A Coming of Age for Beowulf-Class Computing

Thomas Sterling and Daniel Savarese

Center for Advanced Computing Research
California Institute of Technology
1200 E. California Blvd. MC 158-79
Pasadena, CA 91125 USA
tron@cacr.caltech.edu, dfs@cacr.caltech.edu

1 Introduction

Beowulf-class systems along with other forms of PC clustered systems have matured to the point that they are becoming the strategy of choice for some areas of high performance applications. A Beowulf system is a cluster of mass market COTS personal computers interconnected by means of widely available local area network (LAN) technology. Beowulf software is based on open source code Unix-like operating systems that, in a majority of cases, is Linux. The API for Beowulf is based on message passing semantics and mechanisms including explicit models such as PVM and MPI or implicit models such as BSP of HPF. Since its introduction in 1994, Beowulf-class computing has gone through five generations of PCs from multiple microprocessor vendors including the Intel x86 family, DEC's Alpha, and the PowerPC from IBM and Motorola. Originally, Beowulfs were implemented as small clusters in the range of 4 to 32 nodes. Larger clusters of 48 to 96 processors were deployed two and a half years ago. Today there are many systems of 100 to 300 processors with systems of over a thousand processors in the planning stage for implementation over the next year.

The earliest Beowulf systems had a peak floating point performance of approximately 320 Mflops sustaining approximately 70 Mflops on floating point intensive problems. By the third generation, peak performance systems of 3.2 Gflops were delivering sustained performance of 1.2 Gflops on non-trivial real world applications. A year later, larger systems were reported sustaining performance in excess of 10 Gflops. Shortly thereafter, the first Beowulfs were credited with being among the 500 largest computers in the world. In the last two years, Beowulf-class systems were recognized as providing the best price-performance of general-purpose high-end systems by winning the Gordon Bell Prize for supercomputer price-performance two years in a row. From a few experimental systems four years ago, many hundreds of systems have been deployed across the nation and around the world with as many as a thousand such systems operational today.

From this experience and level of acceptance, it is clear that Beowulf-class systems are having a significant impact on the field of high-end computing. But

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 78–88, 1999.

the value of this technology extends beyond simply the numbers of systems deployed. These systems are enabling applications work that might not have been conducted otherwise or would have been done at a substantially lower level. Also they are providing an exceptional vehicle for educational programs in parallel computing hardware, software, and parallel programming techniques, even down to the high school level. Many people are entering the field of parallel computing through the opportunity door opened by the availability of Beowulf systems. The roles and range of this class of systems are proving highly diverse; more than was originally imagined by those conducting the earliest path-finding work in this area.

Beowulf-class systems can no longer be ignored as inconsequential and relegated to the niche of novelty and hobbyist by their detractors. But nor can they be represented as the replacement for all other high-end system types for all computing environments and purposes by their advocates. The controversy surrounding Beowulf-class systems is important because it will determine future directions for advanced development and applied research in a number of related fields including system software, computational techniques, and system area network technology. The results of such work will determine the applicability and utility of this alternative approach to scalable computer systems implementation. This paper addresses the important issues that are at the center of the controversy and that will determine future directions and acceptance of Beowulf-class systems to satisfy near term computing requirements in the medium to high end of the performance spectrum. An attempt is made to clarify three critical questions: applicability, advantages/disadvantages, and economics. Most of these issues are subject to varied interpretation based on the underlying assumptions. The biggest problem with the continued contentions is that the arguments are driven by different expectations and tolerances. In essence, they are an example of attempting to compare apple and oranges. It is the intent of this paper to resolve, at least in part, the existing confusion.

