), we see that the for small problem sizes that fit completely or mostly in the cache, the Roadrunner SuperCluster with its 450 MHz processor is faster than the 350 MHz system. For $L > 6$, however, memory access becomes a limiting factor, and there is little performance difference. For $L \geq 6$, the Roadrunner SuperCluster achieves greater than 60 MF/node for almost all

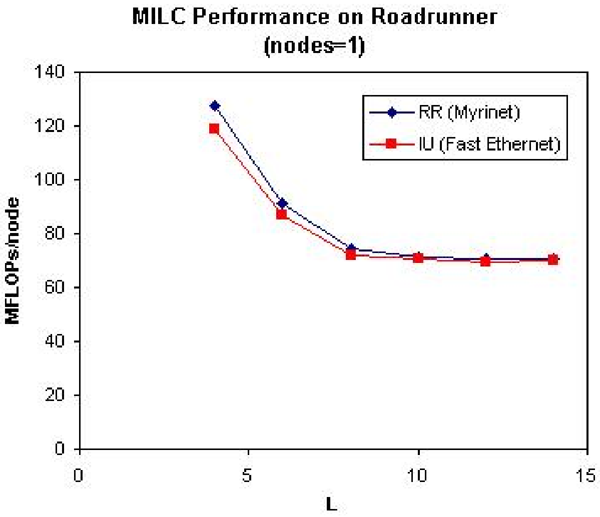

Figure 7. MILC single node benchmark.

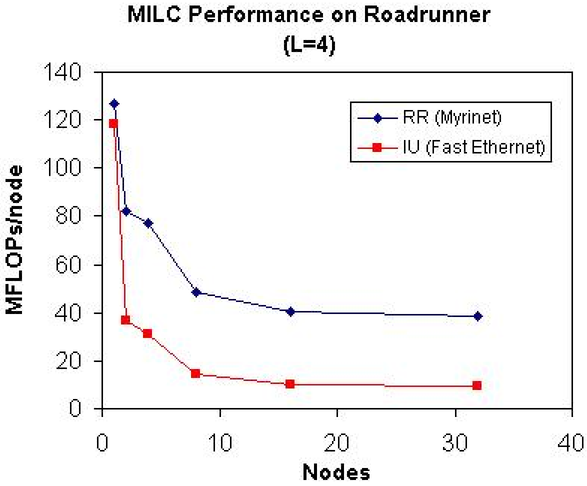

Figure 8. MILC Benchmark with $L = 4$.

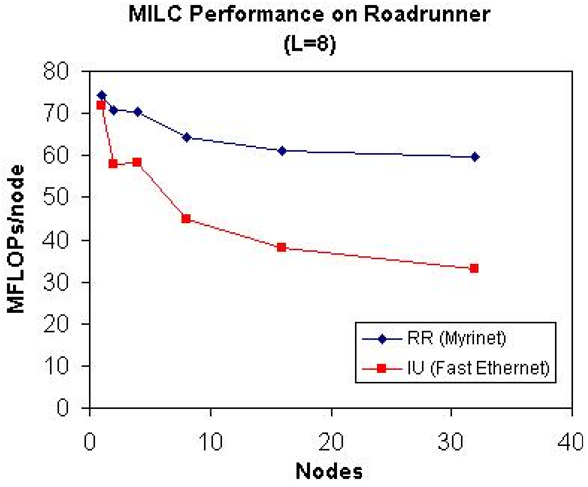

Figure 9. MILC Benchmark with $L = 8$.

cases. For the IU Beowulf, for $L \geq 8$, on up to 32 nodes, the performance is near 50% of the SuperCluster performance. As a point of reference for $L = 8$ on 16 nodes, the Cray T3E-900 achieves 76 MF/node and on an Origin 2000 (250 MHz) MILC achieves 119 MF/node.

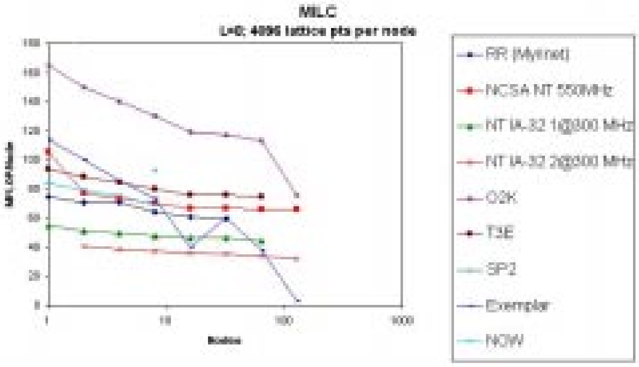

Figure 10. Comparison with additional architectures is shown below for a L=8 case. Non-Roadrunner data - courtesy of NCSA NT Cluster Group.

