Invited Talk

Beowulf-class Clustered Computing: Harnessing the Power of Parallelism in a Pile of PCs

Thomas Sterling
California Institute of Technology and NASA Jet Propulsion Laboratory
1200 E. California Blvd MC 158-79
Pasadena, CA 91125
tron@cacr.caltech.edu
http://www.cacr.caltech.edu/~tron

Beowulf-class systems represent a new generation of scalable high-end computing that provides a revolution in price-performance for many types of applications. Beowulf exploits the unprecedented advances in mass-market commodity off-the-shelf (M^2COTS) technology and combines these with the emergence of community standards in programming methodologies for parallel execution. Beowulf integrates these two developments into a single family of highly accessible distributed computing systems. As an example, in Fall 1996 a 16 processor Beowulf system costing less than $50,000 achieved a sustained performance of 1.2 Gflops on a 2 million particle gravitational N-body simulation. A year later, this same system could be procured for approximately $35,000. Sustained performance of more than 10 Gflops have been achieved at multiple sites with Beowulf systems comprising between 100 and 200 processors costing in the range of $300,000 or approximately $30/Mflops sustained price-performance. Recent advances in the last few months in a new generation of microprocessor and aggressive pricing are expected to improve this by more than a factor of two in the immediate future with the prospect of $10/Mflops possible within less than two years. As a result, the 1997 Gordon Bell prize for price performance was awarded to a joint NASA and DOE team for Beowulf. This paper briefly introduces the basic principles of Beowulf class computing.

Beowulf is a synthesis of widely available hardware and software for low-cost medium- to high-end computing. It was driven by several advances that, in the aggregate, have enabled breakthrough price-performance while retaining a dominant conventional paradigm for parallel computing. An important factor is the reduced vendor commitment to the high end of the computing spectrum, particularly for applications in science and engineering. This is driven by a relatively small market for such systems combined with the high development cost and lead times involved. The result has been a shrinking vendor base and a series of system architectures that have exhibited successive incompatibilities, requiring rewriting of many applications across platform generations. A second key factor has been the closing of the performance gap between high-end microprocessors and those employed in the consumer market. Today, Intel Pentium II parts are becoming available with peak throughputs of 400 Mflops, and DEC Alpha microprocessors are being packaged within a framework of PC interface standards and at competitive pricing. System software, too, is available on PCs that rival, if not exceed, high-end systems in sophistication and robustness. The Linux operating system is the single most widely employed Unix-like operating system available and is competitive with other vendor software of this class. But perhaps most importantly is the recent availability of mass-market network hardware capable of handling the communications loads required by applications. With the advent of low cost 100 Mbps Fast Ethernet driver cards and switches, clusters can be assembled with adequate bandwidth to support many application types and at balanced cost with respect to the rest of the system resources. This will only improve with GigaBit Ethernet as its price drops as well. Finally, the MPP scientific computing community has evolved a major paradigm for parallel computation known as message passing which runs on all major vendor systems. PVM and MPI are industry standards and free software libraries are readily available for PC grade hardware. These important recent trends have provided the foundation and framework for enabling Beowulf-class computing.

A Beowulf-class system is a distributed memory cluster computer comprising components available in the mass market. Processor/motherboards, memory, disk drives/controllers, and network hardware make up the major subsystems and are available from multiple vendors. Market competition has continually driven prices down and performance and reliability up over the last few years. A cluster of PCs is simply an ensemble of off-the-shelf personal computers interconnected by Ethernet or other networks such as Myrianet. While the interconnect topologies may vary widely across Beowulf implementations, the differences are in bi-section bandwidth and latency while maintaining a consistent logical relationship among nodes. Historically, the scientific and engineering computing community has employed variations of the Unix operating system as the environment of choice from scientific workstations, through middle level servers,

