

Personal Parallel Supercomputers

Carlos Silesky, John Sobolewski
University of New Mexico
Computer & Information Resources & Technology
Albuquerque, NM 87131
silesky@unm.edu, jsob@unm.edu

Abstract

High performance or supercomputers are generally believed to be large and expensive systems affordable only by large corporations and well endowed research universities. What is not generally known, is that relatively powerful computer systems, having more than one order of magnitude the performance of a CRAY-1 computer for many scientific and engineering applications, can be build relatively quickly and inexpensively using public domain software, commodity personal computers (pc's) and networking components. Such systems, with a peak theoretical performance of more than one Gigaflop, can be easily built for well under \$50,000 and are creating a price-performance revolution for many applications that previously were confined to running on expensive supercomputers. In this paper such systems are described, together with their implementation, operational characteristics, price-performance comparisons with their commercial counterparts, as well as their limitations. These pc based supercomputers place high performance computing within reach of community colleges, departments and even individuals, in which case they may be called personal supercomputers.

1. Introduction

Not too long ago, supercomputers could only be acquired by national laboratories, larger research universities and corporations

with large research budgets. However, recent advances in commodity microcomputers and network interconnect technologies, coupled with the availability of a free UNIX operating system (Linux [2,3]), have made it possible to easily assemble a cluster of personal computers into a parallel "supercomputer" with a theoretical peak performance of well over one Gigaflop at a cost of less than \$10-20 per Megaflop. Such systems rival the performance of much more expensive systems for many types of engineering and scientific applications and are affordable to organizations and even to individual researchers.

Over the past 2-3 years, several important events have enabled the assembly of these personal supercomputers from commodity parts that can be readily purchased from local personal computers and network component vendors. These events include[1]:

1. The dramatic improvement in microprocessor performance which has been doubling every 18 months, and especially the improvement in floating point performance of these microprocessors, which has been increasing at an even faster rate in the last few years. This has greatly narrowed the performance between high volume (and therefore low cost) Intel (or equivalent) microprocessors used for personal computers and top-of-the-line workstation microprocessors available

from vendors such as DEC, SGI, IBM, HP and Sun.

2. The emergence of the Peripheral Component Interconnect (PCI) bus as a de-facto standard which supports data transfer rates in excess of 100 MBytes per second between the processor and external devices. The PCI bus allows interface cards such as 100 Mbps or 1,000 Mbps ethernet cards to be plugged into any PCI based machine, regardless of the CPU architecture.
3. The improvement in cost/performance for interconnect technologies has perhaps been even more dramatic. Fast 100 Mbps ethernet is capable of data transmission in excess of 10 MBytes per second between two points using interface cards that cost less than \$80 each. Commodity 100 Mbps ethernet switches are now available that can support 100 Mbps transmission on 8-32 ports simultaneously at a cost of less than \$100 per port. The emergence of the Gigabit (1,000 Mbps) ethernet standard promises 1,000 Mbps ethernet interfaces and switches at the above prices within the next 2-3 years.
4. A free UNIX operating system (Linux), originally developed to run on Intel or equivalent personal computers, is now available for DEC Alpha, IBM Power PC and Sun Sparc architectures and supports a large number of peripheral device drivers, notably for storage and networking.
5. The Message Passing Interface (MPI) standard which has enabled the development of reasonably portable parallel application software for UNIX environments using the message passing paradigm for interprocessor communication and data exchange.

These commodity hardware and software components allow reasonably self-

sufficient computer users to build themselves an inexpensive personal supercomputer that is simple, reliable, inexpensive to maintain and reasonably user-friendly with excellent cost/performance for many applications, especially those that are naturally parallel (e.g., Monte-Carlo Techniques), insensitive to communication latency or require relatively few nodes to run. Recent research shows that many scientific and engineering applications fall in these classes. An analysis of about 178,000 batch jobs run at the Maui High Performance Computing Center (MHPCC), for example, shows that 78% of all jobs submitted by a very heterogeneous user base used eight processors or less and 94% of all jobs used 32 processors or less [4]. The cumulative distribution of these jobs is shown in Figure 1. This, together with the fact that a 300 or 350 MHz Pentium personal computer (or equivalent) has approximately the same performance as a processor node at the MHPCC, suggests that a relatively large percentage of these jobs could be run on a personal supercomputer consisting of a cluster of 8-32 personal computers. However, users with very large jobs that are sensitive to communication latency, require large address spaces, use many processors, and require fast execution times should continue to use commercial supercomputers designed for this purpose.