In Figure 10, the MILC benchmark for a fixed problem size is run on a variety of architectures varying the machine size. Here, the SuperCluster performs quite well compared with traditional supercomputers.

6.1.3 CACTUS

CACTUS [1] is a high-performance computing toolkit that implements a framework for solving three-dimensional computational astrophysics problems on parallel machines. In this paper we benchmark the CACTUS grand challenge application using an instance that solves the Bona-Mass hyperbolic formulation of the Einstein equations for numerical relativity calculations.

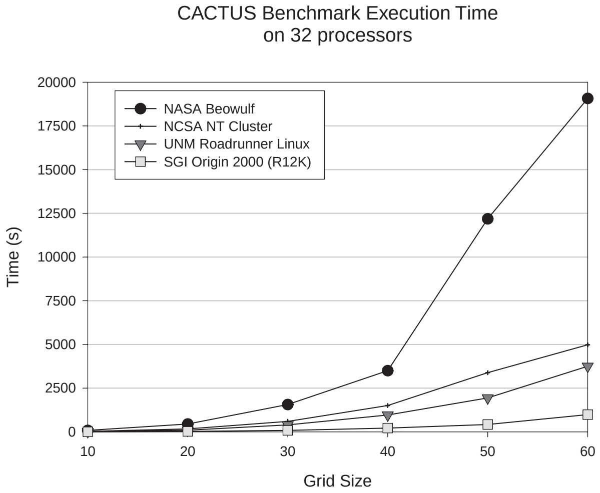

Figure 11. CACTUS benchmark running time for various machines.

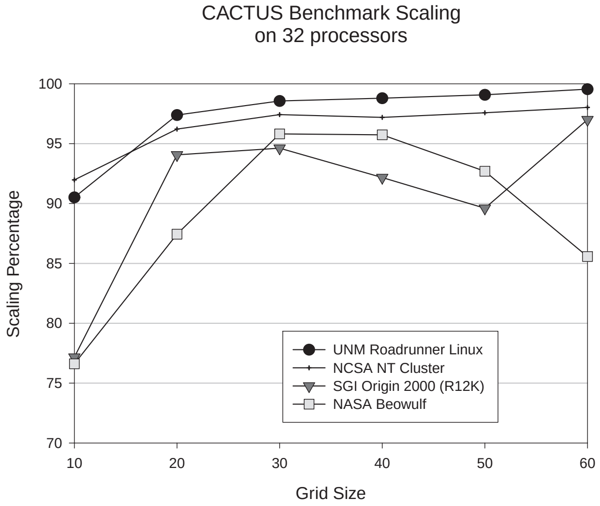

Figure 12. CACTUS scalability.

Figure 11 shows the running time for the CACTUS benchmark for 32 processors as the problem size increases. As a key, the machines are configured as follows. ‘NASA Beowulf’ is a 64-node dual PPro 200 MHz, 64MB RAM, ‘NCSA NT Cluster’ runs Microsoft Windows NT with 32 dual PII 333 MHz with 512 MB RAM, and 64 dual PII 300 MHz with 512 MB RAM, ‘SGI Origin 2000’ is the NCSA Origin 2000 with 32 R12K 300 MHz processors, and ‘RR’ is the Roadrunner SuperCluster. In Figure 12 the scalability of these machines is compared using the CACTUS benchmark and increasing problem sizes. Notice that Roadrunner contains the best scalability for this problem, more than 99 % as the problem size increases.

7. Future Directions

7.1. Maui Scheduler

PBS is one of the few commodity parallel schedulers available for use with computational clusters. This made PBS a natural choice for use in our cluster. However, PBS does not readily support several key features (e.g., backfill scheduling, flexible priority assignments, shortpool policies, etc.) that we believe are needed for effectively managing the resources provided by the Roadrunner cluster. To address these shortcomings, we have undertaken a direct port of the Maui scheduler to the Linux cluster environment. This port uses the Wikiman interface provided by the Maui scheduler and is nearing completion.