to supercomputers and MPPs. Beowulf also employs Unix-like system software with the added requirement that the code be open and available. Linux and BSD are the two most widely used Beowulf software environments. By the nature of the physical structure, Beowulfs are message passing systems. This may be made explicit at the programmer interface or implicit through some higher abstraction. Most Beowulf systems are programmed using the popular PVM or MPI explicit message passing libraries that have become industry standards, or using some in-house optimized messaging functions. An example of a higher level programming methodology is BSP that provides a global mutable name space but a restrictive synchronization model. The actual send and receives are performed automatically by the BSP system while the programmer employs special load/store semantics. More advanced distributed shared memory software systems are also in development. These packages are now being made conveniently available through the series of RedHat CDs used to distribute Linux. While there are no restrictions implied in the application domains to which Beowulfs can be applied (performance aside), Beowulf-class systems primarily target the scientific and engineering community where performance for large applications is important. Such problems are both compute intensive and memory intensive, even requiring large amounts of secondary storage to hold large data sets. Beowulf-class computers are also in use by the computer science community as a low-cost, easily accessible testbed for developing and testing new concepts. Probably the most rapidly growing area for Beowulf is in education where they make ideal platforms for providing hands-on experience in a teaching setting. Beowulfs are even being installed in some high schools.

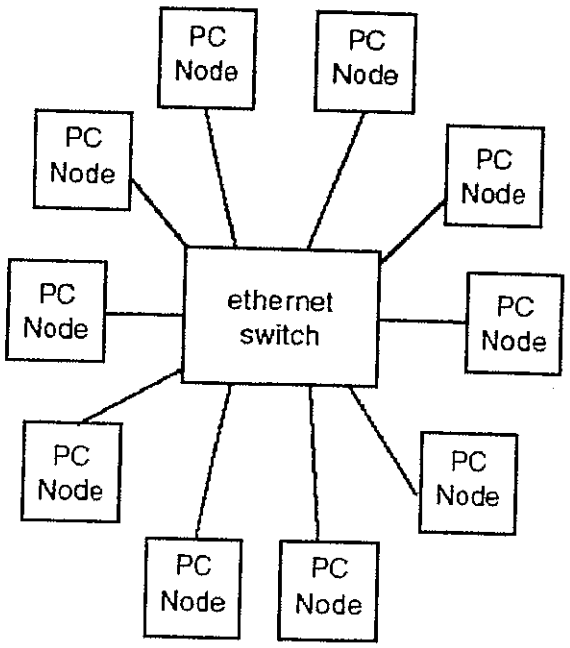


Figure 1 Beowulf Network Layout

While a major motivation for Beowulf is that it offers the best price-performance of any general purpose computing system class, there are a number of other important factors that make Beowulf the approach of choice for many circumstances. PCs and workstations are the first platforms in which new technologies are incorporated. While large MPPs take years to design and deploy once a technology becomes available, PCs are immediately available. Thus, Beowulf-class system evolution responds rapidly to technology trends and the best technology is always available for implementation of the next Beowulf. Because the commodity market for desktop PCs is so vast, many vendors offer essentially the same packages made possible by the existence of industry-wide interface standards. Thus, there is little sensitivity to the decisions of any single vendor other than possible improvements in price and performance. Beowulfs being clusters of whatever M^2COTS processor technologies are available, they are likely to always exist and can be depended upon by applications programmers and system planners. Beowulf system configurations are determined by the end user organization and not mandated by a vendor or systems integrator. There are a wide variety of alternative structures possible with Beowulfs and the exact choice can be made near the end of the system integration cycle. This is called *just-in-place* configuration and reflects flexibility not ordinarily found with other system classes. As needs change, so can the configuration of the system which is fully under the end-user control. Beowulfs are scalable. Systems of a couple of hundred processors have been implemented and plans for systems on the scale of a thousand are being carried out. Design studies have shown that existing components can be employed to scale up to a couple of thousand processors with latency increase of less than a factor of three with respect to the shortest connection between two systems (assumed unloaded nodes). By far the largest source of system software is the computer science community, mostly in academic and government laboratories. The success of the Linux phenomenon attests to the strength of leverages this important base of software investment (paid for by your tax dollars) and integrates the best of these within its environment. It should be noted that these systems are NOT hobbyist or back of the garage computers. They are mature, robust, and accessible. Routinely, their up time exceeds that of new this source of software functionality, in spite of the bad reputation of "graduate student software". Beowulf conventional commercial MPPs with periods of a hundred days or more between reboots typical. The message passing model is not as easy to use as the single memory image programming model employed on a uniprocessor workstation or mainframe. Out of necessity, it has become the dominant cross-platform API for MPPs. Beowulf lives down to the expectations created by MPPs by providing essentially the same MPI programming interface found on almost all commercial MPPs. Porting a code developed on a Beowulf to a more costly commercial parallel system is often trivial

and Beowulfs are often employed in this role of highly accessible software development systems. Finally, there is a quality of Beowulf-class systems that is difficult to quantify but which is viewed as valuable nonetheless. This may be referred to as "user empowerment". Beowulf gives control back to the user and this is proving important, even a strong motivator, for the implementation of new systems.

