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

Daniel Ridge Donald Becker Phillip Merkey
USRA Center of Excellence in Space Data
and Information Sciences
Code 930.5 NASA Goddard Space Flight Center
Greenbelt, MD 20771
{newt,becker,merk}@cesdis.gsfc.nasa.gov

Thomas Sterling
High Performance Computing Systems Group
Jet Propulsion Laboratory
Center for Advanced Computing Research
California Institute of Technology
tron@cacr.caltech.edu

Abstract— The rapid increase in performance of mass market commodity microprocessors and significant disparity in pricing between PCs and scientific workstations has provided an opportunity for substantial gains in performance to cost by harnessing PC technology in parallel ensembles to provide high end capability for scientific and engineering applications. The Beowulf project is a NASA initiative sponsored by the HPCC program to explore the potential of Pile-of-PCs and to develop the necessary methodologies to apply these low cost system configurations to NASA computational requirements in the Earth and space sciences. Recently, a 16 processor Beowulf costing less than $50,000 sustained 1.25 Gigaflops on a gravitational N-body simulation of 10 million particles with a Tree code algorithm using standard commodity hardware and software components. This paper describes the technologies and methodologies employed to achieve this breakthrough. Both opportunities afforded by this approach and the challenges confronting its application to real-world problems are discussed in the framework of hardware and software systems as well as the results from benchmarking experiments. Finally, near term technology trends and future directions of the Pile-of-PCs concept are considered.

Table of Contents

1. Introduction

One of the most remarkable recent technological advances has been the accelerated growth of computational performance. In the last decade, feature size of integrated circuits has shrunk in each dimension by an order of magnitude as yields and die sizes have both increased significantly. The result is a three order of magnitude increase in number of devices per chip, with a similar increase in total chip speed. In a field that has been revolutionized by those advances, the PC has been the greatest beneficiary of that dramatic success. Over the last five years, workstation microprocessors have experienced a rate of performance increase of about 50% per year. As extraordinary as that is, it has been easily surpassed by the performance increase of PC-class microprocessors which has exceeded a factor of two each year for the past four years.

The PC market is two orders of magnitude larger than the workstation market, and the resulting economies of scale have allowed PC prices to decrease while sustaining dramatic performance increase. Today, PC performance overlaps the range of workstation performance, with only the highest speed workstation microprocessors remaining faster than PC processors.

With such change, so comes opportunities. The potential of harnessing the power of parallel processing at the cost of PCs was identified as of possible importance to NASA mission critical applications and consistent with that agency's objective of "cheaper better faster". In real terms, lower cost computing for large scale problems means more science performed per dollar.

In early 1994 the Beowulf project was initiated under the sponsorship of the NASA HPCC Earth and space sciences project to investigate the potential of clustered PCs for performing important computational tasks beyond the capabilities of contemporary workstations but at no greater cost. In October of 1996, it was announced that a Beowulf system had exceeded a Gigaflops sustained performance on a space science application for a total system cost of under \$50K, a breakthrough in performance to price that may have significant implications and impact on a wide range of industrial applications including aerospace.

This paper presents the findings of the three year Beowulf project, the techniques employed to achieve this objective, and recent results from benchmarking experiments that has demonstrated the success of this project.

2. BACKGROUND

NASA HPCC program

The NASA HPCC program was initiated in January, 1992 with the far reaching agenda of advancing the state of massively parallel processing (MPP) and applying it to major computational problems important to NASA mission objectives in computational aeroscience (CAS) and Earth and space sciences (ESS). The ESS project represents a computational domain that includes direct manipulation of large data sets by end user scientists. In this context, the need for powerful end user terminal capability was identified early on in the program definition. Whether the large data set comes from the simulation of large scale physical phenomena, such as the evolution of the galaxy or plasma convections in the solar corona, or from remote sensing platforms such as the planned EOS, scientists need to acquire, examine, explore, manipulate, visualize, and sometimes transform large collections of complex data. Sometimes this may involve numerically compute intensive activities but it almost always involves large amounts of data movement.