2. Personal Supercomputer Architecture

The architecture of a personal supercomputer is very simple and is illustrated in Figure 2. It consists of n processors or nodes and their peripherals (where $n \geq 2$ for a parallel system), an interconnect network fabric to allow each node to communicate with all others, a

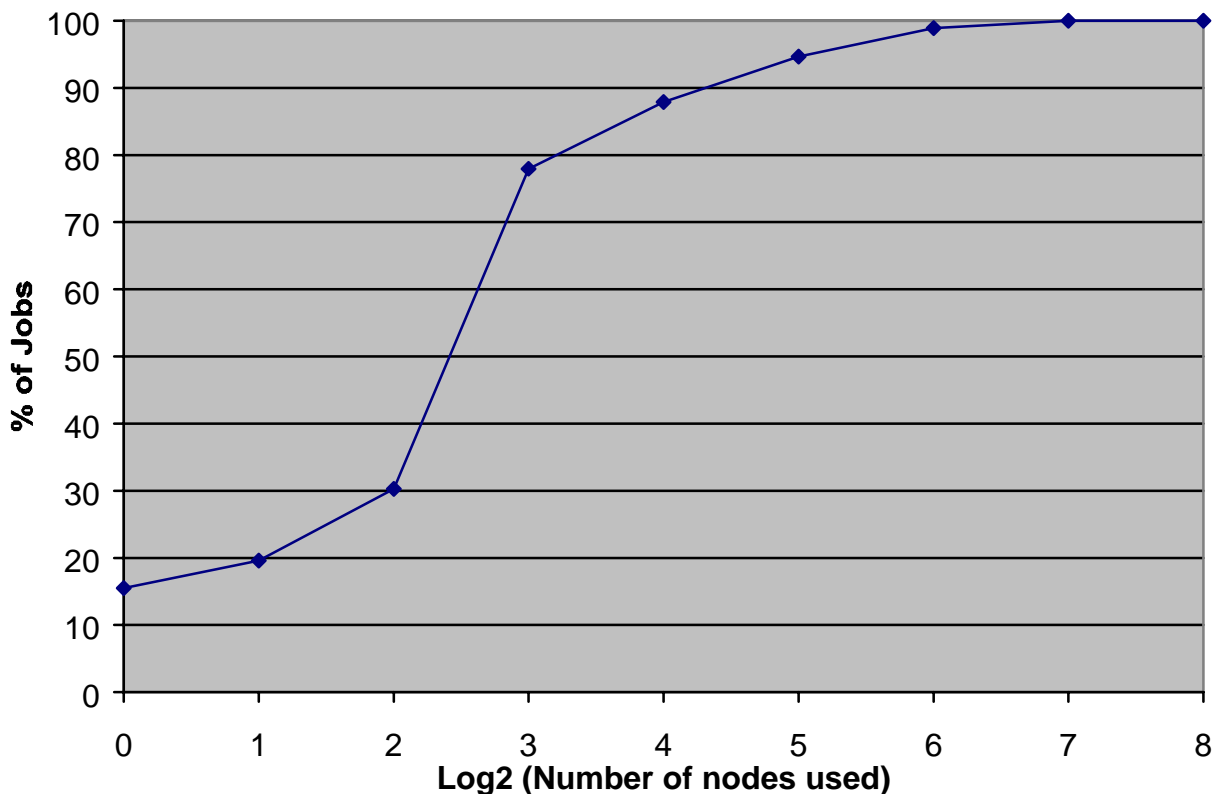


Figure 1. Cumulative distribution of parallel batch jobs processed at the MHPCC as a function of the number of nodes used.

suitable operating system and message passing software. In effect, this is identical to the architecture of a conventional shared - nothing (in the sense that the processors or nodes do not share memory or peripherals) parallel computer where the message passing software is used to exchange (share) data and other information among the processors over the interconnect fabric using the explicit message passing programming paradigm.