The Beowulf Project was initiated in 1993 in direct response to a specified requirement of the NASA HPCC Earth and Space Sciences project. The need for an end-user scientific station was identified which put important demands on price and performance. Among these was a constraint of $50,000 that was derived from the price of high-end workstations of the time. But additional demands on disk capacity and bandwidth as well as performance out stripped any of the vendor offerings by more than an order of magnitude within the price constraints. Analysis showed that this narrow requirement could be satisfied by a cluster of PCs if the right software were available. Fortunately, Linux was in its earliest releases and with source code availability was a potential research tool. Missing were adequate networking software drivers. A team was formed to integrate these components and build the necessary drivers. The Beowulf project was established. The Beowulf project developed four generations of PC clusters with multiple variations of each, performed extensive performance analysis, hosted a wide range of real world applications, and developed most of the network control software used by Linux today. Beowulf has become the single most widely recognized effort to harness PC technology for scientific computing and the term is being trademarked by the community to describe this class of system. Contributions to Beowulf-class computing are being made by many organizations both within the United States and around the world and the NASA led Beowulf project is only a small part of the overall world-wide effort today. The project itself is a combined effort at the NASA Goddard Space Flight Center, the NASA Jet Propulsion Laboratory, and the Center for Advanced Computing Research at the California Institute of Technology. But the project now benefits significantly from the excellent work and experiences at many other research centers. The term "Beowulf-class computing" emerged from the community almost two years ago and the evolution of it is a community endeavor, not under the control of any single institution or project.

With all of its advantages, Beowulf systems are not ideal for all applications and workloads. This is a consequence of its distributed nature imposing a loose coupling among computing nodes. The relatively long latency times and lower bisection bandwidth that is found within tightly coupled imposes some limits on suitability. Applications which are communication intensive, require rapid response to service requests, involve mostly short transactions between nodes, or require random access patterns for information sharing are less likely to perform efficiently on Beowulf systems than those applications which do not incur such properties. But these problems have been well understood by computational scientists for years and methodologies to manage locality in the presence of parallelism have been developed over a sequence of generations of MPPs. Latency tolerant algorithms and problem partitioning for many classes of algorithms have been devised and effectively employed for real world applications. In many cases, the computing component of an application is seen to run faster on a Beowulf than a contending MPP because of the use of most recent technology by Beowulf. But the more tightly coupled MPP will do better on the communications component of the application processing. On a per node basis, Beowulfs are seen to do from about 15% better to within 250% worse than the MPP competition. Overall, Beowulfs exhibit about an order of magnitude price-performance advantage, but this varies with application and competing systems. One class of applications that is likely to make excellent use of Beowulfs is that found in Genetic Programming. The high degree of coarse grain parallelism and relatively low communications demands typical of Genetic Programs are well suited to the balance of capabilities offered by Beowulf.

Beowulf systems are proving extremely useful in the execution of single large applications; essentially in the role of single-user systems. In many environments, this is entirely adequate and has led to a wide acceptance of such systems. However, future work is required in two areas to broaden their utility and convenience. An increase of sophistication of middle-ware is required to manage systems for multiuser simultaneous use. While for large runs a user will want the total capacity of a system, development work on a code requires much smaller configurations, and systems can be shared in time and space. Some schedulers are available and are used but have limitations that sometimes obstruct effective allocation of systems resources. The need for effective resource scheduling among multiple users will increase as the average size of Beowulf systems increase, assuming that larger systems are used by larger groups of researchers. The second area for future work is in heterogeneous computing. The rapid advance of PC technology and the ease with which Beowulf systems may be reconfigured implies that these systems will not be static but evolve in time. A particular system will grow with newer technology being added to old. This is very different from the standard practice of discarding an old machine to be replaced by a new one. Beowulfs have the advantage of being able to continue to benefit from older components while be expanded with newer subsystems. The problem is that load balancing becomes more difficult. One would not want the performance of the system to be constrained by the slowest element. Rather, it is desirable to distribute the workload proportionally according to capability. The software infrastructure does not exist today

that guarantees the most effective use of heterogeneous Beowulfs. Addressing this problem will extend the lifetime of Beowulf components, further improving the amortized cost-performance and enabling larger configuration systems.