A component of the ESS project was specified under the title of "Gigaflops Scientific Workstation". Although it was not clear at the time how this goal would be addressed, it was recognized that the principle objective function to be optimized was end user response time. For many workstation requirements this involved the data access time from secondary storage. Usually, this data would reside on a shared file server via a common local area network (LAN). Ironically, in many cases the same data would have to be accessed repeatedly during a working session because the typical workstation simply did not have the disk capacity to hold all of the requisite data. The result was long latencies to file servers, sometimes tedious response cycles, and burdening of shared resources. The Beowulf parallel workstation project was initiated to address this challenge.

PoPC: A Pile of PCs

A *Pile-of-PCs* is the term used today to describe a loose ensemble or cluster of PCs applied in concert to a single problem. It is similar to *COW* cluster of workstations and *NOW*[1] network of workstations, but emphasizes:

- mass market commodity components,
- dedicated processors (rather than scavenging cycles from idle workstations), and
- a private system area network (SAN)

all with the goal of achieving the best system cost/performance ratio.

Beowulf adds to the PoPC model by emphasizing

- no custom components,
- easy replication from multiple vendors,
- scalable I/O,
- a freely available software base,
- using freely available distribution computing tools with minimal changes, and
- returning the design and improvements to the community.

The Pile-of-PCs approach exploits components that respond to widely accepted industry standards and benefits from prices resulting from heavy competition and mass production. But the Pile-of-PCs approach has other intrinsic advantages that make them of serious interest for certain niche communities and provide a complementary computing medium to high end workstations, symmetric multiprocessors, and scalable distributed memory systems.

One advantage is that no single vendor owns the rights to the product. Many vendors provide essentially identical subsystem types such as motherboards, peripheral controllers, I/O devices, and packaging. Subsystems provide accepted, standard interfaces such as PCI bus, IDE and SCSI interfaces, and Ethernet communications.

A second advantage is that the Pile-of-PCs approach permits technology tracking. In a rapidly changing industry, where a generation may be less than a year and pricing varies significantly quarter to quarter, this approach allows computing systems to be acquired with the best, most recent technology and at the best price. As an example of this, no two Beowulf Pile-of-PCs (and there are a number of them around the country) are exactly the same, although they all run the same software.

This leads to yet another advantage, that of "just in place" configuration. User's needs vary, sometimes dramatically. And its not always clear what the required configuration is. The Pile-of-PCs approach permits extreme flexibility and user-driven decisions about how such a system configuration should evolve. Systems are not preconfigured by a vendor, limited to the vendor's current options lists which may be months out of date. Users can pick and choose from a wide array of sources, try things out, and change the configuration over time.

Beowulf exploits readily available, usually free, software systems that are nonetheless as sophisticated, robust, and efficient as commercial grade software. The software is derived from community wide collaborations in operating systems, languages and compilers, and parallel computing libraries. These are comparable to the quality of vendor offered software systems in many cases.

Two of the most widely used operating system in this class of distributed computing are Linux[2] and BSD: Unix/Posix systems available over the net at no cost. Both have commercial distributors and available commercial support services, full X windowing, most popular shells, and standard compilers for the most programming languages. The two major message passing libraries, PVM and MPI, are both available for these systems and widely used by the community. In addition, source code for many of these is available, permitting easy customization and redistribution without legal constraints typical of proprietary software products. Beowulf uses the Linux operating system for its better performance, better availability of source code, better device support, and wide user acceptance. In particular, Linux based PCs are becoming a mainstay of academic computer labs for their sophistication, accessibility, and low cost.

Two reasons why the Pile-of-PC approach must be considered relate to technology and industry trends. In technology, there is a convergence of workstation and PC microprocessors, with both outstripping the performance growth of traditional vector supercomputers. This country's first true Teraflops computer, being built for the DOE ASCI program, is being provided by Intel using components that are almost the same as those found in Pentium Pro based PCs. The next generation PC microprocessor is expected to be included in at least one scalable high performance computer offered by a major vendor. At the high-end, DEC is working hard to migrate their Alpha microprocessor into the PC market by offering low cost motherboards and compatible software.

