An Assessment of Beowulf-class Computing for NASA Requirements: Initial Findings from the First NASA Workshop on Beowulf-class Clustered Computing

Thomas Sterling
NASA Jet Propulsion Laboratory
California Institute of Technology
tron@cacar.caltech.edu

Tom Cwik
NASA Jet Propulsion Laboratory
cwik@jpl.nasa.gov

Don Becker
Center of Excellence for Space Data and Information Science
Goddard Space Flight Center
becker@cesdis1.gsfc.nasa.gov

John Salmon
California Institute of Technology
johns@cacr.caltech.edu

Mike Warren
Los Alamos National Laboratory
msw@cacr.caltech.edu

Bill Nitzberg
NASA Ames Research Center
nitzberg@nas.nasa.gov

Abstract—The Beowulf class of parallel computing machine started as a small research project at NASA Goddard Space Flight Center's Center of Excellence in Space Data and Information Sciences (CESDIS). From that work evolved a new class of scalable machine comprised of mass market common off-the-shelf components (M^2COTS) using a freely available operating system and industry-standard software packages. A Beowulf-class system provides extraordinary benefits in price-performance. Beowulf-class systems are in place and doing real work at several NASA research centers, are supporting NASA-funded academic research, and operating at DOE and NIH. The NASA user community conducted an intense two-day workshop in Pasadena, California on October 22-23, 1997. This first workshop on Beowulf-class systems consisted primarily of technical discussions to establish the scope of opportunities, challenges, current research activities, and directions for NASA computing employing Beowulf-class systems. The technical discussions ranged from application research to programming methodologies. This paper provides an overview of the findings and conclusions of the workshop. The workshop determined that Beowulf-class systems can deliver multi-Gflops performance at unprecedented price-performance but that software environments were not fully functional or robust, especially for larger "dreadnought"-scale systems. It is recommended that the Beowulf community engage in an activity to integrate, port, or develop, where appropriate, necessary components of the software infrastructure to fully realize the potential of Beowulf-class computing to meet NASA and other agency computing requirements.

TABLE of CONTENTS

1. INTRODUCTION

NASA is relying on a policy of "better, faster, cheaper" to accomplish more of its mission in shorter time with reduced budget. Many aspects of the broad and growing NASA mission rely on the availability of advanced computing capability including science, engineering, data assimilation, spaceborne flight and instrumentation control, data archiving and dissemination, signal processing, and simulation. High-end computing in particular has been a significant challenge due to a number of factors which have limited their availability. These include high cost, decreasing number of vendors, variability of architecture types across vendors and between successive generations of a given vendor, and often inadequate software environments. An alternative approach to achieving medium to high-end computing is the direct application of mass market commodity off-the-shelf (M^2COTS) PC technology. Recent advances in capability and price have contributed to their emergence as a viable path to scalable computing systems for scientific and engineering applications. Beowulf is an early NASA project to explore and develop this potential opportunity for NASA mission computing requirements. Beowulf-class systems are being employed at many NASA centers for diverse uses including Earth and space sciences, computational aerosciences, and computer science investigations. The first NASA workshop on Beowulf-class clustered computing was held on October 22 and 23, 1997 in Pasadena, CA sponsored by the NASA Ames Research Center and hosted by the Jet Propulsion Laboratory. This paper presents an early assessment of Beowulf-class computing systems based on the findings of that workshop and reports of various studies presented there.

2. A REVOLUTION in PRICE-PERFORMANCE

In September, 1996 at the DOE Los Alamos National Laboratory and the NASA Jet Propulsion Laboratory in collaboration with Caltech, two medium-scale parallel M^{2}COTS systems were installed. By October both sites had performed a complex *N*-body gravitational simulation of 2 million particles using an advanced tree-code algorithm. Each of these systems cost approximately $50,000 and achieved a sustained performance of 1.19 Gflops and 1.26 Gflops, respectively. These two systems were demonstrated at Supercomputing'96 in Pittsburgh and were connected together on the floor for the first time using 16-100 Mbps Fast Ethernet point-to-point lines. The same code was run again and achieved a sustained capability of over 2 Gflops—without further optimization of the code for this new configuration. These results were of sufficient interest that a full-page news article entitled "Do it yourself Supercomputers" was published in *Science* in December 1996 . The importance of this experiment was that it demonstrated that real world science applications could be performed for a price-performance of approximately $32/MFlops; roughly 10 times less than that of commercial vendor offerings. Since then, significant reductions in cost have occurred such that essentially the same system can be acquired for less than $35,000. It should also be noted that in that same period, some vendor price reductions have been aggressive and in certain cases, measured price-performance differential between Beowulf and vendor offerings can be as little as a factor of 4 in favor of Beowulf-class systems.

A interrelated sequence of incremental changes in the complicated nonlinear tradeoff space has led to this dramatic new direction in parallel computing. One is the significant reduction in vendor supported high performance computing. While there were once many companies that were dedicated to producing high-end systems, today there are only four such companies and none of these consider this class of system to be their main product line. Companies like Cray Research, Inc. and Convex have been acquired by other mainline workstation and server corporations like Silicon Graphics, Inc. and Hewlett Packard, respectively. And these appear to be supporting market strategies that limit scalability of these systems. A second change is that the performance and capacity of M^{2}COTS-based systems have converged with the capabilities of their workstation oriented counterparts. PC processors and workstation processors are now within the same performance regime. This is due to the rapid increase in floating point performance improvement of PCs over the last three generations which is roughly a factor of 18 while the improvement for workstation processors over the same period is about a factor of 5, both remarkable but clearly the PC is catching up. Both class systems now use the same memories, and interface standards (e.g., PCI). Secondary storage for PCs using EIDE drives is close to that of SCSI-based systems in capacity and performance but significantly less expensive. A major difference between PC and workstation desktop computing used to be their software environments. While workstations had sophisticated Unix operating systems, PCs were limited to DOS. Today, PCs run Linux which is equal to or even superior to any of the vendor offered Unix-like environments and Linux-based PCs are replacing Unix workstations as the single user platform of choice in many academic institutions. It may be that Linux has the single largest installed base of any Unix-like software available today. With the advent of Fast Ethernet, low cost 100 Mbps communications is now M^{2}COTS providing $200 per port price for moderate configurations. Software for Fast Ethernet is now available under Linux for virtually every network interface card. As will be seen, this level of interconnect bandwidth is both necessary and sufficient for many applications and thus balanced clusters using PCs can now be implemented. Finally, MPPs from the era of the HPCC program established a relatively low level of expectation in quality of parallel programming model and runtime execution system. This expected paradigm, message passing, has been formalized and standardized in the MPI distributed programming library and is available on essentially all commercial parallel computers as well as on Beowulf. The result of all of these trends is that in the 1997 and 1998 time frame, clusters of PCs have become a viable alternative to vendor supplied parallel computers for many (but not all) science and engineering applications.

Beowulf-class Clustered PC Systems

What, then, is a Beowulf-class system? It is a combination of hardware, software, and usage which while not all encompassing, yields a domain of computing that is scalable, exhibits excellent price-performance, provides a sophisticated and robust environment, and has broad applicability. All Beowulf-class systems comprise clusters of PCs, sometimes referred to as a "pile of PCs". In most cases, the processor is of the Intel X86 family (e.g., 80486, Pentium, Pentium Pro) but does not exclude other architectures such as the DEC Alpha or Power PC if integrated in a low cost motherboard incorporating de facto industry interface standards.

