ELSEVIER Parallel Computing 27 (2001) 91–108

PARALLEL COMPUTING

www.elsevier.com/locate/parco

# The performance of a supercomputer built with commodity components

Yuefan Deng [1], Alex Korobka

*Center for Scientific Computing, State University of New York, Stony Brook, NY 11794-3600, USA*

Received 6 December 1999; received in revised form 12 June 2000

**Abstract**

We built a supercomputer called Galaxy by connecting Intel Pentium-based computer nodes with Fast and Gigabit Ethernet switches. Each node has two processors at clock speeds varying from 300 to 600 MHz, up to 512 MB of memory, and small 2 Gb local disk. All nodes run the standard RedHat Linux and inter-node communication is handled by a message passing interface called MPI. Local tools are written to visualize the system performance and to balance loads. We have benchmarked a sub-Galaxy with 72 processors by NAS and Parallel LINPACK benchmark suites. We achieved 16.9 Gflops in a standard single precision LU decomposition for $46848 \times 46848$ matrix parallel LINPACK benchmark. A Galaxy with 128 processors costs approximately \$250 000 and it delivers 40 Gflops of performance. This leads to a cost-performance ratio of 160 Kflops-per-dollar, which is to improve further due to increase in processor speeds and network bandwidth at similar cost. Our final system with 512 processors is expected to reach several Tflops. This article first describes the Galaxy architectural details, and then present and analyze its performance in terms of floating point number crunching, network bandwidth, and IO throughput. © 2001 Elsevier Science B.V. All rights reserved.

*Keywords:* Intel Pentium-based cluster; Network computing; NAS and LINPACK; Build your own supercomputer

## 1. Introduction

Building a computer from the commercial off-the-shelf (COTS) parts for scientific computing has come a long way. In the early 1980s, particle physicists at Caltech

[1] On leave at IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.
*E-mail addresses:* deng@ams.sunysb.edu (Y. Deng), korobka@ams.sunysb.edu (A. Korobka).

0167-8191/01/\$ - see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0167-8191(00)00090-9

University [1], Columbia [2], and Fermilab designed and constructed special purpose supercomputers for gauge theory calculations. Astrophysicists at University of Tokyo [3] build a machine called MD GRAPE for *N*-body problems used in astrophysics and molecular dynamics simulations. Recently, several other computational science groups are constructing supercomputers using Pentium or Alpha processors and Ethernet, ATM [10], SCI [11], QsNet [21], Myrinet [22] or other less known networking devices to connect these processors. These groups include Los Alamos' Avalon [4], Duke's Brahma [5], NIH's Lobos [6], Parnass2 in University Bonn [7], and Stony Brook's Galaxy [8]. These projects have a common origin in the pioneering work of Beowulf project [9] at NASA, but despite the similarities, each of them possesses its own unique features.

The basic computing unit in all such systems is the computing node which consists of several (one or two or four) microprocessors such as Intel Pentium and DEC Alpha, 1 Gb of memory, network interface card (NIC), and several Gbs of hard disk, on one motherboard with bus that can support symmetrical multiple processors. These nodes are connected by a mixture of several different network devices such as the Fast Ethernet switches capable of bandwidth up to 100 Mbps, the new Gigabit Ethernet switches with bandwidth up to 1 Gbps, or the ATM, or the proprietary high-bandwidth, low-latency interconnect schemes like SCI or Myrinet. The node operating system is usually the multiprocessor-enabled Linux and these nodes are coupled in software by MPI or PVM with some local additions to the above for convenient and efficient system utilization and management.

Galaxy is a scalable heterogeneous architecture built with easily upgradable modular components. The network has a multistar topology with all nodes and servers connected to a cluster of Fast Ethernet or Gigabit switches.

The Galaxy project was initiated in 1997 because of the cost-effectiveness, flexibility in configuring the system for particular applications, and opportunity to learn hands-on computer assembling. These reasons are common for many projects and perspective groups may be interested in knowing the pains and gains in building a computer for application projects. In this report, we will document our Galaxy project in the following sections, we will discuss the Galaxy hardware, software, floating-point operation and communication benchmarking results.

A system like Galaxy with 512 processors is expected to deliver 100 Gflops of performance at a cost of $1M, yielding a cost-performance ratio of up to 100 Kflops-per-dollar. In comparison to the current industrial cost-performance standard of 10–50 Kflops-per-dollar for systems such as IBM SP/2, SGI MIP Clusters, DEC Alpha Clusters, we gain nearly one order of magnitude in cost-effectiveness in the list price of the purchase. In addition, lower costs of software and hardware maintenance and reuse of depreciated components make Galaxy-like systems certainly more economical.

The Galaxy is running around the clock and it is a major resource for faculty, postdocs, graduate students at Stony Brook to carry out their research projects. These projects include study of three-dimensional fluid interface instability using the front-tracking method [12], thin-film deposition using molecular dynamics and continuum methods [13], gene circuits using simulated annealing [14], and DNA–protein interaction [15], as well as granular materials, among several other projects.

## 2. Hardware

The main components of the Galaxy include front-end node, service nodes, compute nodes, switches and parallel IO sub-system. Front-end node provides an interface for users to utilize and control the entire system, to compile and debug programs and to submit and monitor jobs. Service nodes house network file systems and authentication daemons. Compute nodes follow instructions of the service nodes to perform the ultimate number crunching. The parallel IO system, with channels open to several clusters of processors, provides massive storage and high-speed access data between the storage and the compute nodes. Non-blocking switch glues all these components together, making Galaxy a true parallel system.