In industry the comparatively high performance computing market cannot cannot sustain a separate research and development path. This has been made clear by recent changes in the industry. Cray Computer Corporation closed its doors. Convex was acquired by HP. And Cray Research Inc. was acquired by SGI. With the possible exception of Tera Computer Company, there is no computer company dedicated solely to the production of high performance computers. Thus, an important convergence is underway and, ironically, the Pile-of-PCs approach may be the asymptote.

Research Issues

While Pile-of-PCs is proving a successful path to parallel computing, there are many issues in the realm of applied research that need to be addressed. These relate primarily to resource management and the software tools for distributed computing. The Beowulf project has been addressing these by identifying key gaps in available tools and implementing new user tools to fill those gaps. The collection of software tools coming out of the Beowulf project is known as "Grendel" and is a continuously evolving set. Another challenge of the Pile-of-PCs approach, as exemplified by Beowulf, is the relatively long latencies and modest interconnection bandwidth provided by low cost networking such as Fast Ethernet. This is being addressed by software performance tuning, aggregating networks, and rich interconnect topologies.

As has been the case in the past with other distributed memory parallel computing systems, applications need to be written using parallel message passing for explicating the algorithmic parallelism. But in addition, these algorithms need to be latency tolerant, overlapping computation with communications for greatest efficiency. This is significantly more difficult than sequential programming styles embodied by e.g. Fortran 77. This approach is becoming increasingly acceptable to the user community because these systems are primarily used by computational scientists who have exploited the performance benefits of MPPs of previous generations and therefore are familiar with the issues and methods associated with employing this class of system. This does not make it good; but does make it tolerable, and of more importance, of practical use.

This paper explores the realm of Pile-of-PCs from the base of experience derived through the Beowulf project. The intent is to convey to a potential user community the opportunities and potential pitfalls of harnessing this emerging class of low cost high performance computer. This paper is the first reasonably complete presentation of the diverse topics related to this new challenge. The paper provides a description of the architectures used to date, the basic system software and advanced tools being developed, and accomplishments with application programming including performance. By targeting this paper to a major sector of the applications community, it is hoped that this technology may prove of use and accelerate the process by which high performance computing is applied to the computational challenges.

3. Beowulf Architecture

Characteristics

Beowulf now represents a family of systems that have tracked the evolution of commodity PC hardware and have enjoyed an increasing range of applications. The Beowulf project is driven by the need for high performance scientific computing within the Earth and Space Sciences (ESS) community; a community with a diverse set of requirements that continues to fuel the evolution of the Beowulf-class machines.

The Beowulf Parallel Workstation architecture, an important testbed for emerging ideas in PoPC clusters, was the original response to the needs of the ESS scientists. There are two constraints on the workstation architecture: It must use exclusively commodity hardware and a Beowulf workstation populated with disk and memory should cost no more than a high-performance scientific workstation (e.g. a low-end Origin 200 system from Silicon Graphics), roughly $50,000.

Earth observing applications have developed a body of codes for message-passing machines that port easily to the new architecture. These are large out-of-core classification and registration problems usually programmed SPMD (Single Program Multiple Data) style, with owner-computes data distribution patterns; they present a requirement for high performance image processing coupled with high aggregate bandwidth disk subsystems that can take advantage of these data distributions. To accommodate these needs Beowulf places disks on every node, makes no distinction between compute and I/O nodes, and achieves high network bandwidths by aggregating multiple channels of commodity Fast Ethernet hardware (Figure 1).

Beowulf-class machines are also used as cost-effective platforms for large multi-body codes (N-body, particle-in-cell, DSMC) codes. These large, typically in-core, problems also benefit from high floating-point performance but demand high bandwidth and low latency interprocessor communications. Along with high bandwidth interconnect, the broad availability of small-scale multiprocessor (2 to 4 processors) system boards provide an extremely cost effective way to increase floating-point performance for CPU intensive applications.

Beowulf clusters have been assembled around every new generation of commodity CPU since the 1994 introduction of the 100MHz Intel DX4 processor. The current price/performance point in desktop architectures, Intel's 200MHz Pentium Pro CPU, is the key to a new generation of Beowulf-class machines that retain workstation-level end-user costs but perform favorably (bracketed by a factor of two) against many commercial distributed memory supercomputer architectures on a per-node basis: IBM's SP2, the HP/Convex SPP series, and the Cray T3D.