Beowulf-class systems exclusively employ COTS technology, usually targeted to the mass market where cost benefits of mass production and distribution drive prices down. The exception to this is in global networking technology which while COTS, is not mass market due to the relatively small volume. In this case, a bound is placed on the cost per port and if networking vendors can provide their product within that cost framework, they are included.

Beowulf-class systems employ Unix-like operating systems with source code availability and low (or no) cost. The cost issue is important because to pay for an O/S license for each node of a multihundred-node Beowulf cluster could be prohibitively expensive. Source code availability is important because it enables custom modifications to facilitate parallel computation. Both Linux and BSD are good examples of such software. However, Solaris is also available from Sun Microsystems and with certain customers they have established formal working relationships that achieve the same end. It should be noted that other cluster PC work is being conducted based on the Windows NT operating system.

Beowulf-class systems employ message passing execution models. Usually these are explicit models such as MPI, PVM, or in the near future MPI-2, with some use of low-level mechanisms such as the direct use of sockets. Implicit use of message passing is also employed and is likely to become a larger part of the application codes over time. BSP and HPF provide the user with the appearance of a global name space but are sufficiently constraining as an API that the underlying compilers can automatically insert message passing calls, relieving the programmer of this burden. Some experimental software supported distributed shared memory programming models have been developed. But although the programmer needs not handle the actual generation of message traffic, locality management is still crucial to effective parallel execution. Many programmers feel that given this hands-on locality management requirement, they might as well use explicit message passing.

While there are many important applications in data processing, the focus for Beowulf-class systems is science and engineering applications. These are routinely (but not always) floating point intensive but even so involve a substantial amount of memory access behavior. Some such problems are truly data intensive such as data assimilation from large remote scientific sensors or data archiving for future scientific data product generation. Beowulfs are finding significant use in computer science research for the development of new tools and environments such as for metacomputing where control over a testbed is key to early development, even when the end software will be employed on publicly accessible systems. Probably, the fastest growing area of Beowulf use is as a learning platform for computer science education. These systems offer flexibility and price advantages over vendor offerings that make them ideal for pedagogical purposes.

Beowulf-class systems are just one of a large assortment of parallel computing system types. They are an example of general clustered computing. This is distinguished from metacomputing that may exploit computers all over the country through the internet, tightly coupled systems that include classical mainframes along with MPPs and SMPs, and vector and SIMD architectures that employ efficient hardware mechanisms for managing a certain type of data and flow control structure. Clustered computing assumes that each node has its own standalone operating system connected by moderate to high latency LANs. Workstation clusters such as the COW or NOW work examined many of the issues and developed some solutions now being confronted by the Beowulf-class systems. Another form of clustering is cycle harvesting where networked workstations lying idle are employed on off hours to perform some joint usually embarrassingly parallel workload. Beowulfs fall under the Pile of PCs category using M^2COTS for dedicated moderately coupled systems. NT clusters also fall under the Pile of PCs grouping as do PCs combined for distributed shared memory computation using hardware and software tools.

Advantages of Beowulf

Except for certain examples of special purpose devices and some applications of reconfigurable logic, Beowulf-class systems achieve the best price-performance available in any scalable system. There are some exceptions to this where applications are of such a form and structure that they are unsuitable for Beowulf-class systems, exhibiting poor performance. But for application codes suitable to them, Beowulf-class systems show superior performance for comparably priced systems.

But there are a number of additional advantages that make Beowulf-class systems particularly well suited for certain environments. One broad advantage is their rapid response to technology trends. The latest advances in microprocessor and other mass market technologies tend to find their way into PCs faster than to other platforms. MPPs, SMPs and other moderate to high-end parallel computers require long design cycles and the technology upon which they are based must be available at the beginning. Therefore, it is often the case that vendor parallel systems incorporate technology that is two to three years old. Technology found in Beowulf-class systems can be as little as a few month old (at procurement) because there is no design cycle; once part of a M^2COTS subsystem, it can be immediately migrated into a Beowulf.

Many manufacturers and distributors produce, integrate, and market hardware systems and subsystems that meet de facto industry standards developed through the evolution of the PC platform. This produces competition for virtually every aspect of technology making up Beowulf-class systems and there is no single-point vendor to which Beowulf is vulnerable. Even Intel has competition and must meet its own object code computability requirements. Thus the Beowulf user gets market-driven low prices with second-sourcing on essentially all parts.

One of the most powerful aspects of the Beowulf approach is "just-in-place" configuration. The organization or topology of a Beowulf-class system can be devised by the end user (or system administrator) at the time of installation. The structure can reflect the latest in technology opportunity, or special data flow paths associated with a particular application or workload. Furthermore, such organization can be reversed or changed to adapt to changing needs of new subsystem capabilities. This is not only true with the network but also for the subsystems integrated within a given nodes. As disk prices drop and densities increase, up grades to secondary storage become routine.
Beowulf-class systems are scalable. Initially, such systems were limited to 8 or 16 nodes with a couple of examples of 32- and 48-node systems in 1996. Now, there are many sites with systems comprising over 100 nodes and plans to implement systems of a few hundred nodes within the year. There is at least one example of a thousand node system in the works. Although not a Beowulf-class system, the ASCI Red system incorporates the identical chip level technology as that of Beowulf with the exception of the NIC (network interface controller) and shows effective computation out to many thousands of processors. Much work remains to be done on system organization, networking, and software

infrastructure to make these systems highly scalable across the broadest range of problems, but today, many problems are known to scale at least beyond a hundred nodes.

Beowulf-class systems leverage the enormous body of software developed by the computer science community and continues to draw upon this source as it integrates an ever increasing array of tools to facilitate use of these systems. Linux combined with the Gnu compilers and tools, X windows, MPI, and a host of other contributors makes up a powerful set of capabilities. This brings a lot of developers to arena either intentionally or otherwise to produce new software useful to Beowulf. These include the parallel programming methodologies found on virtually all other vendor supplied parallel computers such as MPI.

The NASA Beowulf Project

Beowulf-class systems and concepts have been developed, sometimes independently, at a number of R&D organizations around the country. One of the earliest was the Beowulf project, from which the name of this class of computing has been derived. The Beowulf project was initiated in late 1993 as part of the NASA HPCC Earth and space sciences program at the Goddard Space Flight Center. This general research program called for a single user "gigaflops" science workstation. At the time, that scale of performance was unavailable from the vendors for less than half a million dollars. An evaluation of the requirements for a scientific station for NASA showed that disk access capacity and bandwidth was far more important to user response time than was floating point performance. A typical session would involve the scientist loading large data sets from a central file server over a shared local area network (LAN). Because the disk and memory space of workstations at that time were relatively small, many of these accesses were redundant copying in the same data repeatedly. This led to long response times to the user, and congestion for shared file server and LAN resources. Analysis showed that for the price of a high-end scientific workstation, assumed to be $50,000, a cluster of low cost PCs could be assembled with an order of magnitude larger disk capacity and approximately 8 times the disk bandwidth. Such a system would comprise 16 80486 (100 MHz) PCs with 32 MBytes of memory and 1.6 GBytes of disk per node, integrated with 10 Mbps Ethernet. The system would be capable of a peak performance of approximately 1 gigaOPS. The requirement for an accessible operating system similar to those used on scientific workstations was satisfied by the then experimental Linux operating system. The critical gap was the lack of adequate low-level networking software. This was recognized as an important enabling technology. Fortunately, an expert in this area was available. All the elements of a new project were available: a charter, funding, a vision, the enabling technology, and the necessary talent. The Beowulf project was born.