2 Advantages: Impact Beyond Dollars

The selection of a Beowulf over a conventional vendor-offered parallel system is rarely driven simply by a desire to pay a lower price for a specified performance level. Should this be the case, the level of computing capability acquired would remain constant while freed dollars from the system procurement budget would be reallocated to some other part of an organization's needs. But an important aspect of the qualitative advantage of Beowulfs stems from their dramatically lower cost, not to save money but rather to acquire much more capability for a limited budget. While at first this may appear to be a distinction without a difference, its impact is substantial. In the first case, nothing changed other than the saving of some money. In this case, an approximate price-performance advantage of an order of magnitude will deliver a Beowulf-class system of more than ten times the peak performance of a vendor supplied system. In science and technology, an order of magnitude is an important differentiator. In the transportation

industry, society supports two entirely separate infrastructures (actually three) because of the times ten difference in speed between the automobile with its interstate highway system and parking garages and the commercial jet aircraft with its FAA traffic control and airports. (railroads are the third independent infrastructure)

On a per node peak floating point performance basis, comparisons between comparable generation systems of equal number of processors and memory capacity show that tightly coupled vendor provided distributed shared memory systems cost between 8 and 12 times that of a Beowulf-class system. This does not include vendor supplied clusters of workstations where the price differential may be only about a factor of 4 (even less in specific cases), still in favor of the Beowulfs. For certain classes of applications, this can translate into a performance advantage of an order of magnitude. Problems that might have taken a full workday can be performed in less than an hour. Or a problem can be performed in a single user's day that would have taken two user work weeks (here a user day is defined as 8 hours, a user week is 40 hours) Such a difference qualitatively alters the way science and engineering can be performed and the impact that computation can have on it.

Perhaps more importantly are those problems that can be addressed due to the possibility of a Beowulf implementation that would not have been attacked by parallel processing at all. The entry level cost of vendor supplied parallel systems is often too high for some environments, principally those in the pure sciences, the social sciences, and many educational institutions. Under these circumstances before Beowulf, these problems were relegated to desktop workstations because the next higher cost system, a small SMP was much too high. Beowulfs gave low cost scalability and a graceful way for low budgeted computations to benefit from higher performance systems. But its not just shorter execution time that yields the qualitative advantage. Also, problem size, resolution, or fidelity from the modeling of additional phenomenology can enhance the quality of the science being performed, yielding new insights, otherwise unattainable under restricted budgets.

User's have greater control over their Beowulfs systems than they do with those provided by vendors while being less vulnerable to changes in vendor marketing plans. High performance computing through the late 80's and early 90's suffered from rapid changes in available system architectures. Many vendors went out of business with customers stuck with their rapidly obsolescent and unsupported orphan systems. Dramatic examples include the Intel Paragon, the TMC CM-5, the Maspar MP-2, and the KSR-1. Even when the company does not meet its demise, radical change in architecture can leave long term customers with no viable migration path. TMC switched from its SIMD class CM-2 to the MIMD class CM-5, Convex went from its vector C-4 to the DSM SP-1000. CRI is abandoning its vector family and its T3 family as well after multiple versions of both. In marked contrast, Beowulfs are forever. As long as there are low cost

desk-top and server PC based systems, and third party LAN or SAN network technology, users will always be able to acquire and assemble the necessary subsystems to create future generation Beowulfs.

Ironically, not only can the specific component types change over time, Beowulf customers benefit from this technology dynamic. Again ironically, the most rapid advances in processor, memory, and networking technologies first impact the lowest end systems; those used within Beowulf clusters. The reasons are first that the biggest market opportunity is in the mass market arena (i.e. PCs) so the new devices are first incorporated in them, and second that large MPP class systems require considerable development time to integrate the new devices prior to delivering the completed parallel systems to their customers. In contrast, as soon as a new processor is packaged on a PC motherboard, it is ready to be installed in a Beowulf 24 hours later. Frequently, new Beowulfs have the hottest microprocessors available. This trend is likely to continue with the near term release of the Compaq DEC EV-6 and the AMD K-7, both of which will deliver more than 1 Gflops peak performance. However, more than processor performance, Beowulfs require advances in communication interconnect technology. For a majority of Beowulf-class systems, Fast-Ethernet at 100 Mbps has been tolerable and cost less than 25% of the total system price. For those applications sensitive to network characteristics, Myrianet has provided order of magnitude advantage in bandwidth and latency, but at an effective cost of about 50% of the total system price. At the same time, large system switches at acceptable per port price are enabling multi-hundred node configurations to be assembled. New techniques based on zero-copy methods such as VIA will greatly reduce software overhead in the next year. The new PCI bus will reduce the effect of this interface as a bottleneck. These trends in communications will match the advances in processor performance to retain if not improve the relative balance of overall system properties.