The latest Beowulf clusters installed at Caltech, Los Alamos National Laboratory, and NASA Goddard Space Flight Center, are all 16 node Pentium Pro machines with configurations that reflect the specific needs of their users. The Caltech and Los Alamos machines (Figures 2 and 3), built for N-body galactic gravitational simulations, feature some of the fastest networks available in the commodity marketplace. Cal-

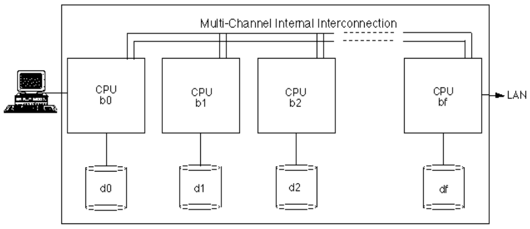

Figure 1: Beowulf Parallel Workstation Architecture

tech's 'Hyglac' cluster is interconnected with degree-1 Fast Ethernet connected to a 16 port crossbar switch. Los Alamos' machine, 'Loki', has a degree-5 Fast Ethernet topology connected as a degree-4 point-to-point hypercube plus degree-1 switched network; the switched network is bypasses long routes through the hypercube and avoids the low performance of the broadcast and multicast operations characteristic of simple hypercubes.

NASA Goddard's latest machine, also a Pentium Pro cluster with 16 nodes, contains 100GB in distributed disk with an aggregate memory-to-disk bandwidth in excess if 1 Gbit per second. These nodes are connected with the 'classic' Beowulf network — dual Fast Ethernet segments bonded transparently in software into a single logical network. This is a general purpose machine built for applications and software development; largely testbed work for the development of a mass storage system.

The most aggressive Beowulf-class machine being considered at this time is the jointly funded, DARPA and NASA, project to develop a network attached secondary storage system. This will be a 64 node, 128 processor system with a terabyte of distributed disk space and aggregate external bandwidth of a gigabyte/second. This special-purpose machine would step beyond the usual \$50,000 price point for Beowulf clusters and will take advantage of a hybrid network topology to achieve better scaling. The nodes will be collected into a series of 8-node 'meta-nodes' each connected internally with Fast Ethernet. These meta-nodes will be attached to a 1.2Gbit Myrinet crossbar, building the cluster into a shallow *fat tree*. (Figure 4)

On the other end of the Beowulf spectrum, clusters are being built to a slightly different set of requirements at Universities across the country. A number of schools: Drexel, GMU, Clemson, University of Illinois at Urbana-Champaign, and Caltech among them belong to an academic Beowulf consortium and assemble machines as frequently for pedagogical purposes as for research. These machines tradeoff absolute performance for increased parallelism—by substituting smaller disks, less memory, and slower CPUs. These machines serve as teaching machines and application reference platforms for distributed computing and are built for dollar amounts within the reach of any university.

4. Grendel Software Architecture

Beowulf-class machines leverage available software as well as off-the-shelf hardware. The Beowulf system is based on the widely available Linux operating system. Linux is a full-featured clone of the UNIX operating system originally designed for x86 processors and extended recently to support all common desktop architectures. In addition to portability, the Linux kernel features POSIX compliance, a TCP/IP protocol stack with a sockets interface, very broad device support, dynamically linked shared libraries, interprocess communication and an efficient virtual memory subsystem with unified buffer cache. The Linux kernel and much of