The Beowulf project was devised to address a set of key questions to determine the viability of the opportunities this class of computing might provide. The first and most fundamental question was if and how to harness a pile of PCs to do real world applications? If so, what level of price-performance gain could be achieved over conventional vendor products of comparable performance? On such a system what was required to devise a software environment to manage PC clusters? And, could PC clusters be scaled to hundreds of nodes to achieve supercomputer performance? A number of issues pertinent to Beowulf-class systems relate to these questions. Hardware related subjects include interconnect bandwidth and latency, both of which impact system scalability. Software issues are the programming environments and resource management tools. Another issue is how these systems evolve in the presence of rapid technology changes. Finally, the possibility of supporting distributed shared memory models of programming and computation must be considered.

The Beowulf project has gone through three generations of processor technology and networking technology as well as critical transitions in motherboards and rapid improvements in disk. The focus for the project has been on applications, networking, facilitating middleware, and scalability. A year ago, JPL and Caltech began a Beowulf project with a strong emphasis on applications. NASA Ames Research Center also began the Whitney project, another Beowulf-class system with a strong emphasis on system tools. In the fall of 1996, using the new Pentium Pro (200 MHz) processor-based 16-node system, both JPL/Caltech and LANL performed the N-body calculation achieving > 1 Gflops sustained performance and > 2 Gflops on the floor at SC'96. Large systems of over 100 processors were assembled at both GSFC and JPL/Caltech in the Fall of 1997 to demonstrate and evaluate scalability. A number of tutorials have been conducted on "How to Build a Beowulf" at various conferences and a book of that title is in the works. All of these activities culminated in the NASA Beowulf workshop in October, 1997.

3. Major Findings

The first NASA Workshop on Beowulf-class Clustered Computing considered the broad set of issues related to this emerging technology. While some of the detailed issues are discussed in the following sections, the strategic findings of the workshop are summarized here.

<u>NASA Beowulf Sites</u>: Beowulf-class clustered systems employing mass market commodity off the shelf (M^2COTS) components are being employed by a number of groups at several NASA centers for a variety of purposes. This type of parallel computing is likely to be the single fastest growing category in 1998 both within NASA and across the nation's high performance computing community.

<u>NASA Beowulf Applications</u>: Beowulf-class clustered computing can be an important source of medium to high-end computing for many NASA mission problems. A number of NASA related applications have been successfully demonstrated on systems ranging up towards a hundred processors achieving several Gflops sustained performance and unprecedented price-performance.

Advantages of Beowulf: Beowulf-class systems are distinguished from commercial parallel systems in critical ways including superior price-performance, high flexibility, reconfigurability, rapid response to technology advances, invulnerability to single vendor decisions, stable system architecture class, and the human benefits of "computer ownership."
Limitations of Beowulf: Beowulf-class systems are not appropriate for all applications or user environments and they should be adopted only after determination of their suitability to meet specific requirements.

Network Bandwidth and Latency Issues: The primary bottlenecks to Beowulf-class system scaling and efficiency are network bi-section bandwidth, latency, and global synchronization. Fast Ethernet provides sufficient per node bandwidth in many cases but large scale systems are limited by global switch throughput. Advanced algorithmic techniques have reduced sensitivity to communication latency and slow global synchronization barriers such that Fast Ethernet latency can be tolerated by many applications.

Emerging System Software Environments: A critical mass of necessary system software is being assembled but additional development in functionality, robustness, and interoperability is required for full benefit of these systems to be realized. Environmental requirements differ dramatically between small single-user Beowulf-systems and large "dreadnought" scale systems supporting a broad user base and multiple simultaneous users. While the small system software environment appears to be converging, dreadnought support is still an open question and few users are completely happy in either case at this stage.

Next Generation Capabilities: Near term technology advances in processor, busses, and networking will make a quantum leap in capability at low cost in the next two years and Beowulf system methodology will be positioned to take immediate advantage of these improvements for end-user workloads.

NASA Beowulf Community Established: The workshop met its important objective of establishing a NASA-wide community in Beowulf-class computing and identifying near term directions necessary for achieving full benefits of its potential capabilities. A framework of web sites, e-mail lists, and special interest groups will be established by this community to facilitate NASA exploitation for "better, faster, cheaper" high-end computing.

4. System Area Network Technology and Scaling

The level of interconnect infrastructure to integrate the processing nodes needs to be sufficient to support application communication requirements but small enough to keep cost within appropriate bounds. It can be expected that as the number of nodes of a system is increased, that the cost of the integration network grows superlinearly. An important factor is the level of interconnect required for balanced computation, where the network does not become the bottleneck. A rule of thumb is that the network should cost approximately one quarter of the total system cost. Given that a current node cost is between $1600 and $1800 per node without communication, the cost for communication per node should be between $530 and $600 per node.

Several alternative technologies are being employed to integrate clusters of PCs. Fast Ethernet has a peak bandwidth of 100 Mbps. Each network interface controller (NIC) costs approximately $50. Latency for messages between applications between separate nodes is roughly 100 microseconds (less than 80 microseconds has been observed). Eight-way and 16-way non-blocking switches are available. With discounts, a 16-way switch costs between $2500 and $3000 per switch. For small systems (16 processors or less), the per node cost for communication is perhaps about $225, much less than the $600 per node that is considered acceptable.

Myrinet is a second high bandwidth network developed to integrate workstations and as a system area network (SAN). Its per node bandwidth exceeds 1 Gbps and it exhibits a latency of below 20 microseconds. Available today are 8-port switches. These are relatively inexpensive devices costing less than $3,000 a piece but are blocking and can lead to issues of contention under heavy loads or when used in large topological structures. Unfortunately, the NICs for Myrinet are expensive in the range of $1,400 per port. With the high price of the interconnect cables, a system employing Myrinet can dedicate half of the cost to the communication network; probably not the balance one would choose.

For small systems of up to 16 or 24 nodes, single-switch solutions using Fast Ethernet are simple and cost effective. Beyond that scaled systems require some compromises in cost and performance. A number of different techniques have been attempted. Just achieving up to 240 processors is not difficult requiring a two-level tree comprising nodes of 16-way Fast Ethernet switches. This costs little more per node as only a single additional switch at the root node of the tree is required. And where locality can be exploited, communication between processors on the same switch is just as fast as the smaller systems. But for system wide random communication, the root node switch can be a severe bottleneck as well as imposing a somewhat greater latency. One solution to this problem would appear to be to use multiple switches at this root network node but this turns out to be impossible because the "spanning tree" routing algorithms used by these switches preclude multiple paths existing between any two points. Thus, bi-section bandwidth can not be increased by ganging multiple channels.