7.2. Improved Message Passing Layers: HPVM & Portals

While the newer implementations of MPICH-GM have shown significant improvement in message passing performance, we will continue to investigate alternative implementations of MPI on Myrinet. One such implementation is HPVM, the MPI implementation based on FM (Fast Messages) being developed by Andrew Chien at UCSD [5]. Because HPVM is used in the NT clusters at the University of Illinois, using HPVM on our cluster would enhance application portability between the two types of clusters as well as a more direct comparison of the Linux and NT approaches. Another implementation that we will evaluate is the Portals communication layer being developed jointly between the University of New Mexico and Sandia National Laboratories. This communication layer was originally developed for massively parallel supercomputers (including the nCUBE, Intel Paragon, and ASCI Red machines) [10] and is being adapted to the Linux cluster environment. Beyond the obvious tie to UNM, we are interested in this communication layer because it is one of the few communication layers that has been shown to scale to systems with thousands of nodes (the ASCI Red machine at Sandia has over 4,500 nodes).

7.3. Advanced Programming Models

Hardware benchmark results reveal awesome performance rates for both SMP nodes and high-performance networks in a SuperCluster; however, few applications on these hybrid SMP clusters ever reach a fraction of these peak speeds. While methodologies for symmetric multiprocessors (e.g., OpenMP or POSIX threads) and message-passing primitives for clusters (e.g., MPI) are well developed, performance dictates the use of a hybrid solution. In contrast, our approach called SIMPLE [2], a hybrid, hierarchical methodology, is a much closer abstraction of the underlying machine and is ideally suited to SMP clusters. SIMPLE provides an integrated complexity cost scheme for SMP clusters that accurately models the hierarchical memory system and networks, while retaining the benefits of the RAM model. Namely, our complexity model will be simple to apply, provide a "rule of thumb" for algorithmic complexity, and aid algorithmic designers in discovering efficient and portable algorithms on clusters.

8. Acknowledgments

We would like to thank Ron Brightwell and Rolf Riesen (Sandia National Laboratories) and Rod Oldehoeft, Peter Beckman, and Susan Coghlan (Los Alamos National Laboratory), Steven Gottlieb (Indiana University) and Robert Sugar (University of California, Santa Barbara) for their help and comparisons of the MILC code, and Ed Seidel, Gabrielle Allen, and Oliver Wehrens (Max-Planck-Institut für Gravitationsphysik and University of Illinois) for the CACTUS code. We also acknowledge Ken Segura (University of New Mexico) for his help running the Roadrunner SuperCluster. We thank Frank Gilfeather, the executive director of the High Performance Computing, Education, and Research Center, for his generous support of high-performance computing at The University of New Mexico and his leadership of the Roadrunner Project.

References

[1] G. Allen, T. Goodale, and E. Seidel. The Cactus Computational Collaboratory: Enabling Technologies for Relativistic Astrophysics, and a Toolkit for Solving PDE's by Communities in Science and Engineering. In *Proceedings of the Seventh IEEE Symposium on the Frontiers of Massively Parallel Computation*, pages 36–41, Annapolis, MD, February 1999.

[2] D. Bader and J. JáJá. SIMPLE: A Methodology for Programming High Performance Algorithms on Clusters of Symmetric Multiprocessors (SMPs). *Journal of Parallel and Distributed Computing*, 58(1):92–108, 1999.

[3] D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawak, and C. Packer. Beowulf: A Parallel Workstation For Scientific Computation. In *Proceedings of the 1995 International Conference on Parallel Processing*, volume 1, pages 11–14, August 1995.

[4] C. Bernard, T. Blum, A. De, T. DeGrand, C. DeTar, S. Gottlieb, A. Krasnitz, L. Kärkkäinen, J. Labrenz, R. Sugar, and D. Toussaint. Recent Progress on Lattice QCD with MIMD Parallel Computers. MIMD Lattice Computation (MILC) Collaboration, A Grand Challenge Application Group.

[5] A. Chien, S. Pakin, M. Lauria, M. Buchanan, K. Hane, L. Giannini, and J. Prusakova. High Performance Virtual Machines (HPVM): Clusters with Supercomputing APIs and Performance. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, Minneapolis, MN, March 1997.

[6] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[7] J. Mechalas. The Masshosts Tool. *SysAdmin Magazine*, 8(3), 1999.

[8] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. Technical report, University of Tennessee, Knoxville, TN, June 1995. Version 1.1.

[9] MRJ Inc. The Portable Batch System (PBS). textitpbs.mrj.com.

[10] S. Wheat, A. Maccabe, R. Riesen, D. van Dresser, and T. Stallcup. PUMA: An Operating System for Massively Parallel Systems. *Scientific Programming*, 3:275–288, 1994.