Cluster-Based High-Performance Computing at LANL *

Peter Beckman Susan Coghlan Rod Oldehoeft

Abstract

The Advanced Computing Lab at Los Alamos National Laboratory is engaged in cluster-based systems development to support large-scale applications and to learn how future systems and software may be designed. In this paper we describe our large SGI cluster and our experimental Linux cluster, outline some important software available on both of them, and discuss some performance results.

1 Two ACL Cluster Systems

The ACL operates both a high-capability commercial system and an experimental Linux system. Both are used for production and research.

1.1 Nirvana

The ACL supports Nirvana, an SGI/Cray system consisting of 16 Origin 2000 multiprocessors. Each SMP has 128 processors, 32GB of memory, a 284GB fibre channel disk array, and three HiPPI network interconnections. This 2,048-CPU assembly is the highest-capability (one TFLOP peak) unclassified computer system in the world (one-third the size of the ASCI Blue Mountain system). This system is used to support several DOE Grand Challenge projects and other large-scale applications: ocean and atmospheric modeling, quantum chromodynamics, numerical tokamak, modeling of multiphase flows. In addition, we support software that models the dynamics of intense particle beams in accelerators; this application is called LINAC[5], and we discuss LINAC performance on our clusters below.

We have recently announced the Terascale Computer Science Testbed[1], wherein researchers with interesting middleware can gain access to the entire system to test and demonstrate the scalability of their products. We hope to attract a diverse set of software elements, including components, libraries, frameworks, run-time systems, distributed software systems, data-visualization packages, and distance computing tools.

Figure 1: LBP PC cluster runs Linux.

1.2 Little Blue Penguin

We also experiment with more affordable Linux-based clusters. Our current system, named Little Blue Penguin (LBP), has 64 nodes, each with two Pentium II processors, 256MB of memory, 6GB disk and a Myrinet network card. The system was built by Alta Technology. In addition, four nodes connect to 500GB of RAID disk, and eight Myrinet ports connect to SGI machines for parallel scientific visualization. Our niche in the Linux cluster world is characterized by SMP nodes and fast networking technology. In Figure 1, each of eight black cabinets holds eight nodes, and four floor-standing boxes comprise RAID storage.

2 Software Is Essential

Today's high-performance computing environments, and the applications that must exploit them, have become much more complex than ever. We now build clusters of large, shared-memory parallel computers, linked together with high-speed networks, in an attempt to achieve previously unheard-

*Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36.

[1] See http://www.acl.lanl.gov/tcst for details.

of speeds and still retain a "general-purpose" capability for running diverse applications.

Demands for greater precision and realism in today's computer simulations of physical phenomena tax the imagination of the most aggressive system designers. Enhancing the accuracy of tomorrow's simulations requires simultaneously accounting for more physical, chemical and biological components. Predictive simulation is essential for making informed, science-based decisions on questions of national importance, including stockpile stewardship, global climate change, wildfires, earthquakes and epidemics.

Collecting a "pile o' PC's," connecting them with some Ethernet cards, and installing Linux and MPI is not a recipe for a general-purpose system. The LBP cluster is far more: included are high-performance networking software from the Real-World Computing Partnership[8] and Myricom, commercial compilers from PGI and KAI, a batch job scheduler (PBS), and remote system administration tools, all in support of production computing for real applications. This LBP software comes from many sources, and collecting, installing and integrating components is a significant effort. An important step toward making clusters easier to use would be a comprehensive package containing all the noncommercial components.

ACL teams have developed several software technologies to "tame the complexity" of SMP cluster-based computing.

- *Object-oriented frameworks:* POOMA[2] is a C++ library supporting common abstractions for computational science applications on high-performance parallel computers. POOMA allows programmers to concentrate on developing applications, without having to worry about architecture or network specifics, confident that high execution speed and cross-platform portability will be maintained.

  PETE[4] is an extensible implementation of the expression template technique[10]. It eases the task of exploiting this technique, which is required for high-performance C++ execution.

- *Scalable runtime support:* Tulip is an implementation of the HPC++[3] interface specification. It provides high-performance inter-machine communication via one-sided operations on global pointers, lightweight threads with Posix and Java APIs, and fast global synchronization.

  SMARTS[9] is a library for integrated task/data parallelism and inter-machine communication; small non-blocking threads execute cache-aware pieces of work overlapped with communication operations, all under the control of a dependence graph encompassing many data-parallel source statements.

- *Tools for performance analysis:* TAU[7], originated at and continued with the University of Oregon, is a set of tools that allows one to analyze the performance of parallel application programs. TAU macros are inserted into source code for program instrumentation, the program is compiled and run to generate a trace file, and the trace file is viewed with TAU tools, or a third-party tool such as VAMPIR.