In conclusion, Beowulf-class computing offers an additional operating point in the space of alternatives for high end scientific and engineering computing. While not ideal for all applications and environments, in many cases it can significantly augment the computing capacity of an institution machine room and offer parallel computing in those many modest budget groups for which vendor offerings are either too costly or inappropriate. The rapidly increasing popularity of Beowulf class systems requires continued advances in resource management software to facilitate resource management and make these systems easier to use.

Bibliography

D. J. Becker, T. Sterling, D. Savarese, E. Dorband, U. A. Ranawake , and C. V. Packer; BEOWULF: A Parallel Workstation for Scientific Computation, Proceedings of the 1995 International Conference on Parallel Processing (ICPP), 11-14, 1995.
http://cesdis1.gsfc.nasa.gov/linuxweb/papers/ICPP95/final.html

Thomas Sterling, Donald J. Becker, Daniel Savarese, Michael R. Berry, and Chance Reschke, Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation, International Parallel Processing Symposium 1996.

P. Ballester, K. Banse, and M. Peron, "Data Flow System Specifications for Pipeline Quality Control," VLT-SPE-ESO-19600-1233 Issue 0.3, October 1996.
http://www.eso.org/dmd/dataproc/prepare/docs/qcpip/qc-pipe/qc-pipe.html

Chance Reschke, Thomas Sterling, Daniel Ridge, Daniel Savarese, Donald Becker and Phillip Merkey, A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation, in High Performance and Distributed Computing, 1996.

Michael S. Warren, John K. Salmon, Donald J. Becker, M. Patrick Goda, Thomas Sterling and Gregoire S. Winckelmans, Pentium Pro Inside: A Treecode at 430 Gigaflops on Price/Performance of $50 Mflop on Loki and Hyglac, Supercomputing '97. ACM Press. Gordon Bell Prize Finalist (to appear).
http://lokiwww.lanl.gov/papers/sc97/

Daniel Ridge, Donald Becker, Phillip Merkey, and Thomas Sterling, Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs Proceedings, IEEE Aerospace, 1997.
http://cesdis.gsfc.nasa.gov/beowulf/papers.html

Chance Reschke, Thomas Sterling, Daniel Ridge, Daniel Savarese, Donald Becker, and Phillip Merkey, A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation. Proceedings, High Performance and Distributed Computing, 1996.
http://cesdis.gsfc.nasa.gov/beowulf/papers.html

Donald J. Becker, Thomas Sterling, Daniel Savarese, Bruce Fryxell, and Kevin Olson Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation, Proceedings, High Performance and Distributed Computing, 1995.
http://cesdis.gsfc.nasa.gov/beowulf/papers.html

M. Beck, H. Bohme, M. Dziadzka, U. Kunitz, R. Magnus, D. Verworner, LINUX: Kernel Internals. Addison-Wesley Longman, 1996.

Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Second Edition. Bell Telephone Laboratories, Inc., 1988, 1978.

Tom Shanley, Mindshare, Inc, Pentium Pro Processor System Architecture. Addison-Wesley Developers Press, 1997.

William Gropp, Ewing Lusk, and Anthony Skjellum USING MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, 1994.

L.G.Valiant, A Bridging Model for Parallel Computation, Communications of the ACM. 103-111, Vol. 33, Issue 8, 1990.

Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker and Jack Dongarra, MPI: The Complete Reference. MIT Press, 1996.

Al Geist, Adam Beguelin, and Jack Dongarra, Pvm: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, 1994.

D. S. Katz, T. Cwik, B. H. Kwan, J. Z. Lou, P. L. Springer, T. L. Sterling, and P. Wang, An Assessment of a Beowulf System for a Wide Class of Analysis and Design Software, to be presented at the 4th NASA National Symposium on Large-Scale Analysis and Design on High-Performance Computers and Workstations, Williamsburg, Virginia, October 1997.