If processing nodes can also serve as networking nodes, then highly varied and sophisticated topologies can be constructed. The impact is that each such processor node adds some delay in the message packet transmission time as well as subtracting from the overall compute capability of the processor itself. Within these constraints, the scalability of such organizations can be substantial in size. One additional difficulty is the limited number of NICs that can be employed from a single processor. NICs with up to four Ethernet ports are available. An up to four of these can be included in a single node, limited by the number of PCI bus slots available. A hypercube topology is possible with degree 4, 8, 12, or possibly 16 (ports) per node which could allow many thousands of processors to be interconnected in a single system. Small hypercube systems have been implemented with some success in spite of the additional processor node latencies. Another such structure with fixed degree (3 or 4) are toroidal topologies, again employing through processor communication. If anything, these permit even larger numbers of processors without increasing the number of ports per processor node. But bi-section bandwidth does not grow adequately while network latency does expand significantly. Such systems are probably only appropriate for certain classes of applications with well behaved and highly localized communication patterns. Hybrid topologies using both switches and through processor communication can provide superior performance and price performance when data access patterns across processors are known and taken into account. Switches connecting subsets of total system nodes can handle high bandwidth traffic without blocking while through-processor node communication permits high scalability and large system structures. This is particularly useful for data flow-like computing data streams.

Another approach is using a hierarchy of communication technologies for moderate bandwidth channels into each node and high bandwidth backplanes for global communication. An example of such a system is the Prominet P550 switch with 22 Gbps switch throughput connecting clusters of 20-port Fast Ethernet channels. Up to 120 ports can be so connected or multiples of these backplanes can be interconnected for even greater system size. Caltech's Center for Advanced Computing Research has a 160-processor system employing two such backplanes. Each handles 80 Fast Ethernet ports and the two backplanes are connected together by four channels, each capable of 1 Gbps data rate. This technology is likely to expand to enable dreadnought-scale systems of a few hundred processors and at a cost of about $500 per processor.

5. Applications and Algorithms

If there is a single objective of Beowulf directed applied research it is the near term application of these systems for high performance at unprecedented cost for a broad range of NASA problems. At issue are examples of such applications, their scaling characteristics, the impact of networking on application behavior, and the quality of support programming environments.

Latency is the biggest difference between a Beowulf and a supercomputer. However there are latency tolerant applications, e.g., when the resolution is sufficient but more physics per grid point gives more return. Highly tuned applications such as benchmarks are often latency tolerant. However, multiprocess applications often have gratuitous synchronizations that inhibit performance on Beowulf-class systems. One important factor is global barrier synchronization which is particularly costly on Beowulf-class systems. It is necessary to reformulate our applications to remove unnecessary (sometimes implicit barriers) synchronizations. A better understanding is needed about concurrent, latency tolerant, hierarchical memory (parallel out-of-core) algorithms. David Bailey of NASA Ames Research Center says that researchers at Ames are becoming interested in pursuing this problem. There is a chance that we can trade network bandwidth for latency by reducing the number of synchronizations.

Even in the short time that Beowulf-class computers have been available, a number of important scientific and engineering codes have been developed to perform real-world end user computations. Table 1 below provides a representative list with an emphasis on but not limited to NASA-related applications:

Table 1. Scientific and Engineering Codes Developed for Beowulf-class Computers

Organization	**Code**
ARC	Implicit CFD, FFTs, Multigrid NAS Benchmarks
Caltech	Astrophysical *N*-body Tree Codes, Simulation Result Analysis Codes, Smooth Particle Hydrodynamics Code, Vortex Dynamics, Reactive Chemical Flow
Clemson	Parallel File System, Gauss Siedel Sorting
Drexel	*N*-body Tree Code, SPH, Weather and Storm Front Modeling, Nuclear Shell Model Code, Particle Scattering
GMU	Combinatorial Optimizations
GSFC, Drexel	*N*-body Gravity Code

Table 1. Scientific and Engineering Codes Developed for Beowulf-class Computers, cont.

Organization	Code
GSFC, Drexel, CESDIS	Prometheus, 2-D PPM, Stellar Convection, 4 Gflops on the Hive
GWU	Level Zero Processing from Telemetry, GSFC code 500
GWU, GSFC, CESDIS	Wavelets, Image Registration
JPL	3-D Planetary Thermal Convection, Ocean Circulation Code Finite Difference Electromagnetics Code, Finite Element EM code, Physical Optics EM code, 3-D Fluid Flow with Multigrid. Parallel MATLIB, 2-D FEM EM with Scattering
NIH	Macro Molecular Simulations and Modeling: CHARMM, AMBER; Quantum Chemistry: GAMESS

Impact of Network on Application Performance

Applications vary significantly in the amount of interprocessor bandwidth required and the latency of such interconnection that they can tolerate. To understand the requirements of a range of applications is nontrivial. However, some studies on representative problems have been performed that are beginning to reveal the level of communication that may be necessary.

Studies conducted by Los Alamos National Laboratory comparing performance of the *N*-body gravitational tree code were conducted on Loki, a 16-processor Beowulf-class system using Fast Ethernet interconnect, with a 16-processor slice of the ASCI Red machine, which uses essentially the same parts but differs in its communication network. The ASCI Red network has much higher bandwidth and lower latency such that it might be considered almost infinite compared to the more modest Beowulf-class system. The studies showed that overall performance improvement of ASCI Red versus Loki was on the order of 25 to 30%.

Analysis performed at the NASA Ames Research Center on three of the NAS Parallel Benchmarks examined the impact of both network bandwidth and latency on a per node basis over a range of several hundred nodes. Two important results were derived from this investigation. The first showed that, at least for these problems, network latencies on the order of 100 microseconds is good enough with the sustained performance near that of the ideal case for more than 200 processors. The codes run (SP, LU, and BT) were developed to be latency insensitive so these results should not apply to any arbitrary, even parallel, code. But these results do demonstrate that for a set of benchmarks crafted to reflect NASA requirements, algorithms can be devised with sufficient built-in latency tolerance that Fast Ethernet technology is acceptable.

The second result relates to network bandwidth, again for the same set of benchmarks. At 100 nodes, degradation of performance for 8 MBytes/second per node bandwidth with respect to essentially infinite bandwidth is approximately 15% for the SP benchmark. At 300 nodes the performance degradation is estimated at 25% for this benchmark. Similar experiments run for LU and BT showed 4% and 8% degradation, respectively, at 100 processors. These findings indicate that Fast Ethernet provides acceptable bandwidth but that some performance improvement would be derived from some increase in network bandwidth. This implies that the use of channel bonding to support dual networks might be a useful approach. There is one underlying assumption of this study which has yet to be justified in practice. It assumes that there is sufficient global network bandwidth to support 100 Mbps per port. Furthermore, no consideration was given to the issue of contention which suggests, therefore, the possible need for non-blocking switches interconnecting hundreds of nodes. Such technology at sufficiently low cost is not available today.