- *Software component support:* PAWS[1] provides mechanisms for combining previously developed parallel applications into a larger whole, incorporating parallel data transfer among components. The applications may run on different machines, with different numbers of processors, and different data distributions.

  SILOON is an automated "glue code" generator for enabling access from various scripting languages to scientific libraries, thus easing the prototyping of high-performance applications.

  PDT is a toolkit for collecting a database of information from source code and making it available for other tools (TAU, SILOON) to access in a uniform way.

All of this software runs on both Nirvana and LBP, to provide a similar computational environment for applications. Software is freely available for noncommercial use[2]. For a more complete description of this software, see [6]

3 Performance

These clusters have helped us to understand trade-offs in system configuration. We describe three sets of performance tests: the STREAMS memory benchmarks, NAS benchmarks, and the LINAC application discussed above.

3.1 Memory Performance Tests

The true bottleneck to high-performance computing is memory system performance. Table 1 compares performance on the STREAM[3] benchmark for the Cray/SGI Origin 2000, the IBM SP2, and Intel and Alpha workstation products. We see that commodity products are competitive or exceed the STREAM performance of integrated systems.

Table 1: STREAM Memory Benchmark

Machine	COPY	SCALE	ADD	TRIAD
SGI O2K	178.1	184.4	198.4	192.2
IBM SP2-thin-II	320.0	320.0	352.0	352.0
Dell PII/450	188.2	173.0	213.3	188.2
DEC ES20 21264s	1087.7	1244.0	1179.0	1323.0

Table 2 gives the bisection bandwidth of several systems. The SGI O2K has a constant bandwidth until cross-box communication is required. The IBM SP2 has a scalable bandwidth, and a Myrinet-based cluster solution has a superior bandwidth, even when handicapped by current PCI performance.

Table 2: Bisection Bandwidth

Machine	32	64	128	256
SGI O2K	48.8	48.8	48.8	6.4
IBM SP2(4-way)	4.7	9.4	18.7	37.4
Myrinet (PCI)	20.2	40.3	80.6	161.0

3.2 NAS Benchmarks

The NAS Benchmarks[4] exhibit a variety of program behaviors that stress systems, and they resemble useful applications. We use them to obtain some general answers about how a cross-section of applications can be expected to perform.

[2] Access it at http://www.acl.lanl.gov/software/.
[3] See http://www.cs.virginia.edu/stream.
[4] See http://science.nas.nasa.gov/Software/NPB.

- Claims are sometimes made that low-cost clusters can easily achieve higher performance than highly integrated systems that cost much more. In Figure 2 we see that for NAS benchmarks this is not true. The superior system bandwidth matters in real applications. Here

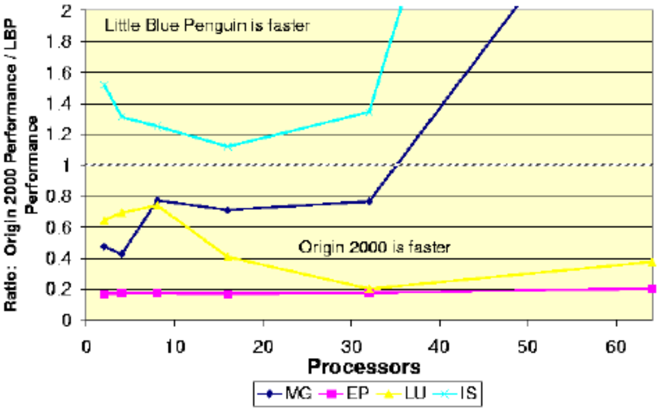

Figure 2: Nirvana vs. LBP on NAS benchmarks.

we see two more interesting phenomena. First, outstanding integer performance for the Pentium yields better IS performance on our Linux cluster. Second, the NAS numbers (as reported) use all 128 processors of an O2K (hence the irrelevant lines connecting O2K performance on 64 and 128 processors). This is inadvisable because there are OS processes providing a small overhead. We limit our own tests on Nirvana to 126 processors per SMP. This overhead is not seen on LBP because we have removed the system daemons that provide general-purpose services not needed on our dedicated cluster.

- Dual-processor systems are available at substantially less cost than twice as many single-processor systems. Folklore has held that not enough price-performance gain is available with dual-processor nodes (based on assumptions about memory system performance), but Figure 3 shows that this is not true. If a dual-CPU node costs 50% more than a single-CPU node, all the benchmarks show a better price-performance with dual processors. The horizontal axis gives the number of processors used, and the vertical axis gives the ratio of price-performance values (curves below 1.0 favor dual-processor nodes). More recent studies have shown that memory systems are improving to reinforce this preference for dual (or even more) processors. However, single-CPU systems will usually do better than multiple-CPU systems in raw performance, in comparisons with equal numbers of CPUs, because more memory-system hardware is deployed in the former case.