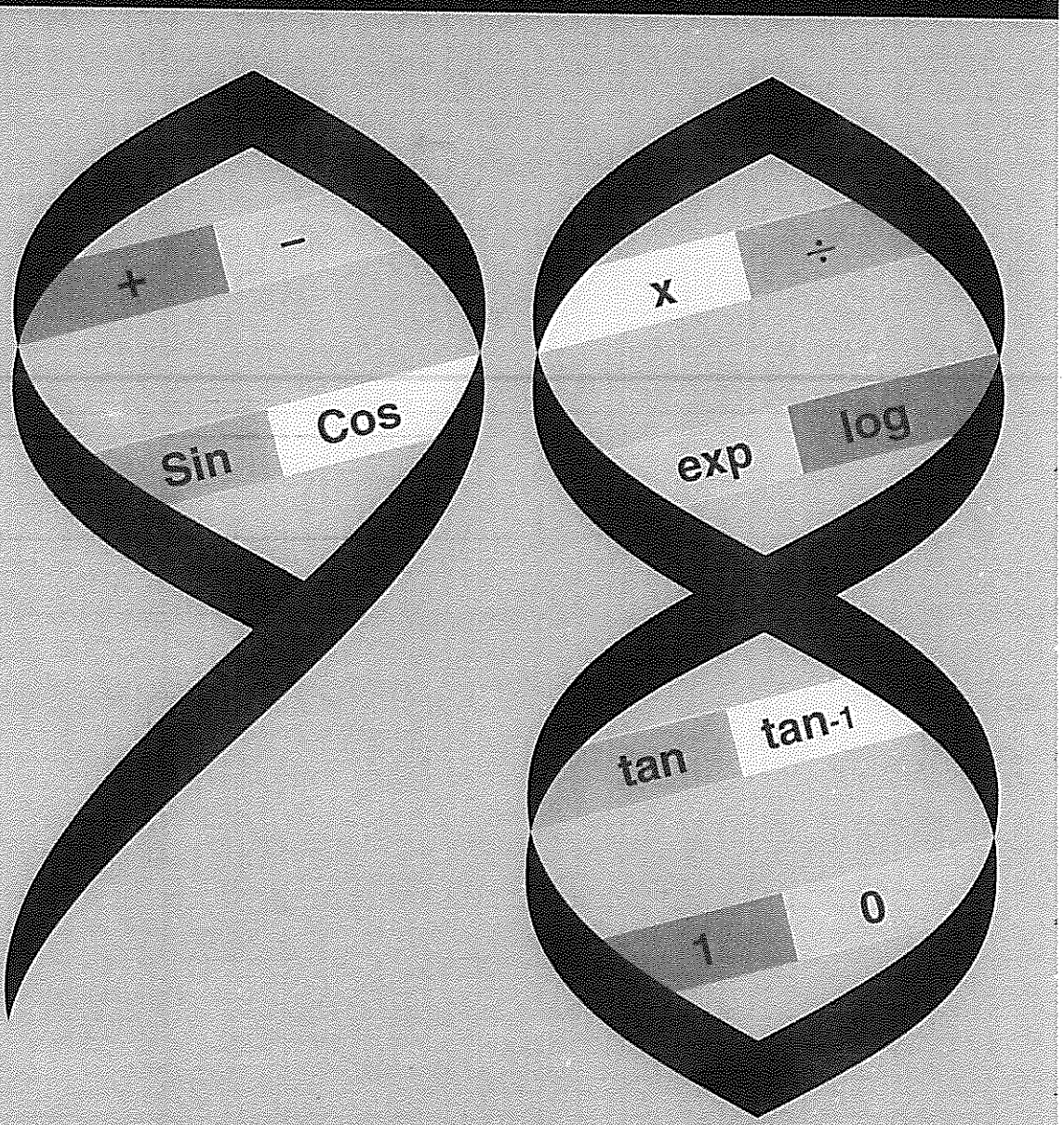
D. S. Katz, High-Perfomance Computational Electromagnetic Modeling Using Low-Cost Computers,

(abstract), IEEE AP-S Symposium/URSI Radio Science Meeting, Montreal, Canada, July 1997.

P. Wang, Massively Parallel Finite Volume Computation of Three-dimensional Thermal Convective Flows, to appear in the proceedings of 4th National Symposium on -Scale Analysis and Design on High-Performance Computers and Workstations, October 1997, Williamsburg, VA. (to appear in Advances in Engineering Software).

CONFERENCE PROCEEDINGS

Genetic Programming



edited by
John R. Koza
Wolfgang Banzhaf
Kumar Chellapilla
Kalyanmoy Deb
Marco Dorigo
David B. Fogel
Max H. Garzon
David E. Goldberg
Hitoshi Iba
Rick L. Riolo

Proceedings of the Third Annual Genetic Programming Conference

July 22–25, 1998
University of Wisconsin, Madison, Wisconsin