An empirical study at the NASA Goddard Space Flight Center compared 10 Mbps Ethernet with 100 Mbps Fast Ethernet in a series of file copy experiments locally and between nodes. It showed that regular Ethernet (10 Mbps) was a bottleneck and severely constrained overall file copy throughput, causing a degradation of 80% in the worst case measured. In contrast, Fast Ethernet was shown to exhibit a degradation of about 15%, worst case. The results of these experiments again demonstrate that Fast Ethernet provides adequate network bandwidth for Beowulf-class systems and is balanced with respect to the other resources comprising these systems.

Algorithm Scaling Characteristics

The following distinguish the characteristics of application algorithms that scale from those that do not scale.

1. Applications that are written in a "parallel out-of-core" style tend to run well on the Beowulf-class systems. This is

because the granularity of the parallelization is large and explicit.

2. Applications that can be written without explicit global synchronizations tend to run well on Beowulf class systems. For example, explicit time step, gridded CFD codes can use pairwise send/receive pairs.

3. Applications that combine all of their communications into one massive communication, such as one global transpose.

Grid-based CFD, Image processing, Tree *N*-body codes already work well on a Beowulf-class system.

There are a number of application-driven roles that Beowulf-class systems can play in significantly advancing the computational objectives of NASA. Some of these are listed below.

1. Low cost, locally controlled computing platform for small groups doing NASA-supported computational science.

2. NGST and Space Interferometry missions both want control over the computing platform. An advantage of a system wholly owned by a project is that the project can control the queuing system. The Beowulf architecture can support an on-demand client/server system.

3. Low cost, locally controlled educational platforms.

4. Remote Validation Centers need inexpensive, reliable systems distributed around the nation to run a small set of standard applications for that region's specific data needs.

5. Data assimilation needs scalable computing systems with large I/O capabilities.

6. Multispectral data processing and image registration are both very parallelizable.

Application Performance Analysis

There is an established literature about scaling of selected applications. Future Beowulf application papers should include performance modeling to include the dependency on computation, memory bandwidth, interprocessor bandwidth, and barrier synchronizations. Exhaustive analytical formulas are great but impractical. Identify the parameters that are useful that can be collected in a automatic manner (e.g., Mike Warren's flop, page faults, integer ops, cache misses).The message passing libraries should give statistics about message traffic and synchronization. We recommend that an empirically fitted performance model can be included in Beowulf papers that try predict future performance on Linux systems. Someone could develop a utility that collect the statistics during a run for MPI-like XPVM or T3E apprentice. Someone could develop a utility that will take the collected statistics and create an empirical formula.

6. SYSTEM SOFTWARE and TOOLS

Beowulf-class systems require a robust software environment for application programming and resource management. NASA contributed significantly to the development of a set of guidelines for software environments on parallel systems under the leadership of Prof. Cherri Pancake of Oregon State University. The principal requirement areas are as follows:

— Application Development
— Debugging/Tuning Tools
— Low-level Programming Interfaces
— Operating System Services
— Ensemble Management
— Documentation

It was an objective of the NASA Beowulf workshop to survey available software and ongoing software development projects for each of these areas as they relate to Beowulf-class systems and to identify gaps within this framework that need to be addressed by future development and/or porting activities. For each of the areas specified above, a significant body of software exists or is under development. A detailed list is provided in Table 2 below.

Table 2. Software/Tool Areas Existing or Under Development

Software/Tool Areas	Name/Status

Application Development

Utilities	sh, csh, grep, egrep, sed, diff, vi, ex, make, et al.; all available via GNU
Languages	f77, C, C++, f90, linker, mixed language support
	GNU C, C++, f77
	PGI C, C++, f77, HPF
	Absoft f77, f90
	NAG f77, f90

Table 2. Software/Tool Areas Existing or Under Development, cont.

Software/Tool Areas	Name/Status
Debugging/Tuning Tools	
Interactive Parallel Debugger/Postmortem Debugger	ARC P2D2 project is being ported to Beowulf-class clusters
Profiling Tools	Single Processor; Standard Gprof Utility
Event/Message-passing Tracing Tools	PABLO project ARC AIMS project may be ported to Beowulf-class clusters
Hardware Performance Monitoring Tools	LANL has the beginnings of a hardware monitoring tool
Low-level Programming Interfaces (Libraries)	
Message Passing	MPI, PVM Argonne MPICH, LAM MPI, publicly available PVM
POSIX Treads	Available
Math Libraries (Optimized Single Processor)	FFT, BLAS, Random; BLAS and Random Number Generator Available
Math Libraries (parallel)	FFT, Plapack Available Array Permutation Timers (Wall-Clock, System, and User Time) All Standard Features Of Linux
Parallel I/O	Clemson PVFS parallel file system (beta quality?) ARC PMPIO MPI-2 I/O library (available) Argonne ROMIO MPI-2 I/O library (available) ARC ESP FS parallel file system (development just beginning)
Operating System Services	
TCP/IP	Standard Part of Linux
Files System (w/Long Names, >4GB File Systems, >4GB Files)	Available, Except >4GB Files Which is in Development
Job Queuing and Scheduling (Batch, Space-sharing, Time-sharing)	EASY LoBoS Queueing System ARC PBS Project
Accounting	ARC Acct++ Site-wide Accounting Project (will be ported to Linux)
Quotas and Limits Enforcement	

Other software requires enhanced development. For example, particularly important is a set of ensemble management tools, "Beoware, Grendel, or Beotools," that is under development and which is providing increasing control of the distributed resources as a single system image.

But for others, development is a priority. These include:

— Common Boot/ Install/Configure Package
— System Monitoring Web-Based Software
— Parallel rsh, ps, kill, top, etc.

Listed below are other software examples that require enhanced development as well.

— Rock Solid MPI
— Batch Job Queuing & Scheduling
— Parallel File System
— f90 (Absoft and NAG supply compilers), perhaps a grant to move g77 to g90
— Optimized math libraries (collect and put into RPMs); BLAS, Linpack, LApack, PLApack (rather than SCALapack); MIT FFT library
— Unified system image software
— Fast synchronization software
— Job accounting
— Gang scheduling

Documentation

There is a substantial body of documentation for Linux, MPI, and PVM. Some documentation directly related to Beowulf specific issues is available via the web. A new book, "How to Build a Beowulf", is being developed by Sterling, Becker, and Salmon.

Other Issues for Software Environments

A critical-mass is beginning to form, but would greatly benefit from more community development. It was agreed to create a new mailing list devoted to Beowulf developers (developers@beowulf.org), and to use the "beowulf.org" address as a clearing house for information. In addition, follow-up communication is essential, and a follow-on workshop should be considered.

In addition to concrete software development, more work needs to be done in three related areas: hardening software, software packaging and Beowulf-specific documentation. Considerable software has been written for Beowulf-class clusters that works for the author (and perhaps a few brave individuals), but isn't ready for prime time. Given additional time and resources, "hardened" versions of this software (e.g., parallel rsh scripts and hardware performance monitoring utilities) would be of great benefit to the larger Beowulf community. General agreement was reached that a common software packaging methodology would enhance cooperation, and that RedHat RPM format should be the basis for such packaging. Finally, Beowulf-specific documentation, beginning with a FAQ and quickstart guide, should be created.

Standards development also would enhance the Beowulf effort. Areas identified for standardization include the parallel model of execution (to foster communication among researchers); the parallel rsh interfaces (to permit portable script development); and a method of describing cluster architecture (to support automated tools for configuration, booting, and monitoring the cluster).