There are many commercial systems using this architecture with proprietary processors, interconnects and software. To reduce the cost of such proprietary systems, less costly commodity hardware and software must be used. The system that was built and evaluated consisted of the following:

1. Eight 233 MHz personal computers each with AMD K6 processor, PCI bus, 128 MB of memory, 512 KB of cache, 8 GB of disk and a 10/100 ethernet card.
2. A Bay Networks 10/100 Mbps Model 350 ethernet switch for the interconnect fabric. This is a switch that allows all mutually exclusive pairs of processors to communicate with each other at full rated speed in a non-blocking manner.
3. The Linux operating system available from Redhat and the public MPI (mpich version 1.0.13) parallel library environment. The software was configured on one machine and cloned on the others.

It is desirable to attach an additional processor to the cluster to serve as a control, software development, and compile processor. In general, this processor should

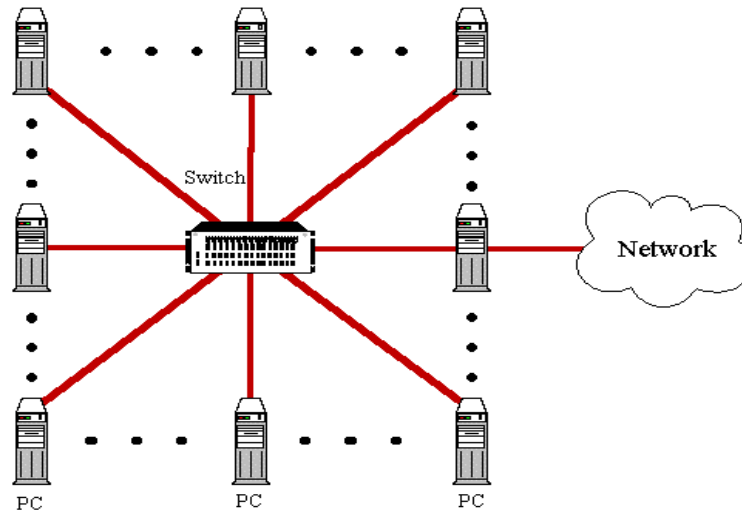


Figure 2. Architecture of a personal parallel supercomputer using commodity components. The switch provides the interconnect fabric.

have additional disk for storing large data sets and needed software tools as well as a larger memory to compile application programs.

3. Cost and Performance Issues

For a given parallel application code that is not memory bound, the performance of the shared-nothing parallel architecture described depends primarily upon the following:

1. The processor and memory performance.
2. The effective communication bandwidth between processors which is a function of the message size m . It is equal to m/t_c where t_c is the total time to transmit a message of length m from one processor to another.

Obviously, better overall system performance can be obtained by using the fastest processors available, such as the proprietary DEC Alpha, Sun Sparc or similar machines. However, for an n node

system, that increases the cost since these more powerful nodes may cost an order of magnitude, or even more, than a 300 MHz personal computer that may be obtained for under \$1,000 today. Similarly, proprietary network interconnects (such as Myrinet for example) with proprietary software drivers can greatly increase the effective communication bandwidth, but usually at much greater cost. In short, using higher performance proprietary processors and interconnects results in a cost-performance tradeoff. Proprietary components not manufactured in volume can also result in significantly higher initial and recurring maintenance costs, although they can result in significantly better performance for certain types of applications.

For parallel programs, it is important to understand the factors that affect their performance on a given system. Perhaps the most important ones are the degree of parallelization (the ratio of parallel to serial code), and the ratio of computation to communication among the nodes, both of which should be as large as possible. Figure 3 shows typical communication patterns for