Connecting this large number of components for delivering ideal performance for a variety of applications is an intricate optimization problem. Fig. 1 is a schematic

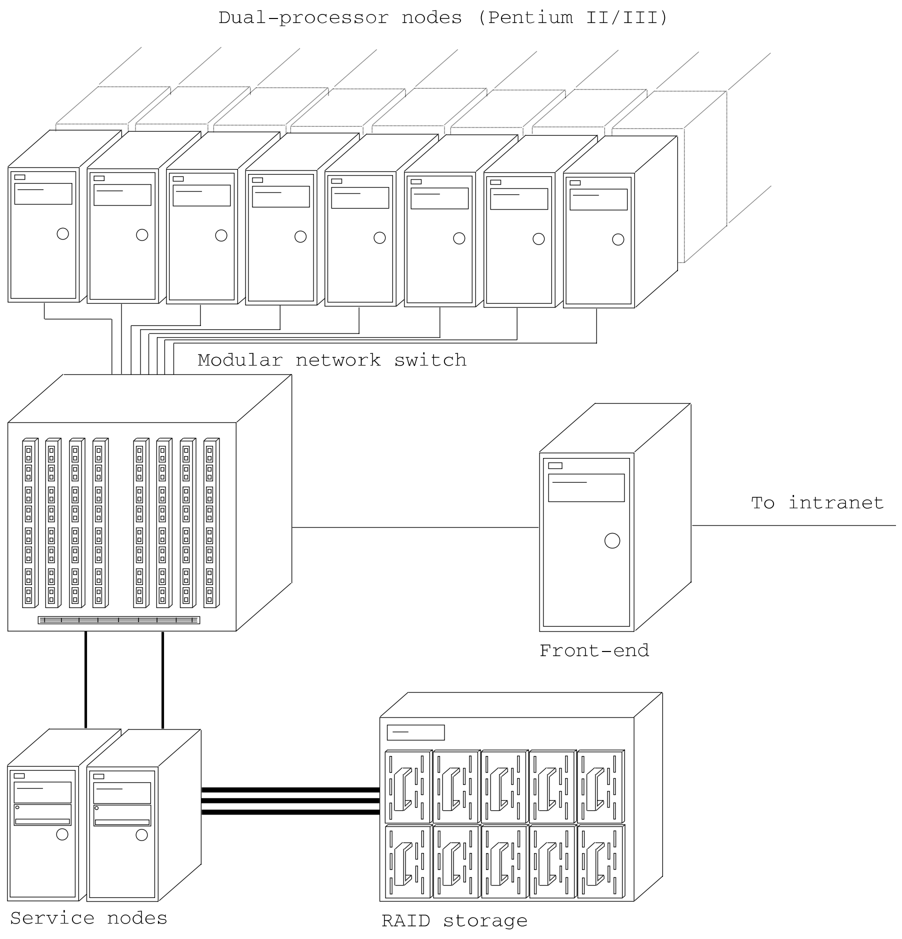

Fig. 1. The schematic view of the Stony Brook Galaxy computer.

view of the Galaxy computer. We have reconfigured the Galaxy according to several different criteria and we concluded that we need to develop software for this optimization.

## 2.1. The front end

The front end is the only link of the Galaxy computer to the users. It provides services for user to log in, compile, debug, submit jobs to the batch queue and to analyze the results. In order to handle concurrent demands of up to a hundred of users we install large amount of memory, and the maximum number of processors as allowed by the system architecture.

Our front end is a four-processor Pentium Pro system equipped with 512 MB of main memory on the American Megatrends Goliath motherboard [16], powered by dual 400 W hot-swappable units. External power is routed through an uninterruptable power supply (UPS) to prevent interference and variations in voltage. The board complies with the Intel Multi-Processor Specification (MPS) version 1.4. The Goliath motherboard has eight 168-Pin Dual In line Memory Module (DIMM) slots. Memory must be buffered 3.3 V fast page Error Correcting (ECC) or Extended Data-Out (EDO) DIMMs. The node uses eight $8\ \mathrm{M} \times 7260\ \mathrm{ns}$ DIMMS. The node has two Intel EtherExpress Pro [17] Fast Ethernet network interface cards, one of which is used for connection to the campus network for users, the other connects to the local Galaxy network. Local IO is handled by a pair of Mylex BT-958 SCSI controllers [18]. Front end maintains internal Network Information Service (NIS) and Domain Name Service (DNS) databases and exports the disk partition that houses mail folders to the rest of the Galaxy cluster.

## 2.2. The service nodes

Initially, in addition to hosting the user environment, the front end ran the entire set of system services such as NIS, NFS, WWW, and Sendmail. As the number of users increased we found that the NFS server daemon was often unable to supply compute nodes with data when the front end was busy performing local IO. Also, whenever the front end crashed from overheating, defective hardware or bugs in the kernel, jobs running on compute nodes were often terminated after their attempts to contact NFS timed out. Therefore we decided to offload the front end by shifting NFS and NIS services to the special service nodes to which regular users do not need direct access.

Current setup has two service nodes. They are dual-processor Pentium II systems based on the SuperMicro P6DBE motherboard with IntraServer ITI-3280 dual-port Ultra Wide SCSI cards connected to the disk array (170 Gb total) and NetGear GA620 Gigabit Ethernet uplinks to the internal Galaxy network. Service nodes have CD-ROM, CD-RW drives and an external tape drive for backup. Service nodes mount disk partitions and export them to the rest of the cluster through NFS while serving as secondary NIS servers.

## *2.3. The compute nodes*

The computational power of the Galaxy comes from the compute nodes. All of these nodes run a local copy of the multiprocessor-enabled Linux operating system. All nodes, assembled with dual Pentium II/III processors, have ATX-compatible system boards with four 168-pin SDRAM slots. We use 128 MB memory modules that give us a maximum of 512 MB of memory per node. Each node has a local 2.2 Gb hard drive partitioned with 128 MB as swap space, 512 MB as scratch space and 1.5 Gb for the operating system. The main criteria in choosing these components is to maximize the cost/performance ratio while prolonging the mean time between hardware failures. For example, we use very inexpensive low-end ISA video cards because they are needed only for certain maintenance operations. On the other hand, when the power supply fails it can damage other parts of the system, thus we use high-quality computer cases.

Another important issue in selection of the hardware for Galaxy is the availability of the stable drivers for the Linux operating system. First generation systems are built with motherboards that have an integrated onboad SCSI controller. Linux OS only recently became widely supported by the hardware vendors. At that time (1997), Linux did not have solid support for the Adaptec 2940 chipset so the second generation had a different board with add-on Buslogic SCSI card.