Finally, transfer of Beowulf technology was addressed. The discussion included approaches ranging from traditional organizational methods, to use of the GNU or Linux "free software methodology" where no restrictions are made on software distribution, to forming an agreement with one of the commercial vendors of Linux (e.g., RedHat) to give them non-exclusive rights to all software (letting them give it away). The consensus seemed to be that the best alternative was to use the free software methodology—anyone can contribute, someone acts as a central contact and integrator, and all developed software is freely distributed—which has resulted in such wide distribution, use and progress with Linux.

Beowulf-class systems come in two flavors: small and "dreadnought." The small systems are characterized by having about 30 nodes and a few users who run a couple of different applications. The dreadnought flavor is hundreds of nodes and many users running a broad range of applications. Although there are a variety of configuration and software choices, the architecture of small systems seems to be converging; larger systems, however, remain an open question.

HPF poses serious problems for Beowulf clusters due to unnecessary implicit synchronizations. However, the use of a global address space is still very appealing to many NASA scientists. There may be "raw" shared memory libraries that may work for a Beowulf-class system with tens of processors. For example, TreadMarks from Rice University has a relaxed consistency shared-memory model that relies on the user to insert the shared-memory synchronization library calls. A prototype of relaxed consistency shared memory was implemented at GSFC called Hrunting, but the work needs to be finished soon.

7. Scalability

The first Beowulf-class systems were small and medium size with many 8-, 16-, and 32-processor systems and a few 48-processor systems. These small- to medium-scale systems were well suited to the software and hardware resources available and were capable of a few Gflops performance on favorable applications. Systems beyond this size (up to a few hundred processors), referred to as "dreadnought-scale" are being planned and assembled and difficulties have been encountered, inhibiting their effective use under certain circumstances. A number of issues were explored to better understand the nature of scalability to dreadnought-scale Beowulf-class systems.

Metrics

Scalability spans a number of metrics in system capability which include the conventional parameters of all parallel/distributed machines:

— Flops
— Ops
— Memory Size
— Memory Bandwidth
— Disk Capacity
— Disk Access Rate
— Network Bandwidth
— Network Latency

Even this list can be expanded based on circumstances such as message packet size, cache behavior as a function of memory access patterns, similar issues related to disk access patterns, and so forth. But additional metrics of scalability are also important to Beowulf-class systems.

Efficiency. Efficiency relates to the ratio of sustained performance versus ideal performance of a multiple processor system. Efficiency is a function of workload scaling as well as system size. It is easier to get effective use of increased processing resources if the problem size grows as well, thus enabling better science in a given time. Alternatively, with a problem size invariant with system size, the same problem is executed in reduced time.

Cost per node. Cost per node is an important discriminator between Beowulf-class systems and vendor supplied systems. Ordinarily, this may be in excess of an order of magnitude difference, although some low end workstations exhibit price differentials (including special discounts) of less than a factor of four. But a fairer cost is that per delivered performance: for example, cost per Mflops or Gops per $M where performance is sustained for some suitable benchmark application(s). As the system size is scaled up, the cost in each case increases for a Beowulf-class system because the efficiency tends to decrease and the network cost per node increases superlinearly.

Number of Users and Applications. The number of users and applications that can be supported at one time as systems scale up is important. For small- to medium-scale systems where the cost is under $100K, these systems are likely to be operated in a single-user mode. But for dreadnought-scale systems where costs may exceed half a million dollars (a 1000-processor system using dual processor nodes will cost under $2M), multiple users and jobs can be anticipated to amortize cost and make better use of available resources.

Application diversity. Application diversity would characterize the generality of the system across algorithm and science/engineering problem types. This is difficult to quantify but is key to establishing both the utility and scalability of Beowulf-class systems.

Determining Factors

The scalability of a system class such as Beowulf depends on a number of complementing factors, each of which must scale accordingly in order for different magnitude systems to operate in a balanced manner. Much of the current work is in making each of these areas scalable to extend the utility of Beowulf systems to 10 Gflops and beyond.

Hardware M^2COTS parts impose limitations in scaling. Even those which claim to scale have not been rigorously tested under real-world application-driven conditions. Network hardware, especially switches, is among the foremost hardware components of concern. The number of disks on EIDE or SCSI channels (and the number of such channels) is a second factor. A third factor is the ability of memory bandwidth to support multiple processors per SMP motherboard.

File system scalability through parallel I/O software, parallel file servers, and RAIDS is essential to provide adequate usable bandwidth to secondary storage.
System software scalability has to support the hardware scalability. Some system software can run serial with no impact on system scalability while other operating system modules have to be changed both quantitatively (e.g., constants, table size) and in terms of methodology (e.g., linear search changed to $O(\log)$ search).

Application software must expose sufficient parallelism to spread across hundreds of nodes with coarse enough granularity that increase communication latency can be overlapped with computation.

Hardware Bottlenecks

Beowulf-class systems are pushing the frontiers of COTS hardware scalability many of the available components have been used almost solely in single processor systems or in small clusters of loosely networked PCs or workstations. In most ways, hardware scalability is directly or indirectly tied to network scalability as that is the medium of integration. Some additional hardware issues also impact scaling such as secondary storage or memory bandwidth as is discussed below. Hardware bottlenecks can be mitigated in part through software at the applications and systems levels. But these limitations tend to narrow the bounds of applications or system size that can ultimately be employed.

There are few network mechanisms capable of supporting Beowulf-class systems scaled to hundreds of nodes. Myrinet switches are 8-port and blocking. While large structures of these can be devised, the blocking cascades, and behavior may not scale for moderate or heavier loads. A number of suppliers of Fast Ethernet switches (8, 16, or 24 ports) use spanning tree routing algorithms that limit connections between any two points to a single path. Thus, ganging multiple channels is not possible to increase bandwidth and a bottleneck in bandwidth results. A new class of switches combining Fast Ethernet with very high bandwidth backplanes is coming on the market, but so far there has been little experience with these. Latency also grows with system scaling and network size which aggravates performance.

The cost of networks grow superlinearly with scale. Beyond a certain point the cost of the network can overwhelm the cost of the rest of the system. A cross-switch cost scales $O(n^2)$. A trade-off between cost, latency, and bandwidth results from different interconnect schemes even as the choice of topologies is limited by other hardware constraints like the spanning tree routing algorithms.

Fast Ethernet is good for small systems but the bi-section bandwidth per node decreases as the height of the tree increases to accommodate an increasing number of nodes, given the fixed degree of the switching nodes. Myrinet is fast and can grow to large sizes, but is initially expensive and an increasing proportion of the switch ports are used for internal network connections; it is easy to make a Myrinet network that is more than half the total cost of a large Beowulf-class system.

While most parts used by Beowulf-class systems are M^2COTS, there are no obvious external market forces driving the development of large-scale, low-cost switches. The market for small- and medium-scale Beowulf systems is expected to be relatively large due to the low entry-level cost and the need for such systems in academic institutions with constrained budgets.

Network bandwidth can be limited, not only by the network, but by its interface to the node. This may be a fundamental constraint of the memory bandwidth of the node or of the interface bus which today is usually 32-bit PCI. Some new motherboards include memory interleaving for higher memory bandwidth. Also, 64-bit PCI with a higher clock rate will be available shortly as well.