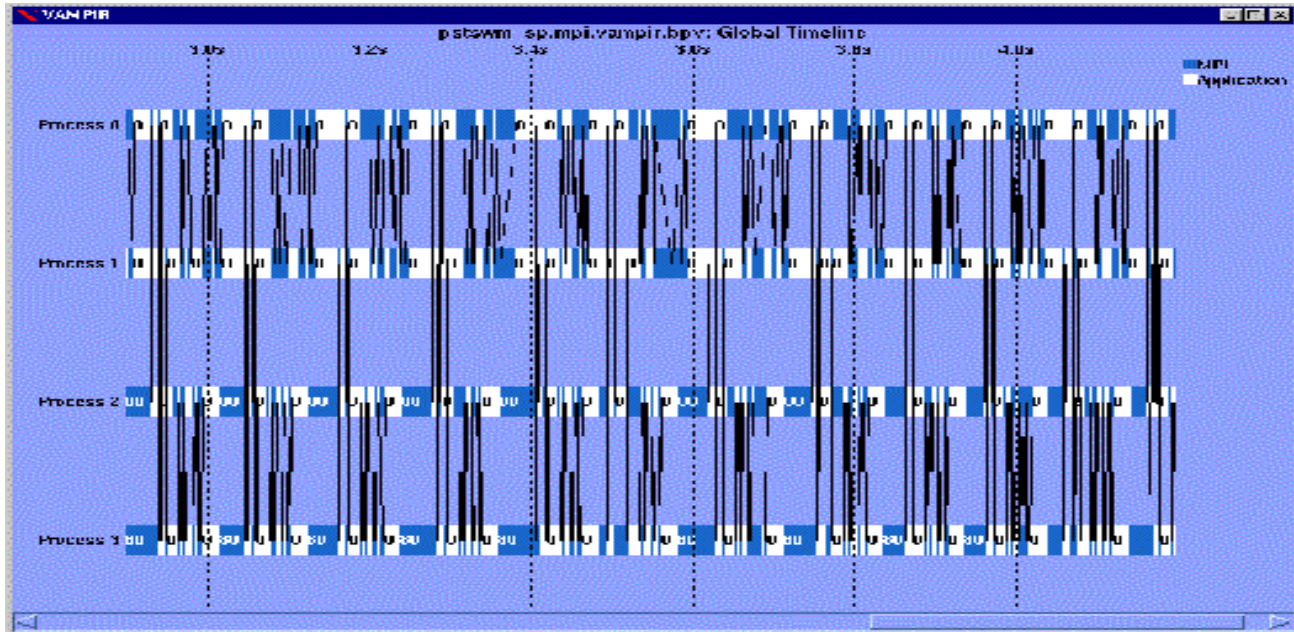


Figure 3. Communication patterns among four processors running the PTSTWN benchmark showing when the processors compute (white areas) and when they communicate with each other (gray areas).

a parallel program running on four nodes. The greatest speedup is obtained when all processors compute all the time. However, when data or other information needs to be exchanged between the nodes, processing is interrupted and needed messages are exchanged using explicit message passing calls (e.g., MPI sends and receives) as shown. The total time for which processing is interrupted must be minimized, and it is, therefore, important to minimize both the frequency and the length of time taken to pass the messages. The latter depends on the effective communication bandwidth, while the former is a function of the parallel application and the algorithm used to implement it.

4. Performance Evaluation

To evaluate the performance of both single and parallel processor systems, a number of standard benchmarks were used. They included:

1. The standard Specint 95 and the Specfp 95 benchmarks [5] which consist of a number of kernels chosen to represent real world applications and which are used to determine the integer and floating point performance of single workstations. Care should be taken when using these benchmarks in that the performance for given workstations may vary by a factor of 2-3 depending upon the amount of tuning performed. The numbers shown in Table 1 for Specint and Specfp performance used tuning and optimization parameters suggested by the respective vendors.
2. The PARKBENCH suite of programs [6] designed specifically to evaluate the performance of both single and multiprocessor systems in engineering and scientific applications.

The PARKBENCH suite consists of programs that test clock accuracy, vector operations, memory bottlenecks, system

interconnect performance, and parallel performance using a number of kernels commonly used in computational fluid dynamics, computational chemistry as well as scientific modeling and simulation. Specifically:

1. Tick 1 and Tick 2 measure system clock resolution and accuracy respectively to ensure that subsequent benchmarks are timed accurately.
2. Rinf1 measures the performance of some of the more common vector operations.
3. Poly 1, 2 and 3 evaluate in-cache and out-of-cache memory performance, as well as system performance when data has to be accessed from another processor.
4. COMMS 1, 2 and 3 test the interconnect network performance by passing data from one processor to another (ping-pong test), doing a simultaneous exchange and an all-to-all exchange to determine the full effective bandwidth of the interconnect switch.
5. SYNCH 1 measures the time taken to synchronize n processors by executing a barrier statement which is frequently used in parallel applications.
6. BT, LU, MG and SP represent a subset of the NAS parallel benchmarks and evaluate floating point performance using kernels with varying amounts of inter-processor communication. These kernels are frequently used to solve partial differential equations commonly encountered in computational fluid dynamics and other scientific and engineering applications.
7. PTSTWN which is NCAR's shallow water benchmark to evaluate floating point performance with frequent message passing (see Figure 3 which shows the communication patterns among four processors running PTSTWN). This benchmark is an

indicator of system performance for scientific modeling and simulation.