By 1999 Adaptec drivers stabilized enough to be used in cluster environments therefore third generation systems feature onboard SCSI along with faster processors and improved memory bus.

Fourth generation nodes will be mounted in the high density cabinet. In order to satisfy constraints on space they will be based on Intel N440BX boards with completely integrated peripherals.

The changes in Galaxy's compute nodes are given in Table 1.

Table 1
Components of the Galaxy's compute nodes

| | 1st Generation | 2nd Generation | 3rd Generation |
|---|---|---|---|
| Time | July 1997–January 1998 | January 1998–January 1999 | January 1999–July 1999 |
| CPU | Two Pentium Pro 200 MHz | Two Pentium II 300 MHz | Two Pentium II 400 MHz |
| Board | Intel PR440FX | SuperMicro P6DLE | AsusTek P2B-DS |
| Memory | 256 MB SDRAM 66 MHz | 256–512 MB SDRAM 66 MHz | 512 MB SDRAM 100 MHz |
| Network | EtherExpress Pro 100 | EtherExpress Pro 100 | EtherExpress Pro 100+ |
| Disk | 2.2 Gb Western Digital | 2.2 Gb Quantum Atlas II | 2.2 Gb Seagate Barracuda |
| SCSI | Adaptec 2940 (onboard) | Mylex BT-958 | Adaptec 2940 (onboard) |
| Case | En-Lite ATX 7230 250 W | En-Lite ATX 7230 250 W | Asus ElanVital T-5 250 W |

### 2.4. The network

Galaxy's internal network uses a modular BigIron 8000 switch [23] with 128 Fast Ethernet ports and 8 Gigabit Ethernet ports. Fast Ethernet is a mature technology and there are many network interface cards that offer higher than 90 Mbps real TCP/IP bandwidth. We use Intel EtherExpress Pro 100 NICs on compute nodes. The main advantage is that the majority of the integrated system boards for the Intel platform use this card as a built-in Fast Ethernet device.

Given the limited choice of the Gigabit network interface cards with Linux drivers and excessive cost, only small percentage of the Galaxy network is handled by Gigabit Ethernet [27]. The Gigabit Ethernet with second-generation NICs can achieve an aggregate bandwidth of 300 Mbps, i.e 30% of the expected peak throughput. Switching to Gigabit cards does little to improve the latency of TCP/IP data transfer because in this case the bottleneck is in the kernel network stack. MPI versions that use faster communication layers, like Virtual Interface Architecture [28], are just becoming available in beta form. The design of the system allows us to take full advantage of the Gigabit Ethernet later when its bandwidth reaches its expected values at a comparable cost-bandwidth ratio to the Fast Ethernet today. At this point, gigabit ports are utilized for the majority of NFS and NIS traffic on service nodes. The current configuration with two service nodes can be easily expanded to scale up the IO system as required by the growing cluster.

Several other network solutions such as the well-received Myrinet [22,24] and ATM [10] are also considered by other similar projects [4–6]. Myrinet provides very high bandwidth with low latency (around microseconds). Each node is connected to the central Myrinet switch through its own PCI card, besides economical concerns, the technical limitation at this time (1999) that the largest Myrinet switch has only 16 ports constraints its acceptability.

### 2.5. Storage

In addition to a small local disk (2.2 Gb) on each compute node for node-specific OS files and occasional needs of paging, a more centralized yet parallel IO system which has multiple direct communication channels to a few (ranging from 2 to 8) service nodes is installed. This is where user home directories, common system libraries and development tools are located. We chose not to use distributed storage because of the redundancy and the ease of management provided by hardware RAID.

At the center of the parallel IO system is the StorComp RAID-7 R3X unit [19]. This system provides a real-time asynchronous management of data transfers for optimal IO rates. Peak rate is 40 Mbps on each of eight Ultra Wide SCSI channels. Data are organized to flow between a microprocessor-controlled smart intermediate storage buffers and the component disk drives, completely independent of any hosts. This asynchronous management of data transfers greatly mitigates the write bottleneck that persists in other RAID systems. This architecture allows for optimization of host demands, drive resources, and scalability.

At this time local disks are used mostly for temporary storage. However, as the cluster grows we may deploy the Parallel Virtual File System [20] to gain up to 64 Gb of additional space from unused 512 MB partitions of each local disk.

## 3. Software

Compute nodes carry minimal software needed to boot the system and maintain it. As in the case with user home directories, all user libraries and tools are located on the RAID partitions exported by service nodes. This guarantees identical development and execution environments. System software consists of three main parts – operating system, communication libraries, and job control system. All of them are freely available and, although almost all of them have commercial analogs, the preference is to use Open Source packages unless the performance of the commercial analog justifies its price like Fortran compiler in our system.

### *3.1. Operating systems*

Linux [29] is the operating system for the compute nodes as well as the service nodes. Galaxy uses RedHat 5.2 distribution on all systems. New compute nodes are installed by booting them from a custom boot disk that formats the hard drive, connects to the LAN, mounts an NFS volume, and unpacks an archive with the image of the configured node. After that, software maintenance (mostly Linux kernel upgrades) is done remotely. Currently, version 2.2 of the Linux kernel is the stable branch and version 2.3 is the development branch. The latter has more fine-grained Symmetric MultiProcessing (SMP) subsystem which provides substantial improvements over the stable release at the cost of shorter average uptime between crashes. The next stable release based on version 2.3 is expected in the late 1999. Service nodes run a standard stable release. Compute nodes run a customized version of the kernel with enlarged network device buffers and explicitly disabled delayed-acknowledgment feature in the TCP/IP stack for TCP_NODELAY connections. These changes improve the latency and the stability of connections during heavy network IO.

RedHat ships with most standard GNU development tools (gcc, make, gdb, ld, etc.). In addition, for visual debugging we installed DDD [41] – a free graphic shell for the GNU symbolic debugger. The lack of an optimizing Fortran compiler presents a problem since the freely available GNU Fortran compiler (g77) is still under development. Therefore we purchased PGI Workstation development kit [42] from the Portland Group. It includes optimizing C/C++/Fortran77/Fortran90 compilers (pgcc, pgCC, pgf77, pgf90) with advanced vectorization capabilities, profiler and visual debugger. We observed up to 20% performance improvements in code recompiled with PGI tools.