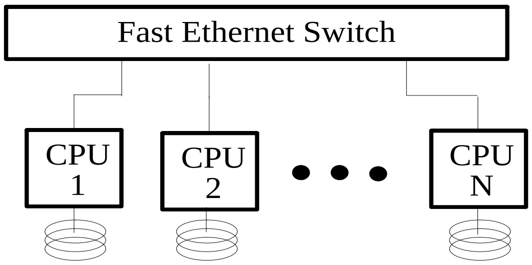

Figure 2: Caltech ‘Hyglac’ cluster

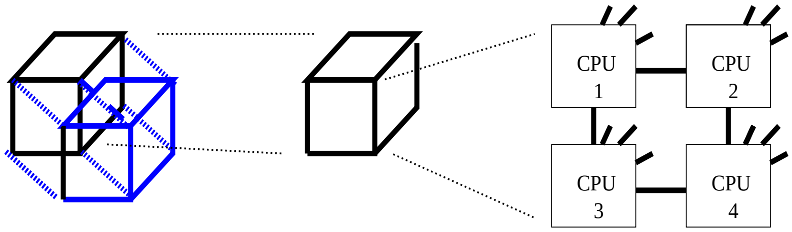

Figure 3: Los Alamos ‘Loki’ cluster

the basic supporting software is distributed under the terms of the Free Software Foundation’s GNU Public License [3] which insures that source code to the system is available, that we can easily share improvements, and that there are no per-node royalties.

The Beowulf software environment, Grendel, is implemented as an add-on to commercially available, royalty-free base Linux distributions. These distributions include all of the software needed for a networked workstation: the kernel, Linux utilities, the GNU software suite, and many add-on packages. Initially we used the very popular Slackware distribution. We are now migrating to the RedHat [7] distribution with its better package management and upgrade system.

The Beowulf distribution includes several programming environments and development libraries as individually installable packages. PVM, MPI, and BSP are all available. SYS-V–style IPC and p-threads are also supported. A considerable amount of work has gone into improving the network subsystem of the kernel and implementing device support. Most of these changes have been incorporated into the kernel source code tree.

In the Beowulf scheme, as is common in NOW clusters, every node is responsible for running its own copy of the kernel and nodes are generally sovereign and autonomous at the kernel level. However, in the interests of presenting a more uniform system image to both users and applications, we have extended the Linux kernel to allow a loose ensemble of nodes to participate in a number of global namespaces. A guiding principle of these extensions is to have little increase in the kernel size or complexity and, most importantly, negligible impact on the individual processor performance.

Global Process ID Space

Normal UNIX processes ‘belong’ to the kernel running them and have a unique identifier within that context. In a parallel distributed scheme it is often convenient for UNIX processes to have

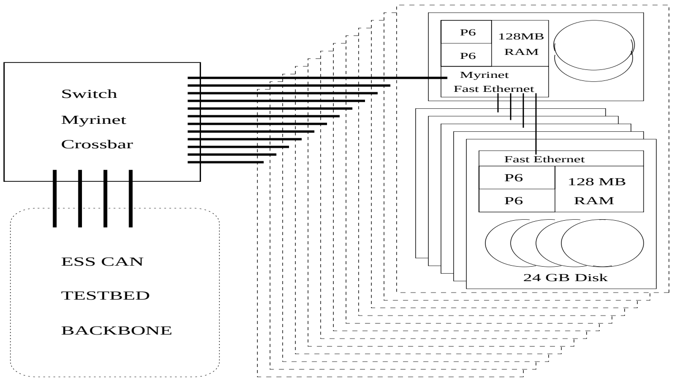

Figure 4: Beowulf Mass Storage System

a process ID that is unique across an entire cluster, spanning several kernels. Some UNIX systems, notably Linux on the Fujitsu AP1000+ [8] multicomputer, support directly the notion of a SPMD context of execution where multiple copies of the same code run on a collection of nodes and share a UNIX process ID.

A more generally useful scheme is available at the library layer in PVM. PVM provides each task running in its virtual machine with a task ID that is unique across all the hosts participating in the virtual machine. The PVM API incorporates library calls that provide the functionality of UNIX kill() and getpid().

Ideally, such a mechanism would be transparently available to all processes, not just those written and compiled to use a specific library. We have implemented two Global Process ID (GPID) schemes. The first is independent of external libraries. The second, GPID-PVM, is designed to be compatible with PVM Task ID format and use PVM as its signal transport. The traditional UNIX calls `kill()` and `getpid()` work transparently with both schemes.