Memory bandwidth limitations impact another aspect of system scaling, the number of processors within a single SMP node. While motherboards with up to 4 processors per node are available, and in a fully cache-coherent configuration, the memory bandwidth is inadequate to support good utilization of these processors except in particularly carefully crafted codes that make particularly favorable usage of the two layers of caches on each processor.

Increases in hardware failures are likely as system scale grows. This is especially true during the initial phase of system operation. Experience shows that 95% of hardware failures occur as a result of infant mortality with the most likely candidates being disks, processor fans, power supplies, and network cards in that order of probability. Past that point, up times for moderate size systems have been measured in months.

Processor clock rates are likely to double over the next two years pushing additional requirements on the memory, I/O interface, and networking resources.

Software Bottlenecks

Scalability from a software perspective relates both to performance efficiency and to usability with increased processor count. With regards to performance, software must provide the means for representing algorithm parallelism and for effectively utilizing system resources. This also involves the efficiency of the system software itself. Software enhances usability in the presence of ever larger configurations by providing a "single system image" of the system to the user so that system control does not require explicit node-by-node manipulation and by providing timely system fault detection and recovery mechanisms.

The "single system image" approach requires system software that represents the various name spaces of the system to be unified across the system, providing a set of user commands that controls the operation of the machine in terms of these name spaces without requiring explicit logon to individual nodes. For example, this provides a global view of all running processes and tools such as "killall" independent of the number of processors involved.
Performance monitoring of system operation is essential for identifying hot spots in parallel execution and load balancing to determine bottlenecks and poor utilization which both impact system performance. User presentation should convey system wide behavior independent of the number of processors employed. It should also be able to represent only those processors in use for a given application in the case of space multiplexing.

Failure management becomes more important as the number of system nodes increases. Static techniques to ignore unavailable nodes are a minimum requirement, but dynamic means of fault detection, isolation, checkpointing, and restart are essential for large system configurations.

SMPs of two to four processors are currently supported by Linux but in a restricted way. Specifically, the current implementation does not permit more than one O/S service call to be performed simultaneously so that operating system work does not speed up with additional processors within an SMP node.

Near Term Directions

Beowulf-class computing system technology is moving aggressively forward in hardware capability and software sophistication. Some of the trends expected over the next two years are considered below. These assume that certain new project starts will be undertaken, in part, as a consequence of assessments such as this first Beowulf Workshop.

<u>Networking</u>. Networking is about to make another quantum jump in M^2COTS products. Gigabit Ethernet, which is now operational, may drop in price the order-of-magnitude required to make it cost effective for Beowulf-class computing. This will eliminate the local SAN connection as a constraining factor on scalability.

Bandwidth. Very high bandwidth network switch backplanes are beginning to become available and at a price that will permit Beowulf-class systems scaled to a few hundred processors.
Linux. New versions of Linux will permit parallel execution of O/S service calls within SMPs.

MPI-2. MPI-2 will provide new features for better control and management of system processes, communication, and synchronization.

I/O. For I/O, 64-bit, high bandwidth PCI buses will be incorporated in future motherboards to improve network and disk access to processor and memory systems.

Memory. Memory capacity and bandwidth through interleaving will be significantly enhanced on motherboards.

System Software. Its expected that continued improvement in systems software for job scheduling, debugging, and the system environment will occur incrementally through the contributions of many sources.

8. Heterogeneous Computing

One of the challenges to the viability of Beowulf-class computing is the issue of longevity of investment. How long can a particular resource be employed effectively before it becomes obsolete? The parallel computing industry has seen machines removed from the floor, not because they are unable to perform their task, but because their maintenance contract is too expensive for the level of performance delivered. In one instance last year, a particular 16-processor computer less than three-years old was powered down for the last time because the annual cost of its maintenance contract could purchase an entirely new and complete Beowulf of comparable performance. But similar trends challenge Beowulfs themselves. Technology advances are moving very fast. Every few months a new processor with a higher clock speed, larger cache, or improved instruction set becomes available, making the installed base of processors slightly less state-of-the-art. Is the only answer to throw out a system every eighteen months, or is there a better way to extend the contribution of the value made? An alternative is to consider the possibility of Beowulf-class systems comprising multiple generations of technology. In part this possibility exists because there are a number of different metrics of performance and just because a processor node loses ground in one dimension does not mean it can not perform important functions in the other dimensions. For example, in three generations, floating point performance has improved by about a factor of 20 but disk access rates have improved less than a factor of 3. This mixing of generations and distinguishing capabilities breaks the model of uniform processing elements and leads to the conclusion that cost effective Beowulf-class systems, at least of moderate to dreadnought-scale, will invariably become heterogeneous in nature.

There are a number of reasons why heterogeneous computing will be the norm for Beowulf-class computing. Longevity of investment permitting multiple generation machines is one. Another reason is that the just-in-place feature of Beowulf assemblage encourages these systems to expand in time. Each new part is generally optimized for the sweet-spot in price and capability. For example, disk prices have plummeted over the last year resulting in new nodes having twice the disk capacity of the older ones. This is a form of heterogeneity. Processor cache sizes are changing. Currently, Pentium II processors with MMX and 300 MHz clock are being selected for Beowulf sites instead of the Pentium Pro at 200 MHz, and systems that are expanding may also start to incorporate these newer processors. Network structures also do not expand uniformly. This is in part due to the rather stringent size constraints and in part because it is easier to incrementally add subnetworks than reconstruct the entire system network from scratch (although it is not all that difficult). Going from 16 processor to 18 processors may require the addition of a whole new layer to the global network. When a Beowulf-class system is being used for a particular and known workload, data movement requirements can be supported by specific network structures to match data rates resulting in non-uniform topologies. Different processor nodes can play different roles. There has been a long history of using some processor for computation while others are dedicated principally to I/O or secondary storage. This diversity of function across the processor array will be if anything extended in Beowulf-class systems. The availability of 2- and 4-processor SMP nodes provides yet another degree of freedom in system structure. But the sharing of such resources as I/O busses and memory bandwidth imposes more variation in overall system behavior.

The implications of employing heterogeneous systems are far reaching and will require advanced means of using future Beowulf computers as well as sophisticated system software to support such methodologies. Task scheduling must deal with differences in node performance. In order for this to be possible, parallel applications will have to be structured with finer granularity tasks to balance variations in node throughput. Otherwise, system response time would be fixed to that of the slowest computing resource. Distributed file managers must contend with non-uniform node disk capacities and complex access handling throughputs. User optimizations have to be effective across variations in system scale and configuration. Ultimately, system resources have to be virtualized with respect to the user workload, presenting a sea of resources to all users supported by runtime adaptive allocation system software. These requirements will impact all levels of software including application algorithm design, language constructs, compiler controls and optimizations, and runtime system software policies and mechanisms.

9. Future Directions

The in-depth review of the status, capabilities, limitations, and opportunities for the use and evolution of Beowulf-class systems identified a number of near-term tasks that, if performed, would accelerate the utility of Beowulf-class systems in meeting NASA requirements as well as those of the general community.