The flexibility permits custom system configurations on site by users. This just-in-place configuration capability allows structures of PCs to be organized such that the resulting system topology conforms to the data flow paths of the anticipated workload. Thus, Beowulf-class technology provides the means to establish semi-custom systems to meet particular requirements. Furthermore, such configurations can be altered at the site by local personnel to adapt to changing requirements or to incorporate technology upgrades. This evolvability provides a dynamic capability that extends lifetime and utility of Beowulfs and optimizes their applicability to user-specific workloads.

Beowulf-class systems leverage the community investment in open source software including Linux and the Gnu tools. Many experts around the world have contributed to this large and robust sophisticated software environment. Equally important, many users have provided a continued stream of feedback that has driven the maturing and debugging of the package. The result is that a very low cost highly capable environment is being employed by Beowulf. Although

a small detail, the cost of Linux for a cluster system does not increase with the number of nodes comprising the system as would be the case with at least one widely distributed commercial operating system. The use of PVM and MPI have provided a means of creating highly portable applications software that will also run on many vendor parallel platforms. Indeed, a role of Beowulfs has been as development platforms in preparation to run on larger commercial systems. Amusingly, in some cases at least, the computational scientists have never bothered to make the transition to the larger, often contended, commercial systems.

3 Disadvantages: Limitations in Application and Usability

Beowulf-class systems are the third wave to performing high speed computations. The first wave is very high speed processors or a few such processors such as would be found in a CRI vector computer or an IBM mainframe. The second wave is the massively parallel systems incorporating tightly coupled microprocessors with custom high bi-section bandwidth internal networks and low overhead hardware mechanisms for at least some aspects of resource management (such as cache coherence). Beowulf-class systems do not fall into either of these categories and where applications demand the particular strengths of these first two genre of high-end computers, Beowulfs often may not perform well. Beowulfs, along with clusters of workstations, and other forms of loosely coupled systems provide a third path to high end computing that delivers very high price-performance for those problems that do not require (and should not pay for) the tightly coupled support of the first two system families. A disadvantage of Beowulf-class systems is that they are not as general as some vendor offerings.

The software environments currently available on most Beowulfs are primitive compared to the environments provided by vendors. While the basic node operating system, usually Linux, is very powerful and complete, and the equal of any commercial node operating system (perhaps better in some cases), the set of tools provided to manage the ensemble of nodes is usually superior for the vendor supplied systems. For some user environments, this is very important. While advances are being made and experimental tools are being tested at a number of organizations, a common framework has as yet not been adopted by the community as a whole. New users of Beowulf-class systems will not find readily available the wide array of useful services that exist, for example, on an SGI SMP.

The customer-engineer approach to maintenance of vendor supplied hardware and software at the customer site is not available to Beowulf users. The turnkey hands-off support that is provided with, for example, the IBM SP-2 is not the normal service that Beowulf sites can anticipate. When something goes wrong, there isn't a single one-stop-shopping 1-800 number that the user can call

to solve the problem. There are many environments where this is not acceptable. As one colleague, somewhat exasperated by what he called the Beowulf Hype Factory put it: there is a price on low-cost. Beowulf is not suitable for all computing environments. Only those organizations with some in-house expertise, like systems administrators, can expect to employ Beowulfs effectively.

Floor space and packaging is an unfortunate problem with Beowulf-class systems. The two approaches to packaging Beowulfs is to develop some custom approach which can package motherboards tightly or to use the mass market per node tower boxes with power supplies that are incredibly inexpensive and robust. Unfortunately, the low cost attractiveness of the mass market packaging takes up a premium in floor space. Even with strong/tall utility shelves, the space required can be as much as twice that of custom packaging. For some machine rooms, such waste is unacceptable, especially as Beowulfs expand in scale through a number of hundreds of nodes. While custom packaging experiences have demonstrated significant improvements for space utilization, the cost of such packaging could be as much as half the cost of the system.