- PC-based systems commonly use inexpensive, 100Mb Ethernet as an interconnection medium. We have found that fast networks are essential to using such systems for general-purpose work, and can in fact have better price-performance. The price-performance of Myrinet vs. 100 Mb Ethernet connections is seen in Figure 4. The vertical axis gives the ratio of 100BT to Myrinet price-performance values (curves less than 1.0 mean that Myrinet has higher price-performance). This result is counter-intuitive, since a Myrinet card is much more expensive as a 100BT Ethernet card. Both node processors were used in this comparison.

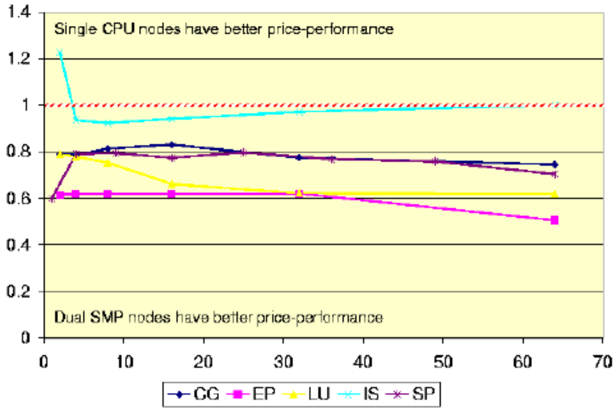

Figure 3: Price-performance comparison between single and dual processor nodes.

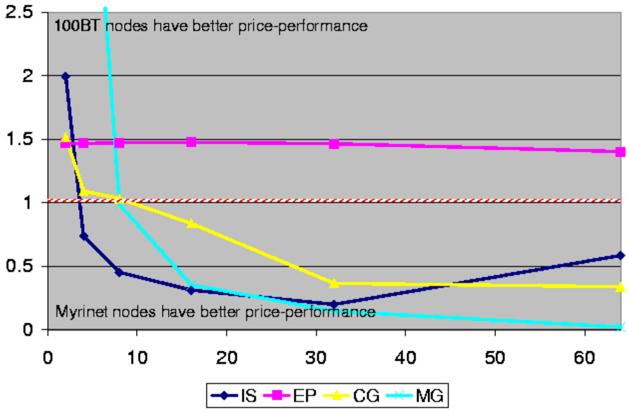

Figure 4: Price-performance comparison between 100BT and Myrinet networks.

3.3 LINAC Performance

Of course, the most useful benchmarks are the applications one expects to run. LINAC provides another perspective on relative performance of technologies on real applications. It spends most of its time computing the forces on each particle in the linear accelerator simulation, and uses a standard FFT-based parallel particle-in-cell algorithm. Parallel FFTs require much data motion, since for each dimension the parallel grid must be transposed. The rest of LINAC's execution is relatively localized (e.g., in a gather-scatter section), requiring communication for data in nearby regions of space.

We have some observations about LINAC running on Nirvana and LBP.

- In Figure 5 we plot the performance ratios between our LBP and Nirvana clusters. For the entire application, LBP performs at about 70% of Nirvana, across a wide range of processors. For the FFT portion, requiring a lot of data movement, the tightly integrated Nirvana system excels.

- Figure 6 compares 100BT Ethernet with Myrinet networking, using one or both node processors.

A great deal of information about LINAC performance is available in this figure.

- The shapes of the left ends of the ratio curves is typical: 100BT is often observed to yield good performance for

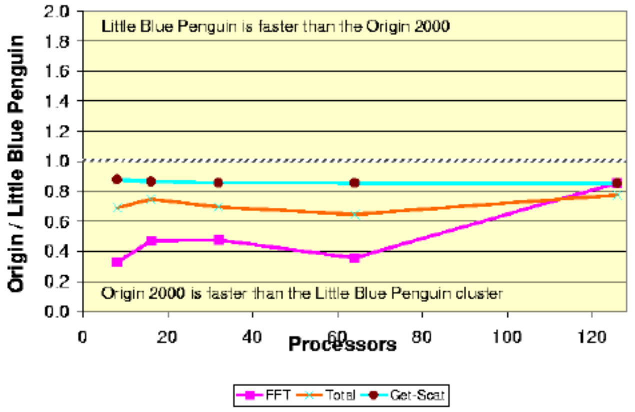

Figure 5: LINAC performance: Nirvana vs. LBP.