GPID-PVM makes use of an undefined range in the PVM Task ID space. GPID-PVM reserves the PID range from 1 to NODE_MAX_PROCS for 'local' processes – processes (such as init) that by necessity are replicated across the cluster and would clutter a global process ID space. With this scheme, children inherit their type (global or local) from their parent unless the `clone()` system call explicitly instructed otherwise at process creation time. In an example system, each node might right run a PVM daemon as its master dispatch mechanism. Each PVMD would clone() itself immediately to migrate into the global PID space; all children spawned by that daemon would also exist in the global space. The GPID definition includes fields inherited from PVM that provide special IDs for PVM and other dispatch daemons.

Performance Impact

Runtime PID assignment requires no internode communication; kernels assign PIDs from their static non-overlapping GPID range. The added work is only one (or rarely, two) variable references per process creation. Remote delivery of signals is significantly more complicated and does require internode communication, but this work is done in what was formerly an error path for signals to invalid PIDs.

A guiding principle for system services has been to separate policy from mechanism, embodying the mechanism within the kernel and the implementing policy outside. Thus setting the process ID range, reliable delivery of signals, and recovering from failures is handled by a user-level process.

An additional impetus for the combined kernel-level and user-level implementation is that there are few paradigms where a kernel is expected to be an endpoint for communication rather than merely handling communications on behalf of user processes. This presents a set of conceptual challenges that apply broadly to global kernel-level name-spaces that require communication for coherence. All kernel-level to user-level communication is required to be stateless in the kernel for the sake of reliability, and happens through an interface to the Linux VFS (Virtual Filesystem Switch).

Unified /proc Filesystem

While the GPID extension is sufficient for cluster-wide control and signally of processes, it's of little use with a global view of the processes. To this end, work is underway on a mechanism that will allow unmodified versions of standard UNIX process utilities (ps, top, etc) to work across a Beowulf cluster.

Linux has an advanced implementation of the */proc* [9] pseudo-filesystem. System and process information is presented in the form of a "filesystem" generated in real-time by the kernel. This scheme was originated as a interface for debuggers, and was popularized with the Plan 9 [10] operating system. A basic */proc* presents a subdirectory for each process on the local processor. The Linux implementation extends */proc* to present almost all system information in this format.

The */proc* filesystem is used by all of the common system monitoring tools on Linux, e.g. 'ps' 'top'. Most of these tools work unchanged with the conceptually simple step of combining the */proc* directories of the cluster using the existing NFS capabilities. The implementation complexity comes from handling pseudo-files that have no length until they are read, and there is performance impact from gathering fresh directory information with each request.

Programming Models

There are several distributed application programming environments available on Beowulf. The most commonly used are the PVM [12] and MPI [13] environments, with BSP [15] also available and used. A distributed shared memory package is planned.

PVM and MPI

While there are multiple, disparate programming paradigms, the most widely used is message passing. Even when hardware systems support shared memory mechanisms, message passing is still often used by application programmers for portability. Beowulf support the popular PVM [12] and MPI [13] programming models with a very slightly modified Oak Ridge PVM package and an unchanged Ohio State LAM MPI[11] package.

BSP

Traditional UNIX kernels make a fundamental distinction between namespaces ; UNIX kernels manage them by casting them as either explicit (system call / filesystem interface) or implicit (memory interface, mmap()). While it is far simpler for the systems programmer to provide coherency across namespaces managed explicitly, these sorts of mechanism can be the bane of application programmers. Using explicit message passing techniques to parallelize serial applications or port parallel codes developed for shared-memory architectures can be both non-obvious and tedious. Time to port and develop must be considered as part of any net productivity gain from a platform. In many cases, BSP (Bulk Synchronous Parallel) libraries provide active messaging to eliminate server-side message passing application code. However, client-side remote memory references must still be explicit. This removes the local/remote address space transparency that is frequently convenient and complicates the referencing (indirect or otherwise) of distributed objects.

Distributed Shared Memory

The Linux kernel provides a VFS-like interface into the virtual memory system. This makes it simpler to add transparent distributed backends to implicitly managed namespaces. Page-based systems can be created that allow the entire memory of a cluster to be accessed either almost or completely transparently.

