Beowulf
Introduction & Overview

In the summer of 1994 Thomas Sterling and Don Becker, working at CESDIS under the sponsorship of the ESS project, built a cluster computer consisting of 16 DX4 processors connected by channel bonded Ethernet. They called their machine Beowulf. The machine was an instant success and their idea of providing COTS (Commodity off the shelf) base systems to satisfy specific computational requirements quickly spread through NASA and into the academic and research communities. The development effort for this first machine quickly grew into a what we now call the Beowulf Project. Some of the major accomplishment of the Beowulf Project will be chronicled below, but a non-technical measure of success is the observation that researcher within the High Performance Computer community are now referring to such machines as "Beowulf Class Cluster Computers." That is, Beowulf clusters are now recognized as genre within the HPC community.

The next few paragraphs will provide a brief history of the Beowulf Project and discussion of certain aspects or characteristics of the Project that appear to be key to its success.

The Center of Excellence in Space Data and Information Sciences (CESDIS) is a division of the University Space Research Association (USRA) located at the Goddard Space Flight Center in Greenbelt Maryland. CESDIS is a NASA contractor, supported in part by the Earth and space sciences (ESS) project. The ESS project is a research project within the High Performance Computing and Communications (HPCC) program. One of the goals of the ESS project is to determine the applicability of massively parallel computers to the problems faced by the Earth and space sciences community. The first Beowulf was built to address problems associated with the large data sets that are often involved in ESS applications.

It may well be that this is simply the "right time" in history for the development of Beowulf class computers. In the last ten years we have experienced a number of events that have brought together all the pieces for the genesis of the Beowulf project. The creation of the causal computing market (office automation, home computing, games and entertainment) now provides system designer with a new type of cost effective components. The COTS industry now provides fully assembled subsystems (microprocessors, motherboards, disks and network interface cards). Mass market competition has driven the prices down and reliability up for these subsystems. The development of publicly available software, in particular, the Linux operating system, the GNU compilers and programming tools and the MPI and PVM message passing libraries, provide hardware independent software. Programs like the HPCC program have produced many years of experience working with parallel algorithms. That experience has taught us that obtaining high performance, even from vendor provided, parallel platforms is hard work and requires researchers to adopt a do-it-yourself attitude. A second aspect to this history of working with parallel platforms is an increased reliance on computational science and therefore an increased need for high performance computing. One could argue that the combination of the these conditions: hardware, software, experience and expectation, provided the environment that makes the development of Beowulf clusters seem like a natural evolutionary event.

We are constantly reminded of the performance improvements in microprocessors, but perhaps more important to the Beowulf project is the recent cost/performance gains in network technology. The history

of MIMD computing records many academic groups and commerical vendors that have built multiprocessor machines based on what was then the state-of-art microprocessor, but they always required special "glue" chips or a one-of-a-kind interconnection schemes. For the academic community this lead to interesting research and the exploration of new ideas, but usually resulted in one of a kind machines. The life cycle of such a machines strongly correlates to the life cycle of the graduate careers of those working on them. Vendors usually made choices for special features or interconnection schemes to enhance certain characteristics of their machine or to tailor a machine to a perceive market. Exploiting these enhancements required programmers to adopt a vendor specific programming model. This often lead to dead ends with respect to software development. The cost effectiveness and Linux support for high performance networks for PC class machines has enabled the construction of balanced systems built entirely of COTS technology which has made generic architectures and programming model practical.

The first Beowulf was built with DX4 processors and 10Mbit/s Ethernet. The processors were too fast for a single Ethernet and Ethernet switches were still too expensive. To balance the system Don Becker rewrote his Ethernet drivers for Linux and built a "channel bonded" Ethernet where the network traffic was striped across two or more Ethernets. As 100Mbit/s Ethernet and 100Mbit/s Ethernet switches have become cost effective, the need for channel bonding has diminished (at least for now). In late 1997, a good choice for a balance system was 16, 200MHz P6 processors connected by Fast Ethernet and a Fast Ethernet switch. The exact network configuration of a balanced cluster will continue to change and will remain dependent on the size of the cluster and the relationship between processor speed and network bandwidth and the current price list for each of components. An important characteristic of Beowulf clusters is that these sorts of changes---processors type and speed, network technology, relative costs of components---do not change the programming model. Therefore, users of these systems can expect to enjoy more forward compatibility then we have experienced in the past.

Another key component to forward compatibility is the system software used on Beowulf. With the maturity and robustness of Linux, GNU software and the "standardization" of message passing via PVM and MPI, programmers now have a guarantee that the programs they write will run on future Beowulf clusters---regardless of who makes the processors or the networks. A natural consequence of coupling the system software with vendor hardware is that the system software must be developed and refined only slightly ahead of the application software. The historical criticism that system software for high performance computers is always inadequate is actually unfair to those developing it. In most cases coupling vendor software and hardware forces the system software to be perpetually immature. The model used for Beowulf system software can break that rule.