### 3.2. Message-passing interface (MPI)

MPI [30] was designed by a broadly based committee of vendors, implementors and users. As documentation for MPI states, the goal of the package is to develop a widely used practical, portable, efficient and flexible standard for writing message-passing programs. There are several free implementations of this standard for distributed memory systems on a variety of platforms. Most comprehensive of them are MPICH [32], developed in Argonne National Laboratory and Mississippi State University, and LAM [33], maintained in University of Notre Dame. Both implementations are available on Galaxy. Most Galaxy users prefer MPICH which is faster than LAM on our system.

The standard MPICH distribution running on Fast Ethernet network proved sufficient for our initial needs. Porting applications from Intel Paragon parallel supercomputer to MPI does not take very long since the MPI application programming interface (API) is functionally very similar to Intel's *nx* library [31]. To facilitate compilation of MPI code with PGI tools we have a version of MPICH library recompiled with pgcc. Although MPICH supports communication through the shared memory on SMP systems we achieved better results by using a uniform configuration with TCP/IP as the communication layer rather than with mixed shared memory-TCP/IP configuration. Since MPI through TCP/IP is not very efficient, most proprietary systems provide either a complete MPI implementation or a driver for one of the MPI packages (MPICH, LAM) that bypasses TCP/IP stack. In the last few years Virtual Interface Architecture (VIA) gained prominence as an attempt to establish an industry standard for communication in clusters of servers and workstations in system area networks with emphasis on performance and scalability. Linux implementation (M-VIA) has been developed by National Energy Research Scientific Computing Center [34] with the VIA driver for MPICH available in the near future. M-VIA coexists with TCP/IP, it does not offer a big increase in bandwidth but it is capable of substantially improving the latency of communication by eliminating the bottlenecks of TCP/IP stack. The current version of M-VIA supports Intel EtherExpress Pro 10/100 NIC and several other chipsets making it possible to do a smooth transition to MPICH based on VIA when it becomes available. Combining these components will be the next step in improving the Galaxy communication capabilities. This approach avoids the laboring process of porting user code to the new API.

### 3.3. Job control

Job control on Galaxy is performed through the Portable Batch System [35] (PBS) which was developed in NASA Ames Research Center to provide a centralized interface to the large-scale computing resources. It is compliant with POSIX 1003.2d standards for shells, utilities, and batch environments. PBS installation consists of a server process and a job scheduler on the front end and a job monitor on each of the compute nodes. Node allocation requests are routed through the job scheduler that executes them when sufficient resources become available. This allows users to submit jobs to the batch queue without checking the number of free nodes on the system. In addition to the batch queue daemon, a general purpose resource monitor that reports miscellaneous OS parameters, like system load, top 2 CPU consumers, free memory, board temperature and so on, is available. In its turn, the front end runs a proxy server for this information and a java applet is implemented to view these parameters remotely. MPICH *mpirun* startup script needed several modifications to make use of compute nodes automatically allocated by PBS. We altered the default mode of operation of *mpirun* to create a PBS script and submit it to the common queue.

## 4. System performance

There are several standard benchmark suites that provide a way to compare the performance of different architectures on different problems. Galaxy performs quite well on code with coarse to medium grained parallelism. Finer grained parallel problems require low latency transport layer do not scale as well. This is a common deficiency of clusters that use Fast Ethernet for communication.

### *4.1. NAS benchmarks*

The NAS Parallel Benchmarks [36] implement in Fortran algorithms found computationally intensive aero-physics applications (kernels and simulated computations fluid dynamics) to measure the performance of parallel computer systems.

The following tables show the results of the standard NAS benchmarks on the third-generation Galaxy nodes (400 MHz Pentium II). The numbers in each entry denote per-processor performance in Mflops for problems defined in Table 2. Most Class C and Class B benchmarks for small number of nodes, particularly FT, require more memory than available on our system. We chose not to run them since such runs would take too long without producing new information. The performance on small number of nodes is well represented by the Class A results. Tables 3–5 do not include the results for EP benchmark. EP produces 0.06 Mflops/processor and scales with 100% efficiency in all three classes. The performance of the system may vary

Table 2
NAS parallel benchmarks problem definition

| Benchmark code | Class A | Class B | Class C |
|---|---|---|---|
| Embarrassingly parallel (EP) | $2^{28}$ | $2^{30}$ | $2^{32}$ |
| Multigrid (MG) | $256^3$ | $256^3$ | $512^3$ |
| Conjugate gradient (CG) | 14 000 | 75 000 | 150 000 |
| 3D FFT PDE (FT) | $256^2 \times 128$ | $512 \times 256^2$ | $512^3$ |
| Integer sort (IS) | $2^{23}$ | $2^{25}$ | $2^{27}$ |
| LU solver (LU) | $64^3$ | $102^3$ | $162^3$ |
| Pentadiagonal solver (SP) | $64^3$ | $102^3$ | $162^3$ |
| Block tridiagonal solver (BT) | $64^3$ | $102^3$ | $162^3$ |

Table 3
IS, LU, MG, FT, and CG benchmark performance (Mflops/processor)

| # Processors | | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|---|---|
| IS | A | 4.76 | 3.00 | 1.07 | 0.59 | 0.34 | 0.38 | 0.08 |
| | B | 4.56 | 2.84 | 0.89 | 0.56 | 0.40 | 0.23 | 0.34 |
| | C | | | 0.74 | 0.52 | 0.33 | 0.31 | 0.12 |
| LU | A | 76.63 | 63.39 | 59.54 | 57.41 | 53.48 | 39.74 | 29.93 |
| | B | | 61.47 | 57.84 | 52.88 | 45.70 | 53.32 | 42.55 |
| | C | | | | 55.31 | 44.31 | 47.44 | 54.05 |
| MG | A | 42.75 | 36.36 | 29.26 | 31.44 | 23.43 | 17.99 | 14.20 |
| | B | | 38.43 | 30.75 | 33.86 | 25.09 | 19.49 | 15.06 |
| | C | | | | 31.23 | 26.31 | 28.15 | |
| FT | A | 16.75 | 14.82 | 11.47 | 9.13 | 7.78 | 7.27 | 9.01 |
| | B | | | 10.58 | 10.63 | 9.09 | 7.38 | |
| | C | | | | | | | 8.12 |
| CG | A | 32.61 | 27.49 | 16.08 | 12.48 | 5.25 | 3.07 | 0.70 |
| | B | | | 15.37 | 12.98 | 7.92 | 6.24 | 3.48 |
| | C | | | | | | 7.75 | 4.54 |