An additional environment being added to the Beowulf packages is page-based Network Virtual Memory (NVM), also known as Distributed Shared Memory (DSM). The initial implementation is based on the ZOUNDS (Zero Overhead Unified Network DSM System) system from Sarnoff [14]. Page-based distributed shared memory uses the virtual memory hardware of the processor and a software-enforced ownership and consistency policy to give the illusion of a memory region shared among processes running an application.

A more conventional DSM-NVM implementation is planned, along with support for a network memory server.

Parallel Filesystem

Beowulf systems can take advantage a number of libraries written to provide parallel filesystem interfaces to Networks of Workstations. Jovian (University of Maryland at College Park), PIOUS, and the Parallel Virtual File System (PVFS) developed by Clemson in concert with NASA Regional Data Centers have been run on Beowulf. Portable Parallel File System (PPFS) from UIUC runs on systems similar to Beowulf.

MPI-IO is expected to become core software for new applications. However, PVFS alone will automatically enable several key NASA applications. PPFS is being developed alongside the Scalable IO Initiative; support provides contact with the multi-agency, multi-vendor initiative to address the increasing demands on IO subsystems within the high performance computing community. Jovian support makes additional legacy applications (e.g. Pathfinder[18]) available to Beowulf users.

5. Applications Scaling and Performance

The Beowulf Architecture provides sites with the flexibility to build machines tuned to the particular demands of their application. However, clusters built to date have retained the balance of the original while delivering high performance to the application. There are a number of applications which port to Beowulf with little more than a recompile – MPI, PVM, BSP and many other common libraries are all available. One important example application is Warren-Salmon gravitational N-body codes.

N-body

N-body codes have always been a target application for Beowulf-class machines. In the past, Beowulf machines at NASA Goddard were used as proxys for larger machines. Codes were developed and debugged on a Beowulf and migrated to large distributed-memory machines (T3D, Paragon, CM5). Recently, however, code has migrated the other way. Michael Warren and John Salmon, recipients of the 1992 Gordon Bell Prize for performance in large-scale scientific computing, have migrated their highly optimized

N-body codes from the Thinking Machines CM5 onto the clusters installed at both institutions. The port was reported to take “man-minutes” — a simple recompile of the existing code that worked without modification. Rewardingly, each cluster sustains over 1.2 GigaFlops on 10 million body problem.

The direct solution to the equations of force for a system of gravitationally interacting particles requires an $O(N^2)$ calculation. Tree codes are a collection of algorithms which find approximate solutions by exploiting the naturally locality in the system. Particle information is sorted into a tree-based spatial hierarchy; an intermediate node in the tree is responsible for storing average quantities (e.g. mass, center of mass, and high order moments of the mass distribution) for the particles stored at the leaf nodes “underneath” that node. To approximate the force on each body, the algorithm searches the tree and uses the information stored at the intermediate nodes whenever it satisfies certain approximation criteria. This results in $O(N \log N)$ scaling, but presents difficulties for the parallel programmer; the tree search is not known *a priori* for a given particle, the tree is unstructured and frequent indirect addressing is required.

The Warren-Salmon algorithm builds the tree data structure differently from standard Barnes-Hut algorithm by decomposing 3-space into Morton–order 1-space. In doing so it is able to control the “shape” of the tree to match the tree (and the induced communication patterns) to the granularity of the particular distributed-memory constraints.

6. Discussion and Conclusions

Advantages

The Beowulf system has been a success on several fronts. Beowulf systems have been been constructed at other sites both for academic and scientific uses. Its price/performance compares very favorably to other modern parallel architectures; a Beowulf Pile-of-PCs has equaled the performance of IBM SP2s with comparable nodes at less than one tenth the price to the end user for important problems. The absolute performance has surpassed 1 GFLOPS on a non-trivial application — a goal established at the beginning of the project. Some of system software developed as part of this project has been widely distributed and used. And a site can retain the low cost of Beowulf and trade-off absolute performance for increased parallelism (by substituting larger numbers of less expensive processors) for pedagogical and instructional uses.

Limitations

The Pile-of-PC methodology is still experimental, and does not match all of the valuable services provided by computer vendors. It is not for everyone. Rather, it is an emerging opportunity in the high performance computing field and complements rather than competes with the HPC industry commercial products.