The above benchmarks were used to evaluate the performance of personal computers and workstations with various commodity interconnects and of commercial parallel scalable machines with high performance proprietary interconnects. The personal computers evaluated were conventional 233 and 266 MHz systems with 512 Kb of cache and 128 MB of memory. The commodity interconnects included 10/100 Mbps ethernet switches and OC-3 (155 Mbps) ATM switches, while the proprietary interconnect was the high speed switch used on IBM SP scalable parallel systems. Furthermore, the SP systems evaluated included four types of different nodes, including:

- 66 MHz Power2 thin and wide nodes. The difference between the two is that the wide node has twice the memory bandwidth of the thin node and, therefore, has superior performance for applications that are memory intensive.
- 120 and 160 MHz IBM P2SC nodes which have similar architectures to the 66 MHz nodes, but have higher clock speeds.

5. Results and Observations

The results of the single processor benchmarks are shown in Table 1. From these figures, the following general observations can be made:

1. All systems attain only a fraction of their peak theoretical floating point performance when running benchmarks that attempt to represent real applications.
2. The Spec benchmarks indicate that the 233 MHz personal computer is slightly

faster than a 66 MHz IBM SP Power 2 wide node for integer applications but slightly slower for floating point applications.

3. The other benchmarks, on the other hand, indicate that the 233 MHz personal computer has 20-95% of the performance of the 66 MHz wide node, while the 266 MHz system has 25-150% of the performance, depending upon the benchmark.
4. Using linear extrapolation, these results suggest that the latest personal computer (e.g., a 450 MHz system) would have a performance of about 20-70% of the latest commercial node such as the 160 MHz IBM P2SC, but that this performance is achieved at only about 10% of the cost. In other words, the personal computer has a 2-7 cost/performance edge over the commercial node.
5. The wide variation in relative performance between the processors for the various benchmarks is due to differences in the internal architecture of the respective processors, in particular the number of functional units and the effective CPU to memory bandwidths. This variation also demonstrates the pitfalls inherent in using the wrong benchmarks to determine the relative performance of various systems in a given environment. For example, while the Spec benchmarks show that the 233 MHz personal computer and 66 MHz IBM wide node are close in performance, the other benchmarks clearly show that the latter is significantly faster for scientific and engineering applications such as those represented by the 100x100 linpack, MG or SP benchmarks.

Figure 4 shows the effective bandwidth, in MBytes/sec, as a function of message size

for a variety of interconnects and protocols. Here, the effective bandwidth is assumed to be equal to m/t_c where m is the message size and t_c is the total time to transmit a message of length m from one processor to another. The following observations can be made:

1. In general, the effective bandwidth increases with increasing message sizes, implying that frequent and small message sizes lead to inefficiency and poor parallel performance.
2. Proprietary interconnect switches using proprietary protocols are much faster than the commodity components and protocols tested. Proprietary switches using standard protocols such as TCP/IP have a much lower bandwidth than those using proprietary protocols, but that bandwidth is still higher than those using the commodity interconnects tested.
3. Socket interfaces are faster than using MPI, but the latter results in greater program portability.
4. ATM is only slightly faster than 100 Mbps ethernet but is 5-10 times more expensive per port.
5. For large message sizes, 100 Mbps ethernet on a personal computer is 10 times slower than the proprietary switch but is about 60 times cheaper. This implies a cost/performance advantage by a factor of 6 for 100 Mbps ethernet. However, much of this cost performance advantage disappears for small message sizes (e.g., $m < 512$ Bytes) because of the high latency of the complex TCP/IP protocol compared to the simpler proprietary protocols used on commercial switches. Much of the difference in complexity is due to the fact that TCP/IP was designed to operate (and recover from errors) over local and wide area networks using error-prone transmission media, whereas the proprietary protocols are designed to

work over very short distances with the assumption of virtually error-free transmission.