Table 4
BT and SP benchmark performance (Mflops/processor)

| # Processors | | 1 | 4 | 9 | 16 | 36 | 64 |
|---|---|---|---|---|---|---|---|
| BT | A | 65.64 | 44.75 | 39.89 | 35.06 | 32.00 | 22.69 |
| | B | | | 42.27 | 39.45 | 35.29 | 33.20 |
| | C | | | | | 39.40 | 35.97 |
| SP | A | 47.50 | 29.08 | 24.79 | 21.11 | 13.04 | 8.37 |
| | B | | 30.86 | 27.01 | 22.83 | 18.52 | 13.18 |
| | C | | | | 25.09 | 21.34 | 19.51 |

Table 5
Best entire system and normalized per-processor performance in Mflops on LINPACK LU factorization

| CPU | Matrix | Block | Mesh | Total | Per-proc |
|---|---|---|---|---|---|
| 1 | $5430 \times 5430$ | 64 | $1 \times 1$ | 301.1 | 301.1 |
| 2 | $7680 \times 7680$ | 32 | $1 \times 2$ | 567.7 | 283.8 |
| 8 | $15360 \times 15630$ | 32 | $2 \times 4$ | 2077.4 | 258.7 |
| 18 | $23616 \times 23616$ | 32 | $3 \times 6$ | 4442.5 | 246.8 |
| 32 | $30720 \times 30720$ | 32 | $4 \times 8$ | 7773.0 | 242.9 |
| 50 | $36800 \times 36800$ | 32 | $5 \times 10$ | 11874.2 | 237.5 |
| 72 | $46848 \times 46848$ | 64 | $6 \times 12$ | 16919.3 | 234.9 |

depending on the version of the Linux kernel. Results presented here were obtained on the version 2.3.14 which was the most recent development release at the time. Compute nodes on Galaxy have two CPUs, thus all multiprocessor runs were made in SMP mode with two processes per node. It should be noted that the SMP performance is about 10% lower than in the case with a single process per node.

The scalability of NAS benchmarks is presented in Fig. 2 showing the parallel speedup for all runs excluding IS. Since integer sort routine transmits short messages using collective communication MPI functions it is highly dependent on the message

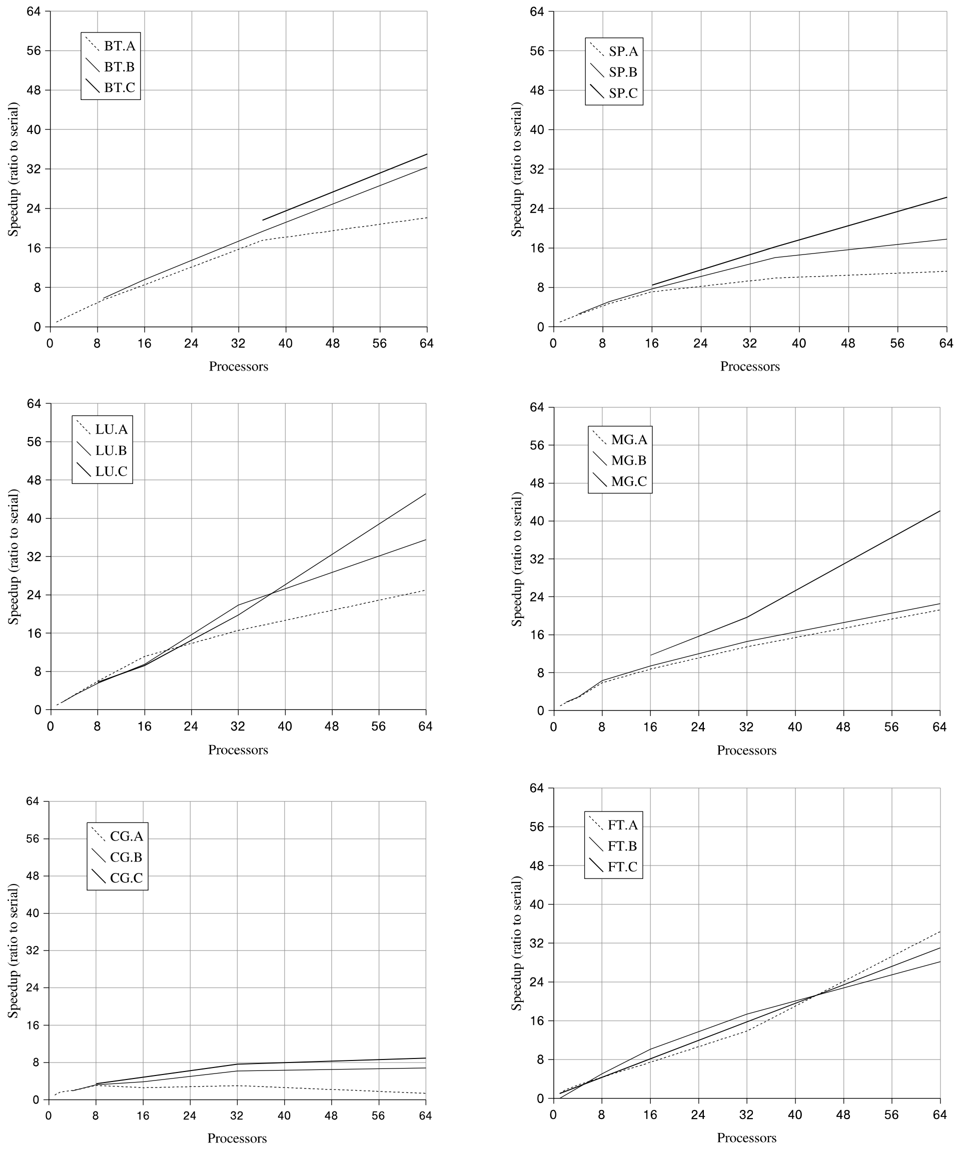

Fig. 2. Parallel speedup for NAS problems.

bandwidth. This is not efficient with TCP/IP over Fast Ethernet and, as a result, IS benchmark scales poorly.