4 Comparative Economics

Controversy surrounds the economics of Beowulf-class systems. Users and general proponents simply quantify the costs of the component ensembles, sometimes with necessary physical packaging. Detractors identify a number of support services expected of conventional vendor provided systems that are not directly available to Beowulf users. The contention is that the cost of providing such support would eliminate or at least severely narrow the putative cost gap. There is merit to both arguments but it is based on a contradiction of assumptions. Here, the principle issues are examined that determine the real economics of Beowulf clusters deployment.

4.1 Hardware Costs

In fact, it is hard to say exactly what the difference in hardware costs is between Beowulfs and vendor systems. For Beowulfs to a first order, it can be considered the combined purchase costs of the subsystems comprising the total system ensemble. But this will vary among systems, even of equivalent form, dependent on negotiated purchase price between procuring institute and supplying distributor. It will also vary with time, even with short spans, as the market moves aggressively forward in the presence of constant change in product offerings and in the context of mass market competitive pricing.

It is harder to determine the actual cost of the vendor offerings. The vendors correctly assert that their hardware prices include hidden support services and delivered base level software. However, there are also additional costs for annual

maintenance contracts and unbundled software. Finally, the list prices are often substantially higher than the final negotiated price based on various discounts that are applied to win the sale.

The third issue is that the comparison of hardware costs relates systems that are not actually equivalent. Often the processors are not the same. And almost always, the Beowulf networks are lower in bandwidth and higher in latency than the more tightly coupled vendor systems. Beowulf comparisons based on peak performance to purchase price are therefore a distortion. Routinely factors of 10 to 40 are reported on a per node basis of peak performance to procurement cost. It is not to be assumed that all Beowulf nodes are necessarily lower in performance than vendor provided systems. Some recent Beowulfs are employing Compaq DEC Alpha microprocessors packaged for the lower-end market. These have the highest peak performance of any microprocessors. The EV-6 and AMD K7 are likely to continue the availability of high performance micros to Beowulf ensembles. But the networks are routinely of lower capability. It would be interesting to build a comparative table based on bi-section bandwidth versus system cost and see if the order of magnitude difference between vendor systems and Beowulf systems for comparable nodes would be retained; probably not. While as not severe a gap, memory capacities are also not always equivalent. Finally, I/O bandwidth is another area in which some vendor systems significantly exceed comparable Beowulf systems.

The first conclusion must be that just from a hardware perspective, comparisons are difficult to make. The systems are rarely comparable against various metrics even if the number of nodes between the two is the same. The second conclusion is that as always, the real cost advantage is on a per application basis. This varies dramatically in a number of dimensions. Memory capacity requirements can vary by three orders of magnitude across applications using the same level of performance. Inter-node communication bandwidth and latency requirements can also vary by three orders of magnitude or more across applications for the same scale of system. Some applications are embarrassingly parallel with essentially no data sharing and intra-program synchronization points. Some have restricted synchronization and partitioned data and task sets that for sufficient granularity can permit efficient execution in spite of limited communications framework. These two classes of applications demonstrate excellent price-performance, even when the communication costs are much higher for Beowulfs than some commercial machines. The communication demands if small enough allow Beowulfs to operate with excellent price-performance even with relatively weak system area network hardware and sometimes superior raw performance. The third conclusion is that there are some highly sequential applications that will not yield significantly better price-performance for Beowulfs than vendor systems, and sometimes worse, due to the high communications demands.