The first Beowulf was built to address a particular computational requirement of the ESS community. It was built by and for researcher with parallel programming experience. Many of these researchers have spent years fighting with MPP vendors, and system administrators over detailed performance information and struggling with underdeveloped tools and new programming models. This lead to a "do-it-yourself" attitude. Another reality they faced was that access to a large machine often meant access to a tiny fraction of the resources of the machine shared among many users. For these users, building a cluster that they can completely control and fully utilize results in a more effective, higher performance, computing platform. An attitude that summarizes this situation is "Why buy what you can afford to make?" The realization is that learning to built and run a Beowulf cluster is an investment; learning the peculiarities of a specific vendor only enslaves you to that vendor.

These hard core parallel programmers are first and foremost interested in high performance computing applied to difficult problems. At Supercomputing '96 both NASA and DOE demonstrated clusters costing less than $50,000 that achieved greater than a gigaflop/s sustained performance. A year later,

NASA researchers at Goddard Space Flight Center combined two clusters for a total of 199, P6 processors and ran a PVM version of a PPM (Piece-wise Parabolic Method) code at a sustain rate of 10.1 Gflop/s. In the same week (in fact, on the floor of Supercomputing '97) Caltech's 140 node cluster ran an N-body problem at a rate of 10.9 Gflop/s. This does not mean that Beowulf clusters are supercomputers, it just means one can build a Beowulf that is big enough to attract the interest of supercomputer users.

Beyond the seasoned parallel programmer, Beowulf clusters have been built and used by programmer with little or no parallel programming experience. In fact, Beowulf clusters provide universities, often with limited resources, an excellent platform to teach parallel programming courses and provide cost effective computing to their computational scientists as well. The startup cost in a university situation is minimal for the usual reasons: most students interested in such a project are likely to be running Linux on their own computers, setting up a lab and learning of write parallel programs is part of the learn experience.

In the taxomony of parallel computers, Beowulf clusters fall somewhere between MPP (Massively Parallel Processors, like the nCube, CM5, Convex SPP, Cray T3D, Cray T3E, etc.) and NOWs (Networks of Workstations). The Beowulf project benefits from developments in both these classes of architecture. MPPs are typically larger and have a lower latency interconnect network than an Beowulf cluster. Programmers are still required to worry about locality, load balancing, granularity, and communication overheads in order to obtain the best performance. Even on shared memory machines, many programmers develop their programs in a message passing style. Programs that do not require fine-grain computation and communication can usually be ported and run effectively on Beowulf clusters. Programming a NOW is usually an attempt to harvest unused cycles on an already installed base of workstations in a lab or on a campus. Programming in this environment requires algorithms that are extremely tolerant of load balancing problems and large communication latency. Any program that runs on a NOW will run at least as well on a cluster.

A Beowulf class cluster computer is distinguished from a Network of Workstations by several subtle but significant characteristics. First, the nodes in the cluster are dedicated to the cluster. This helps ease load balancing problems, because the performance of individual nodes are not subject to external factors. Also, since the interconnection network is isolated from the external network, the network load is determined only by the application being run on the cluster. This eases the problems associated with unpredictable latency in NOWs. All the nodes in the cluster are within the administrative jurisdiction of the cluster. For examples, the interconnection network for the cluster is not visible from the outside world so the only authentication needed between processors is for system integrity. On a NOW, one must be concerned about network security. Another example is the Beowulf software that provides a global process ID. This enables a mechanism for a process on one node to send signals to a process on another node of the system, all within the user domain. This is not allowed on a NOW. Finally, operating system parameters can be tuned to improve performance. For example, a workstation should be tuned to provide the best interactive feel (instantaneous responses, short buffers, etc), but in cluster the nodes can be tuned to provide better throughput for coarser-grain jobs because they are not interacting directly with users.

The Beowulf Project grew from the first Beowulf machine and likewise the Beowulf community has grown from the NASA project. Like the Linux community, the Beowulf community is a loosely organized confederation of researcher and developer. Each organization has its own agenda and its own set of reason for developing a particular component or aspect of the Beowulf system. As a result, Beowulf class cluster computers range from several node clusters to several hundred node clusters. Some systems have been built by computational scientists and are used in an operational setting, others

have been built as test-beds for system research and others are serve as an inexpensive platform to learn about parallel programming.

Most people in the Beowulf community are independent, do-it-yourself'er. Since everyone is doing their own thing, the notion of having a central control within the Beowulf community just doesn't make sense. The community is held together by the willingness of its members to share ideas and discuss successes and failures in their development efforts. The mechanisms that facilitate this interaction are the Beowulf mailing lists, individual web pages and the occasional meeting or workshop.

The future of the Beowulf project will be determined collectively by the individual organizations contributing to the Beowulf project and by the future of mass-market COTS. As microprocessor technology continues to evolve and higher speed networks become cost effective and as more application developers move to parallel platforms, the Beowulf project will evolve to fill its niche.

Contact: Phil Merkey merk@cesdis.gsfc.nasa.gov
Page last modified: 1998/09/11 20:20:22 GMT
CESDIS is operated for NASA by the USRA