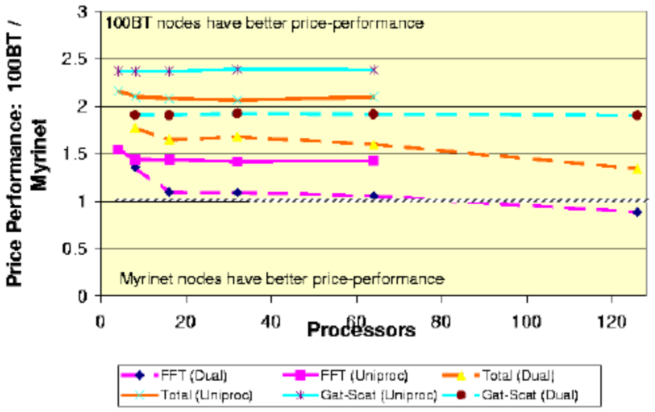

Figure 6: LINAC price-performance: 100BT vs. Myrinet, single vs. dual processors.

small configurations running applications like LINAC that are not too communication-intensive.

- The solid curves are for runs with one CPU and one network adapter per node. In general, the Myrinet adapter is not sufficiently loaded by one CPU to be cost-effective. The communication-intensive portion of LINAC (FFT) loads the Myrinet adapter more heavily, and this portion has a lower ratio. The gather-scatter portion of LINAC is easily parallelized, and shows a higher ratio, with the entire application lying between these two curves.

- The dashed curves are for runs with two CPUs and one network adapter per node. Here, the FFT portion of LINAC loads the Myrinet adapter sufficiently so the price/performance is equal to the 100BT case. The entire application shows better price/performance for 100BT, but the trend is downward for larger configurations.

These and many other experiments help us to characterize the tradeoffs and economies of putting together high-performance commodity Linux clusters that compete as well as possible with highly integrated supercomputers.

4 Summary

The ACL operates commercial and experimental clusters of SMPs to support applications of relevance to LANL, and to experiment with configurations and software to understand how to build and operate future systems. Nirvana is largely a production system, but we will also collaborate with other groups who have interesting middleware to help develop and demonstrate scalability. Our current Linux cluster has shown us that both multiple-processor nodes and fast networking are cost-effective for real applications. It also has a growing list of applications.

References

[1] P. Beckman, P. Fasel, and W. Humphrey. Efficient coupling of parallel applications using PAWS. In *Proc. High Performance Distributed Computing Conf.*, July 1998.

[2] J. Cummings, J. Crotinger, S. Haney, W. Humphrey, S. Karmesin, J. Reynders, S. Smith, and T. Williams. Rapid application development and enhanced code interoperability using the pooma framework. In *Proc. SIAM Workshop on Object-Oriented Methods for Interoperable Scientific and Engineering Computing*, October 1998.

[3] D. Gannon, R. Bramley, T. Stuckey, J. Villacis, J. Balasubramanian, E. Akman, F. Breg, S. Diwan, and M. Govindaraju. Component architectures for distributed scientific problem solving. *IEEE Computational Science and Engineering*, 5(2), 1998.

[4] S. Haney, J. Crotinger, S. Karmesin, and S. Smith. Easy expression templates using PETE, the portable expression template engine. Technical Report LA-UR-99-777, Los Alamos National Laboratory, 1999.

[5] W. Humphrey, R. Ryne, T. Cleland, J. Cummings, S. Habib, G. Mark, and J. Qiang. Particle beam dynamics simulations using the pooma framework. In D. Caromel, R. Oldehoeft, and M. Tholburn, editors, *Computing in Object-Oriented Parallel Environments, Proc. ISCOPE 98*, December 1998.

[6] R. Oldehoeft. Taming complexity in high-performance computing. In *Proc. Int. Symp. on Computational Science*, May 1999. To appear.

[7] S. Shende, A. Malony, J. Cuny, K. Lindlan, P. Beckman, and S. Karmesin. Portable profiling and tracing for parallel, scientific applications using C++. In *Proc. SPDT'98: ACM SIGMETRICS Symp. on Parallel and Distributed Tools*, pages 134–145, August 1998.

[8] H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato. PM: An operating system coordinated high performance communication library. In *Lecture Notes in Computer Science*, volume 1225, pages 708–717. Springer-Verlag, April 1997.

[9] S. Vajracharya, S. Karmesin, P. Beckman, J. Crotinger, A. Malony, S. Shende, R. Oldehoeft, and S. Smith. SMARTS: Exploiting temporal locality and parallelism through vertical execution. In *Proc. Int. Conf. on Supercomputing*, 1999.

[10] T. Veldhuizen. Expression templates. *C++ Report*, 7(26), 1995.