### *4.2. Parallel LINPACK benchmarks*

This variation of the standard LINPACK benchmark [37] is used to rate the performance of parallel computers (see Fig. 3). This version utilizes high-performance parallel linear algebra routines from SCAlable Linear Algebra PACKage (SCALAPACK). The next table presents results of running the single precision version on third generation Galaxy nodes. The code was compiled by Portland Group compiler version 3.0 with optimization settings -fast -pc 64. The executable was linked to the version 1.1n of the ASCI Red Pentium BLAS library [40] that contains routines from the Intel Math Kernel library optimized for processors from the Pentium Pro family. Problem sizes were adjusted to achieve the maximum utilization of memory on the compute nodes while avoiding swapping to the hard disk. Multiprocessor cases were ran two processes per node.

The performance scales almost linearly. After the first processor, every additional one contributes, on the average, 78% of its performance in the uniprocessor mode.

### *4.3. Communication bandwidth and latency*

Data pathway on Galaxy flows through several different hardware and communication layers. The simplest way is through the loopback virtual device that is used for TCP/IP communication between applications running on the same machine. In ideal situation this is equivalent to a memory copy. However, the architecture of TCP/IP layer necessitates several copy operations between auxiliary buffers with additional overhead going to the maintenance of complex control structures. Thus, the expected result is far from the maximum sustainable memory bandwidth of

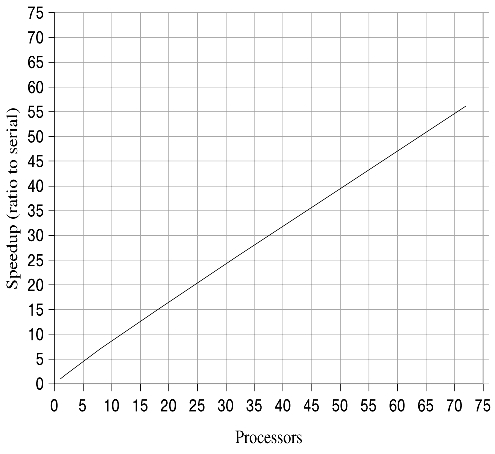

Fig. 3. Speedup for LINPACK parallel benchmark.

300 MB/s as measured by the STREAM benchmark [38]. The bandwidth is even lower for the MPI version of the benchmark since the current MPI driver for clusters of workstations is built on top of TCP/IP.

In order to evaluate the performance of the Galaxy network we used freely available program NetPIPE [39] written in Iowa State University. It measures the throughput of the TCP/IP stream for the range of message sizes. The results for the loopback device are given in Fig. 4.

Same pattern holds for communication through the network. TCP/IP throughput between two third-generation Galaxy nodes connected with Fast Ethernet reaches 91% of the nominal 100 Mbps. The figure for MPI is 82%. First generation Packet Engines GNIC-I [25] adapters are not nearly as efficient with 16% of the peak performance for TCP/IP throughput and 15% for MPI. Second generation Gigabit Alteon AceNIC [26] does not work very well when used in default configuration. However, after latency tuning it shows significantly better performance with 290 Mbps for TCP/IP and 200 Mbps for MPI. Overall, its performance is much better than the Fast Ethernet but still far from expected peak of 1000 Mbps. Since most Gigabit cards support 64-bit PCI bus they may approach this value when used in systems based on 64-bit architecture such as Compaq Alpha or Intel Itanium.

From the Fig. 5, it is clear that the default TCP/IP settings prove to be particularly inefficient for 100–1000 byte sized MPI messages on AceNIC. This is the example of the problem encountered by the other Linux clustering projects, such as the Coral Project in ICASE [43]. The modification of the TCP/IP stack developed there is applicable to the 2.3.x kernels through manual patching. It allows to disable delayed acknowledgment for TCP_NODELAY connections which eliminates the gap as shown in the Fig. 6. In the mixed case, the probability of the packet to be subjected to the delayed acknowledgment was 50%. It should be noted that Linux TCP/IP performance may vary depending on the version of the kernel. The jitter artifacts with MPI messages become less pronounced in the development versions more recent than 2.3.26.

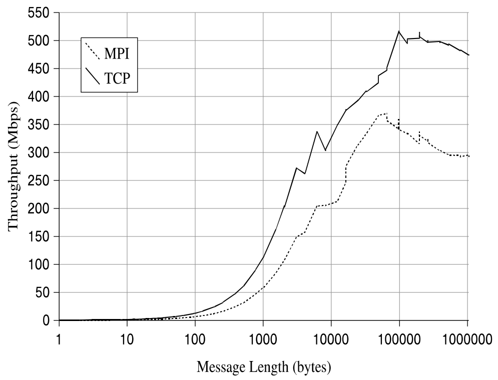

Fig. 4. The bandwidth of TCP/IP and MPI through the loopback interface on a Galaxy node.