In general, results from the COMMS 2 and 3 benchmarks support the above observations. The switch saturation

benchmark (COMMS3) for example showed that with 100 Mbps ethernet, each processor could transmit at 6.01 MBytes/sec, while the IBM proprietary switch supported a transmission rate of 71.16 MBytes/sec per processor.

Benchmark	Processor Type					
	IBM Power2		IBM P2SC		PC	
	66 MHz Thin	66 MHz Wide	120 MHz	160 MHz	233 MHz	266 MHz
Max. Th. Mflops	266	266	480	640	233	266
Specint base 95	-	3.19	-	-	5.24	5.92
Specfp base 95	-	8.51	-	-	7.04	-
100 x 100 (Mflops)	-	60.81	61.33	98.77	19.02	23.52
Poly 1 (Mflops)	-	89.05	280.33	385.26	76.76	120.21
Poly 2 (Mflops)	-	81.03	338.92	855.57	77.69	132.35
Rinfl (secs)	-	12.5	7.47	4.01	59.27	45.24
PTSTWM (secs)	-	4.01	3.37	2.03	8.81	9.61
BT (Mflops)	-	53.47	76.05	90.32	16.04	20.83
LU (Mflops)	-	54.96	70.62	99.92	22.96	19.71
MG (Mflops)	-	44.52	50.55	65.53	9.5	10.70
SP (Mflops)	-	41.27	55.01	64.1	11.16	15.27

Table 1 - Single Processor Performance

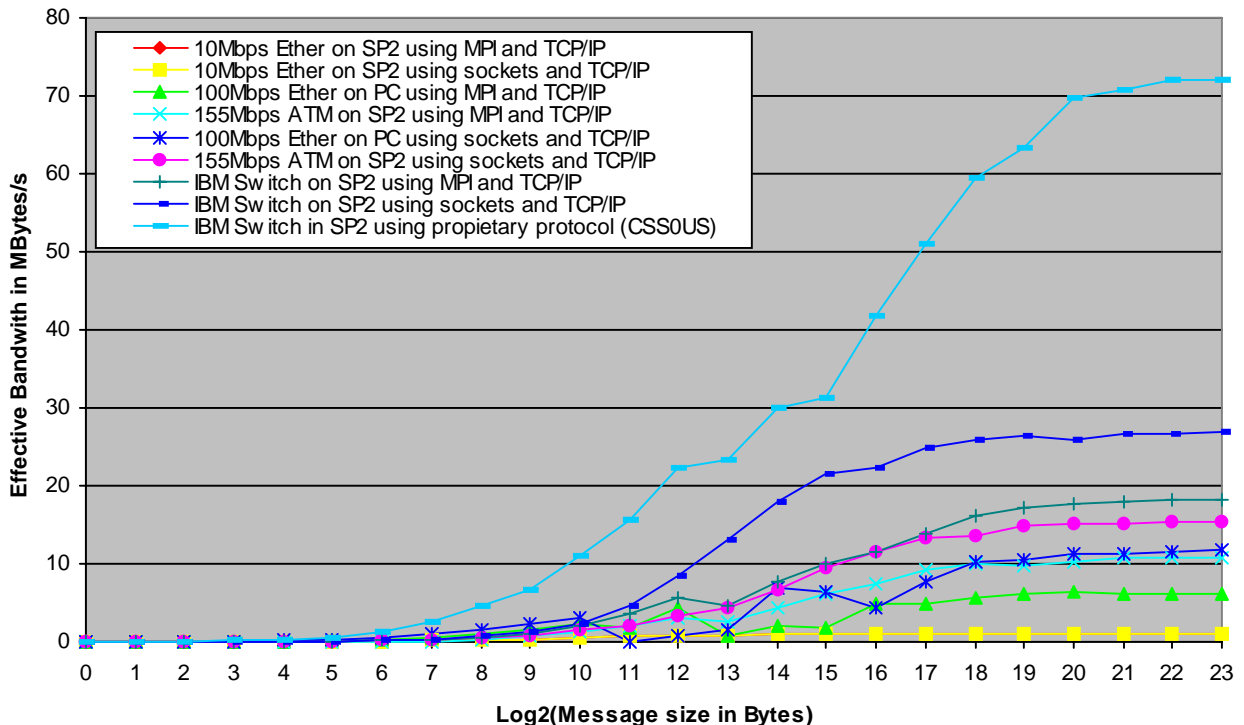


Figure 4. Plot showing the effective bandwidth as a function of message size for various types of interconnects and protocols.