4.2 System Management Costs

The cost of Beowulf systems does not reflect the cost of system support and maintenance. This is true. But the impact of this varies dramatically depending on the environment in which a new Beowulf is deployed. At a new site having no support for PCs or other computing facilities, additional full time support personnel have to be hired to manage the software and perform routine hardware maintenance. Under these circumstances, the cost should be attributed to that of the Beowulf. But in many other instances, Beowulfs are deployed in environments that are MPP/Unix/PC capable. For example, any site that already manages a farm of PCs already has in place the necessary talent to manage the hardware resources of a Beowulf including the standard maintenance. It is a point of confusion that there is no vendor maintenance support. All of the hardware components come with warrantees. Beowulf users do the same thing with broken Beowulf nodes that any owner of a PC does; ship it back to the supplier. The same is true with the hardware maintenance of the system area networks. The software management costs must also be considered. At many Beowulf sites, there is already a strong presence of Unix operating system workstations and even high-end computers with Unix-like software environments. With the proliferation of Linux, many more sites already have staff who have direct familiarity with Linux installation procedures. Beowulf-system operation under these circumstances leverages these existing talent resources. Often, the user and the systems administration are the same person. The additional overhead in doing it this way is relatively small after an initial installation period of a few days.

The primary discrepancy in viewpoint about the economics of Beowulf clusters is the expectation of system services. Beowulfs do not require the machine room mentality typical of vendor provided large systems because they are often used by one or a few people. The complex management infrastructure required of an expensive heavily shared MPP is simply unnecessary for many Beowulf installations. Because of their very low cost, they can be owned by a small organization and treated as a local resource with informal management; no accounting, no job control, not even automated partitioning. In some cases, Beowulfs are organized as single application systems. These run the same program continuously on newly supplied data sets; the application is the system. Some applications run for days or weeks without interruption. Again, due to their low cost, they can be dedicated to single applications of this sort.

Then it is clear that both sides of the debate about the economics of clusters versus vendor offerings are valid depending on the expectations and underlying assumptions. What has to be understood by the conventional computing community is that Beowulfs do open up an alternative approach for accomplishing some of the important computation that had been performed previously solely on commercial systems. And that furthermore, there is additional work being done that would not have been done at all. Cost advantage depends on the nature of the problem and the environment in which it is being performed.

5 Trading Peak Performance for Total Work

Beowulf-class systems can be dedicated to the execution of a single application for a long period of time because of their low cost. This opens a new important approach to accomplishing large scale computation at exceptional cost. But it also puts some additional responsibilities on the support software.

Historically, large applications required large systems with high peak performance. Such systems are routinely expensive and shared among many users. At any one time, a user is allocated some part of the machine (assuming a multi-head system that can be simultaneously divided among multiple users) for a limited amount of time. At some later time, the same user may be allocated the next installment of time and resources for the same problem. The average amount of work accomplished by this shared system on behalf of the user's application is the product of the allotted resources and time divided by the duty cycle (how frequently the user gets on the machine). This can often be surprisingly low while the cost of these premium resources can be very high.

It is a myth that large problems demand large systems. It is true that some classes of algorithms require tightly coupled systems with high bi-section bandwidth as discussed earlier. But for problems that are latency tolerant, a time space trade-off is possible. Large problems demand large work where work is the time-space product. A few processors working for a long time can accomplish the same amount of work as a large number of processors working for a short time. Beowulf-class systems provide an acceptable medium for achieving this trade-off between time and resources. A moderate scale Beowulf of 32 Compaq DEC Alpha EV-6 processors in one week would perform the same work as a thousand processor T3E in a day (actually a little less than a day) with a cost difference of approximately a factor of a hundred.

But the additional problem of memory capacity of the two systems implies that the T3E would store far more data during the execution. And for some problems, this is particularly important. However, studies have shown a wide variance in memory requirements for applications of equivalent performance. While some problems require memory in bytes comparable to performance in flops (a 100 Gflops computation requires 100 Gbyte memory), the majority of science and engineering codes require less than one percent of this capacity. For these problems, the 32 EV-6 machine would have sufficient main memory to support the storage of the required data set. But for those problems that demand the larger memory capacities, a second opportunity exists. Because these applications are constructed using latency tolerant methods, the use of out-of-core computing techniques can address them memory discrepancy challenge. Out-of-core techniques allow secondary storage to replace part of main memory, using memory as a kind of cache and windowing across the larger data set held on disk. For some problems and algorithm types, swapping data between disk and memory can be done effectively in spite of the long delays to disk. As a result,

we find that large scale science and engineering computations can be enabled by moderate scale low cost Beowulfs.