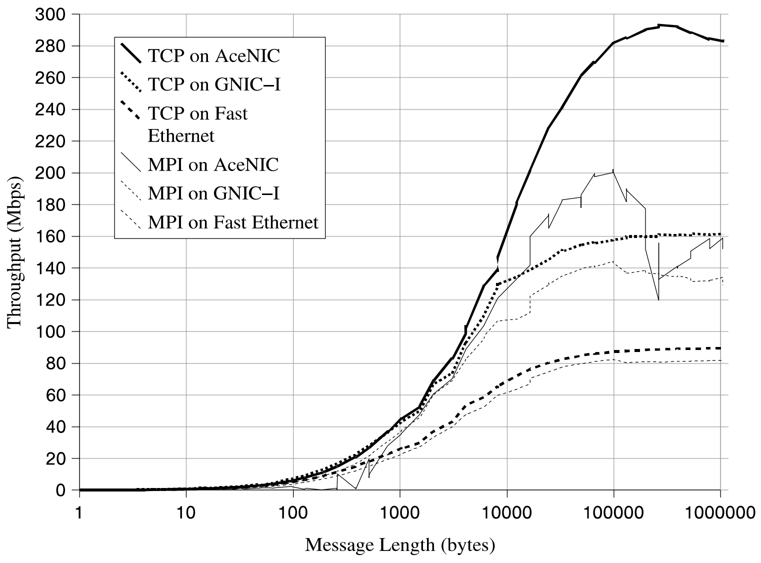

Fig. 5. The bandwidth of TCP/IP and MPI through three different network devices – two generations of Gigabit Ethernet cards and Fast Ethernet.

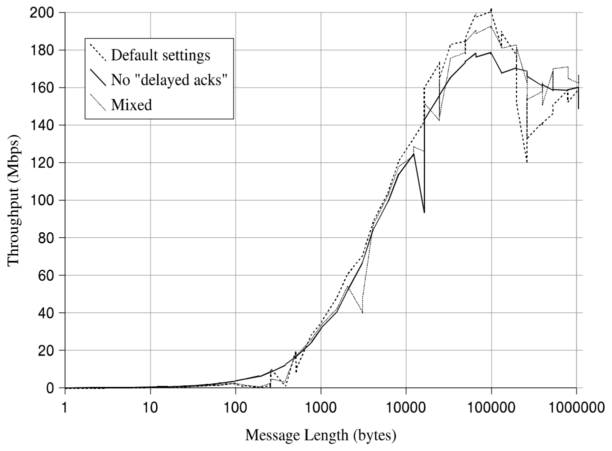

Fig. 6. Effects of changing the "delayed ack" strategy of the Linux kernel version 2.3.26 on MPI bandwidth though AceNIC.

The one-way latency of one-byte long messages on the loopback interface as measured by *netperf* [44] is 58 μs, most of which is overhead due to the TCP/IP stack. The number for TCP/Fast Ethernet latency is 128 μs, TCP/Gigabit Ethernet latency is 98 μs. Latency increases proportionally to the message length, as shown in the Fig. 7.

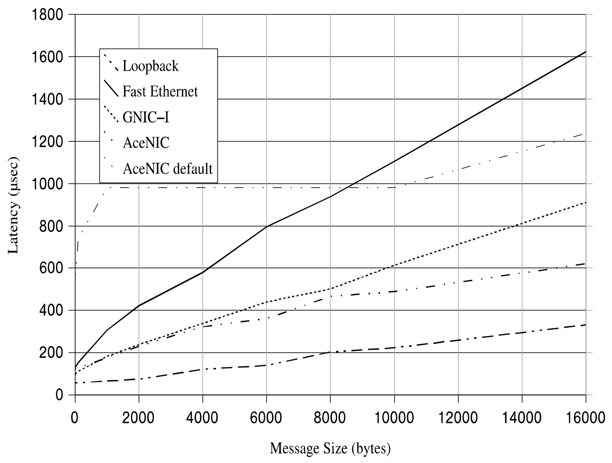

Fig. 7. One-way latency of messages through the loopback, Fast Ethernet and Gigabit Ethernet devices.

For a comparison, the Trapeze project [45] in Duke University reported the latency of 59 μs for TCP/Myrinet interface. MPI on systems like QsNet and Myrinet is capable of achieving short message latency of less than 10 μs [21,46] when used with custom transport layers, not based on TCP/IP.

It should be noted that Gigabit Ethernet does not automatically provide better short message latency. Gigabit Ethernet inherited the frame size from Fast Ethernet which led to the much higher packet rates for the adapters to process. Often, NIC vendors implement a feature known as interrupt coalescing when the adapter waits until a certain number of packets arrived before generating an interrupt for the operating system. In this case, the latency increases compared to the regular Fast Ethernet cards. This is what we observed with Linux driver for Alteon AceNIC adapter which enables coalescing by default. After decreasing the above number of packets to four and lowering the wait time to 4 μs the latency for one-byte long messages improved from 458 to 120 μs and stayed low as the message size increased.

## 4.4. Local applications

One of the most extensively used applications on Galaxy is FronTier. It is an implementation of front tracking method for numerical solving of problems with significant discontinuities. The code uses domain decomposition in a rectangular grid for parallelization. The problem is very computational intensive and, as a result, it scales well on Galaxy since it does not depend as much on the communication speed.

Table 6 shows a sample of the performance of FronTier for the computation of two-phase mixing properites in Rayleigh–Taylor instability [47].

Table 6
Scalability of FronTier code on Galaxy

| CPUs | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|
| Execution time (s) | 11 645 | 5971 | 3310 | 2051 | 1240 |

## 5. Economic discussions

The cost for each Galaxy node including the processors, memory, disk, motherboard, etc. at the time of purchase is close to $2000 (see Table 7). The cost of the software is negligible. Commercial development tools are often priced in per-host terms, thus, we avoid purchasing multi-host licenses by making the front end as large as possible to accommodate all users. When the cluster is completed the shared common cost for the network switches and the parallel IO system will add up to $1000 for each node, resulting in an average total node cost of $3000. An dual-processor node is capable of delivering more than 400 Mflops. This yields a cost-performance ratio of more than 100 Kflops-per-dollar. But, the complete and nearly turn-key systems such as the Cray T3E, IBM SP/2, SGI MIPS Clusters and DEC Alpha Clusters do not exceed 50 Kflops-per-dollar. This estimate considers only the one-time purchasing cost; the high-priced maintenance charge is extra. Not to mention the incovenience of necessary alteration to the system such as adding some nodes or adding memory to some nodes. The technology is mature enough and it is a win-win-win to build one's own supercomputer: save money, acquire a new skill and do better research.