Strategic Directions: There is the opportunity to significantly enhance parallel computing capability through Beowulf-class systems and ways should be devised to enable their early use.

System Software Development: Near term development of a robust Beowulf software environment should be supported by employing existing tools where possible, integrating research system software where appropriate, and developing (or sponsoring) new software components where necessary. In support of this, a common framework for packaging all related software should be adopted.

Hardware Systems: Scaling studies of critical hardware and software components should be conducted to establish means and methods for implementing Beowulf-class system structures across a broad range of capabilities and size from small through "dreadnought" scale systems.

Applications and Algorithms: Suitability of Beowulf for a broad range of applications should be determined and the underlying factors quantified and understood. A new generation of latency tolerant algorithms will have to be developed for the important computational schema. This must include detailed performance measurement and modeling of parallel applications and may require new monitoring tools. Needed are optimized serial and parallel libraries for important classes of computing such as BLAS and FFT.

10. Summary of Conclusions

Beowulf-class computing derives its heritage from many years of experience in hardware technology, parallel computing methodologies, and computational techniques. The driving force behind the relatively sudden expansive application of the Beowulf approach to cluster PCs is the recent maturity, capabilities, and synergism of these factors that bring into balance the requirements, costs, and delivered performance. This paper is concluded with a summary of the status of Beowulf-class computing as understood by the NASA community represented at the recent workshop.

It has been repeatedly demonstrated that Beowulf-class systems and other parallel systems comprising clusters of PCs are very good for some real-world problems. As techniques for harnessing such clusters are better understood, the range of applications will grow.

The price-performance of Beowulf-class systems is excellent, often substantially exceeding any other form of computing system. As a result, it makes possible access to parallel computing at an unprecedented degree due to its low entry level cost.

Not all applications and algorithms are suitable for PC clusters, including Beowulfs. This is due either to insufficient parallelism that scales with system size, or high communications requirements and latency sensitivities.

Historically, workstation microprocessors significantly out-performed their PC counterparts, especially in floating point capability. That gap has closed almost entirely and is anticipated to be eliminated in the next generation of PC microprocessors. This closure is due in part to the factor of 20 floating point performance gain achieved across the last three generations of Intel X86 architectures as well as the migration of the DEC Alpha microprocessor family down to the PC market.

Historically, workstations were ten times the cost of PCs and per node costs of parallel systems versus PCs could be as much as a factor of 25. Recently, vendors have been aggressive in cost reductions and with the still slight edge in performance of these systems, the price performance gap has been closing. While order of magnitude ratios are still achievable for Beowulfs running particularly well suited applications, in other cases Beowulf-class systems have been seen to deliver price-performance advantage of only about a factor of four.

The major advantage of implementing MPPs with custom technology was to provide tight coupling of resources by high bandwidth and low latency internal system area networks (SAN). But COTS interconnect technology has advanced dramatically in the last three years and for applications with modest communications requirements, is good enough. This trend is likely to continue over the next couple of years with order of magnitude improvements in communication price performance and latency.

Shared-memory systems are seen to provide a superior user interface with a single-system image when compared to distributed-memory systems (DSMs) such as Beowulfs. But even on DSMs, users often have to be directly engaged in the management of locality by explicit allocation of data objects and executable tasks to separate processing nodes. Many computational scientists consider this substantially detracts from the value of DSM and believe that one might as well use message-passing under these circumstances as provided by Beowulf-class systems environments.

Institutional and user needs in price and performance differ widely and, while one class of users might find price-performance to be the dominant issues, others may find customer support to be more important to the value of the end system. Beowulf-class computing provides one subspace in the overall tradeoff space. Therefore, it is likely in the foreseeable future, that institutional computing will—and must—include a mix of system types and capabilities including, but, however, not limited to, Beowulf-class systems. Such a mix is likely to provide the best center-wide resources when measured by overall user satisfaction.

Acknowledgments

Over the last several years, many people have made substantial contributions to the evolution and effective application of Beowulf-class computing. These include those who contributed to the first NASA workshop on Beowulf-class clustered computing. Among those are Bill Nitzberg and David Bailey of Ames Research Center, Robert Ferraro of JPL, Clark Mobarry, John Dorband, and Jim Fischer of the Goddard Space Flight Center, Phil Merkey of USRA CESDIS, Dan Ridge of the University of Maryland, and Tarek El-Ghazawi of George Washington University, as well as the important work on clustered computing conducted at the University of Wisconsin and UC Berkeley. Many others could be cited as well. This paper and the emerging computing methodologies it represents owes much to those and many others who have contributed much of their time and talent. To all who have participated in this revolution, the authors owe a deep debt. Our thanks also to Mike MacDonald for his help in the preparation of this paper.

REFERENCES

[1] Bailey, David, T. Harris, W. Saphir, R. Van Der Wijngaart, A. Woo, and M. Yarrow "The NAS Parallel Benchmarks 2.0," *Technical Report NAS-95-020*, 1995.

[2] Becker, Donald J., Thomas Sterling, Daniel Savarese, Bruce Fryxell, Kevin Olson, "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation," *Proceedings, High Performance and Distributed Computing*, 1995.

[3] Becker, Donald J., Thomas Sterling, Daniel Savarese, John E. Dorband, Udaya A. Ranawak, Charles V. Packer, "Beowulf: A Parallel Workstation For Scientific Computation," *Proceedings, International Conference on Parallel Processing*, 1995.

[4] Becker, Jeffery C., Bill Nitzberg, and Rob F. Van Der Wijngaart, "Predicting Cost/Performance Trade-offs for Whitney: A Commodity Computing Cluster," *NAS Technical Report NAS-97-023*, 1997.

[5] Fineberg, Samuel A., and Kevin T. Pedretti, "Analysis of 100Mb/s Ethernet for the Whitney Commodity Computing Testbed," *NAS Technical Report NAS-97-025*, 1997.

[6] Reschke, Chance, Thomas Sterling, Daniel Ridge, Daniel Savarese, Donald Becker, Philip Merkey, "A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation," *Proceedings, High Performance and Distributed Computing*, 1996.

[7] Ridge, Daniel, Donald Becker, Thomas Sterling, Philip Merkey, "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs," *Proceedings, IEEE Aerospace Conference*, 1997.

[8] Salmon, John K., Michael S. Warren, and G. S. Winckelmans, "Fast Parallel Treecodes for Gravitational and Fluid Dynamical N-Body Problems," *Intl. J. Supercomputer Appl.*, **8**, *29-142*, 1994.

[9] Salmon, John, and Michael S. Warren, "Parallel Out-of-Core Methods for N-body Simulation," *8th SIAM Conference on Parallel Processing for Scientific Computing, Philadelphia*, 1997.

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

[11] Warren, Michael S., Donald J. Becker, M. P. Goda, John K. Salmon, and Thomas Sterling, "Parallel Supercomputing with Commodity Components," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), 1372-1381*, 1997.

[12] Warren, Michael S., John K. Salmon, Donald J. Becker, M. P. Goda, Thomas Sterling, and G. S. Winckelmans, "Pentium Pro Inside: I. A Treecode at 430 Gigaflops on ASCI Red, II. Price/Performance of $50/Mflop on Loki and Hyglac," *Supercomputing'97*, 1997.