One problem does occur as the duration of computation is extended. Although reliability for Beowulf class systems is very high, nonetheless, longer computations incur increased probability that they will fail to complete correctly. The common solution for this is checkpoint-restarting. Other reasons for checkpointing are important beyond the possibility of a hardware failure. Sometimes even a Beowulf needs to be shared among a couple of users. It is nice to be able to migrate a running program to a modified configuration. The means is the same as that for checkpointing; save the data set and have the means for reallocating the intermediate data and control-state to a different configuration. Currently, all checkpointing is done by the applications programmers with little help from the system software support tools. Over time this will need to change.

6 An Agenda for Beowulf

Future Beowulf implementations will require a richer set of software tools to improve usability, portability, reliability, and reconfigurability. Some of the key requirements are discussed below. In each case, there are example and experimental tools that have been implemented at various institutions. Where appropriate, these need to be generalized, packaged, and distributed in a way that will make them generally useful to the Beowulf community.

6.1 Resource Discovery

Beowulf systems currently lack a standard means of determining the identities of their constituent elements and possible sub-elements for the purposes of executing tasks, reconfiguring the system, and supporting higher level functions such as system monitoring and scheduling. A telling example of why such a mechanism is needed is the way current system software is forced to provide this functionality for itself. Parallel programming libraries such as PVM and MPI require the definition of host files that are located in different parts of the file system. Schedulers such as PBS require similar custom support. Beyond not being able to readily identifying the nodes in a system, it is not possible for arbitrary processes to determine if a user may gain access to a given subset of the system or what system resources are available. Basic functionality, such as determining what nodes are part of the system, what nodes a user can use, and what sub-partitions of the system exist can provide a common foundation for general system software for Beowulf systems.

6.2 Resource Allocation/Assignment

Organizations that rely on commodity clustered computing as the backbone of their computational science research programs often share Beowulf clusters between multiple research groups. They require the ability to dynamically reconfigure access to portions of a cluster based on user requirements. Batch scheduling

and job queues do not adequately meet the resource management needs of these organizations. They want to be able to arbitrarily allocate sets of resources to sets of users for periods of time, not only to run applications, but to perform development work and perform general experimentation. Current generation Beowulf system software does not meet these requirements. The ability to create arbitrary named groups and subgroups of resources, perhaps managed through an LDAP server, with associated property lists will add a new level of flexibility in the management of Beowulf clusters and provide underlying infrastructure for system software such as schedulers.

6.3 System Status Monitoring

Closely tied to both resource allocation and discovery is system status monitoring. A function as simple as tracking what nodes are usable and unusable in a cluster is integral to the aforementioned tasks. Although many monitoring tools have been independently developed, it is not currently possible to export the information provided by these tools in a generally usable fashion to other programs that may wish to make use of such data. Either a common set of system information gathering APIs or a common protocol for speaking to monitoring daemons will eventually have to emerge. The purpose of such an approach is twofold. The first is to allow future system software to leverage the valuable information provided by system monitors. And the second is to allow the interchangeability of system monitoring software in the same way it is possible to replace a mail or ftp server from one vendor with that of another.

6.4 Remote Dynamic Process Instantiation

Beowulf clusters can only perform useful work by executing processes on their component nodes. This essential function is currently served by one of the most inefficient possible mechanisms: rshell. A fundamental problem for large Beowulf clusters is that rshell does not scale well due to its use of reserved ports. The number of processes that can be started using rshell from a single node is limited to the number of reserved ports available to the command, which in the best case is 512. Not only does rshell not scale to larger systems, but it requires the persistent establishment of one or two TCP connections depending on whether or not standard error is required. Most PVM and MPI programs do not require this and would prefer to instantiate processes on a remote node in a fire and forget manner. In addition, some system software is more logically designed using something akin to an rfork() mechanism. We are already seeing signs of improvement in this area as the Beowulf community begins to experiment with alternative remote execution approaches, including an rfork(), currently under development at NASA Goddard.