## 6. Galaxy's future

Currently, Galaxy has 128 processors (300 and 400 MHz Pentium II) and a mix of fast Ethernet and Gigabit switches. The system is attached to a parallel sub-IO system with more than 170 Gb of disk space. Eventually, we hope to have a configuration with 512 processors connected through Gigabit Ethernet.

Table 7
Past, present (10/99) and expected near-future costs of a Galaxy node

| Components | Past costs (5/99) | Present costs (10/99) | Future costs (5/00) |
|---|---|---|---|
| Two Pentium processors | 800 | 800 | 800 |
| 512 Mb of SDRAM | 500 | 600 | 500 |
| Integrated motherboard | 500 | 450 | 450 |
| Local disk | 200 | 150 | 150 |
| All others | 100 | 100 | 100 |
| Total | 2100 | 2100 | 2000 |

## Acknowledgements

We wish to thank Dr. J. Glimm, Dr. F. Tangerman, Mr. E. Roman, and Mr. G. Smith for their help in various stages of the Galaxy project. Financial support from the New York State, ARO, and NSF is greatly appreciated.

## References

[1] California Institute of Technology URL: http://www.caltech.edu/.
[2] Columbia QCD project URL: http://phys.columbia.edu/.
[3] GRAPE project URL: http://grape.c.u-tokyo.ac.jp/gp/.
[4] Avalon (Alpha Linux Cluster) URL: http://cnls.lanl.gov/avalon/.
[5] Duke Brahma project URL: http://www.phy.duke.edu/brahma/.
[6] NIH Lobos project URL: http://www.lobos.nih.gov/.
[7] University Bonn Parnass2 URL: http://wwwwissrech.iam.uni-bonn.de/research/projects/.
[8] Stony Brook Galaxy URL: http://galaxy.ams.sunysb.edu/.
[9] D.J. Becker, T. Sterling, D. Savarese, J.E. Dorband, U.A. Ranawake, C.V. Packer, BEOWULF: A parallel workstation for scientific computation, in: Proceedings of the 1995 International Conference on Parallel Processing (ICPP), 1995, pp. 11–14.
[10] ATM Forum URL: http://www.atmforum.com/.
[11] Dolphin Interconnect Solutions URL: http://www.dophinics.no/.
[12] J. Glimm, E. Isaacson, D. Marchesin, McBryan, Front tracking for hyperbolic systems, Adv. Appl. Math. 2 (1981) 91–119.
[13] R. Alan McCoy, Y. Deng, Parallel particle simulations of thin-film deposition, Int. J. High Performance Comput. Appl. 13 (1) (1999) 16–32.
[14] K. Chu, Y. Deng, J. Reinitz, Parallel simulated annealing algorithms by mixing states, J. Comput. Phys. 148 (1999) 646–662.
[15] Y. Deng, J. Glimm, Y. Wang, A. Korobka, M. Eisenberg, A.P. Grollman, Prediction of protein binding to DNA in the presence of water-mediated hydrogen bonds, J. Mol. Model. 5 (1999) 125–133.
[16] AMI Goliath motherboard URL: http://www.ami.com/motherboards/s730spec.html.
[17] Intel PRO/100 network cards URL: http://www.intel.com/network/products/desktop_adapters.htm.
[18] Mylex BT-958 SCSI controllers URL: http://www.mylex.com/products/mp/index.html.
[19] Storage Computer RAID-7 R3X.
[20] Parallel Virtual File System URL : http://www.parl.clemson.edu/pvfs/.
[21] QSW High Speed Interconnect URL: http://www.quadrics.com/web/public/fliers/qsnet.html.
[22] Myricom URL: http://www.myri.com/.
[23] Foundry Networks URL: http://www.foundrynetworks.com/.
[24] Atomic2 Project URL: http://www.isi.edu/atomic2.
[25] Alcatel Internetworking URL: http://www.ind.alcatel.com/.
[26] Alteon Networks URL: http://www.alteon.com.
[27] Gigabit Ethernet Alliance URL: http://www.gigabit-ethernet.org/.
[28] Virtual Interface Architecture URL: http://www.nersc.gov/research/FTG/via.
[29] SMP Linux URL: http://www.uk.linux.org/.
[30] MPI Forum URL: http://www.mpi-forum.org/.
[31] Intel Paragon Reference Manuals URL: http://quest.cc.purdue.edu/Paragon/Docs/Intel/Manuals/.
[32] MPICH Implementation URL: http://www-unix.mcs.anl.gov/mpi/mpich/.
[33] LAM/MPI Parallel Computing URL: http://www.mpi.nd.edu/lam/.
[34] National Enegry Research Scientific Computing Center URL: http://www.nersc.gov/.
[35] Portable Batch System URL: http://pbs.mrj.com/.
[36] NAS Parallel Benchmarks URL: http://www.nas.nasa.gov/Software/NPB/.

[37] NetLib Repository URL: http://www.netlib.org/.
[38] STREAM Benchmark URL: http://www.cs.virginia.edu/stream/.
[39] Q.O. Snell, A.Mikler, J.L. Gustafson, NetPIPE: A network protocol independent performance evaluator, in: IASTED International Conference on Intelligent Information Management and Systems, June 1996.
[40] ASCI Red Libraries URL: http://www.cs.utk.edu/ghenry/distrib/.
[41] Data Display Debugger URL: http://www.cs.tu-bs.de/softech/ddd/.
[42] The Portland Group URL: http://ww.pgroup.com/.
[43] The Coral Project URL: http://www.icase.edu/CoralProject.html.
[44] Network Performance Benchmark URL: http://www.netperf.org/.
[45] Duke Trapeze project URL: http://www.cs.duke.edu/ari/trapeze/.
[46] Laboratory for High Performance Computing URL: http://lhpca.univ-lyon1.fr/.
[47] D. Saltz, D. Sendersky, Computation of two-phase mixing properties in Rayleigh–Taylor instability, Int. J. Multi Phase Flow (1999) submitted.