The opportunities made available through this approach provides a lower entry level price to parallel computing, thus increasing the parallel processing user community earlier and helping build such a market. These initial entry level users are more likely to acquire higher grade vendor supplied systems once they have experienced the advantages and overcome the psychological barriers to parallel computing. Vendors of traditional high performance computers provide full support and maintenance. Users of Piles-of-PCs must provide that on their own. While there are many places where this is easy to do, such as academic or national laboratories, there are many markets where this would not work well, such as the business or banking industries.

Future Work

While much has been accomplished, much remains that can be done. Several projects, as mentioned throughout this paper, are in progress.

Our goal is to make assembling a cluster an easy afternoon's activity, with no expert intervention required. That is not yet the case. Work remains to be done on writing instructions sufficiently clear, and preparing a robust packaged distribution that avoids most pitfalls.

References

[1] K. Castagnera, D. Cheng, R. Fatoohi, et al. "Clustered Workstations and their Potential Role as High Speed Compute Processors," *NAS Computational Services Technical Report RNS-94-003,* NAS Systems Division, NASA Ames Research Center, April 1994.

[2] Linux Documentation Project, Accessible on the Internet at World Wide Web URL `http://sunsite.unc.edu/mdw/linux.html`.

[3] GNU General Public License, Version 2, June 1991, Free Software Foundation, Inc., 675 Massachusetts Ave, Cambridge MA 02139.

[4] T. Sterling, D. Becker, D. Savarese, et al. "BEOWULF: A Parallel Workstation for Scientific Computation," *Proceedings of the 1995 International Conference on Parallel Processing (ICPP),* August 1995, Vol. 1, pp. 11-14.

[5] T. Sterling, D. Savarese, D. Becker, B. Fryxell, and K. Olson, "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation," *Proceedings of the Fourth IEEE Symposium on High Performance Distributed Computing (HPDC),* August 1995, pp. 23-30.

[6] C. Reschke, T. Sterling, D. Ridge, D. Savarese, and D. Becker. "A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation,", *Proceedings of the Fifth IEEE Symposium on High Performance Distributed Computing (HPDC),* August 1996.

[7] Red Hat Software, Inc. Homepage, Accessible on the Internet at World Wide Web URL `http://www.redhat.com`.

[8] A. Tridgell, P. Mackerras, D. Sitsky and D. Walsh, "AP/Linux - A Modern OS for the AP1000+" Austrailian National University Technical Report; `http://cap.anu.edu.au/cap/projects/-linux/index.html`.

[9] T.J. Killian, "Processes as Files", *USENIX Summer Conference Proceedings*, June 1984.

[10] R. Pike, D. Presooto, K. Thompson, and H. Trickey, "Plan 9 from Bell Labs", Proceedings of the Summer 1990 UK Unix User Group Conference, July 1990, pp. 1-9.

[11] LAM / MPI Parallel Computing Homepage, Accessible on the Internet at World Wide Web URL `http://www.osc.edu/Lam/lam.html`.

[12] V. Sunderam, "PVM: A Framework for Parallel Distributed Computing," *Concurrency: Practice and Experience,* December 1990, pp. 315-339.

[13] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, "MPI: The Complete Reference,", The MIT Press, Cambridge, Massachusetts, 1996.

[14] R. G. Minnich, "ZOUNDS: A Zero Overhead Unifed Network DSM System" Sarnoff Technical Report; `ftp://ftp.sarnoff.com/pub/mnfs/www/-docs/cluster.html`.

[15] "The Bulk Synchronous Parallel (BSP) Computing Model Worldwide Homepage" `http://www.bsp-worldwide.org/`.

[16] J. E. Barnes and P. Hut, *Nature, 324*, 1986, p. 446.

[17] A. Heirich, and J. Arvo, "Scalable Monte Carlo Image Synthesis", To appear in *Parallel Computing* 1997.

[18] R. Bennett, K. Bryant, A. Sussman, R. Das, J. Saltz, "Jovian: A Framework for Optimizing Parallel I/O", Proceedings of the 1994 Scalable Parallel Libraries Conference.