The results of the parallel benchmarks for 4 processors are shown in Table 2. The interconnect fabric was the 100 Mbps ethernet switch for the personal computers and the proprietary high performance switch for the IBM SP system. Comparing these results with those shown in Table 1, the following observations can be made:

1. While a single 233 MHz personal computer node has 20-95% of the performance of an IBM 66 MHz Power 2 wide node, the four processor personal computer system has only about 12-40% of the performance of its commercial counterpart, but at only 5-7% of the cost, which represents a large improvement in cost/performance.
2. The pc supercomputer performs best when the computation to communication ratio is highest (e.g., LU benchmark) and worse when that ratio is lowest (PTSTWM benchmark).
3. The PTSTWM benchmark runs faster on a single personal computer than on a 4 node personal computer system (8.81 vs 16.5 seconds). This illustrates an important concept in parallel programming, namely that additional processors can actually degrade the performance of certain parallel programs that perform frequent message passing among processors. Such programs can

cause the computation to communication ratio to decrease rapidly as more processors are used, causing a decrease in performance.

It is anticipated that Gigabit (1000 Mbps) ethernet interface cards and switches will become commodity ports soon, which together with special low latency drivers, should greatly improve the parallel performance of the personal computer based system. If performance is important for pc based supercomputers, a proprietary interconnect such as Myrinet could be used. Myrinet has a relatively low latency (about 14 microseconds) and a sustained bandwidth in the 70-90 MBps range. However, it requires special drivers, it does not scale easily beyond 32 processors and it doubles the cost of pc based supercomputers.

These results also show that much work remains to be done to better understand how the combination of interprocessor communication characteristics and interconnect network performance affect the scalability (in terms of speedup, scaleup and CPU efficiency) of application codes running on parallel machines using commodity parts. Specifically, modest investment in faster interconnects such as Myrinet, Gigabit ethernet, or Fiber Channel

Benchmark	Processor Type					
	IBM Power2		IBM P2SC		PC	
	66 MHz Thin	66 MHz Wide	120 MHz	160 MHz	233 MHz	266 MHz
BT (Mflops)	-	163.77	211.15	287.7	35.4	47.65
LU (Mflops)	-	207.04	262.65	429.18	84.59	86.45
MG (Mflops)	-	44.52	81.97	127.19	14.0	16.8
SP (Mflops)	-	125.79	181.23	229.84	42.38	51.87
PTSTWM (secs)	2.3	2.06	1.73	0.94	16.5	9.24

Table 2 - 4 Processor Parallel Performance

interconnects may result in much improved cost/performance ratios for these "new" commodity systems and may make these personal supercomputers useful for a much broader set of applications than is currently possible.

6. Conclusion

Recent hardware and software developments have made it possible for individuals and small organizations to easily build simple parallel computers using personal computers and other commodity parts. While such machines have only a fraction of the performance of their commercial counterparts for most applications, that performance is achieved at an even smaller fraction of the cost, resulting in an overall cost/performance improvement. Such machines can be built by reasonably computer literate users. They are reliable, relatively easy to use and support, and have relatively low recurring hardware and software maintenance costs. While they are not as scalable as their commercial counterparts, they make education and research in parallel computing affordable for small educational institutions, individual departments and even for individuals.

Improvements in interconnect performance and a deeper understanding of how that affects the performance of parallel application codes are needed to make these systems cost effective for a broader set of applications.

References

1. M. Warren, D. Becker, P. Goda, J. Salmon, T. Sterling, "Parallel Supercomputing with Commodity Components," available from <http://loki-www.lanl.gov/papers/pdpta97/>
2. Parallel Computing Using Linux: Installation Guide. <http://www.kachinatech.com/parallel/installation.html>
3. J. Sanders, "Linux, Open Source, and Software's Future", IEEE Software, Sept/Oct 1998, pp 88-91.
4. S. Mamidi, J. Sobolewski, "Workload Profile in a Large Parallel Computing Environment", UNM/EECE internal report, in preparation for publication.
5. The Standard Performance Evaluation Corporation. <http://open.spec.org/>
6. PARKBENCH (PARAllel Kernels and BENCHmark) <http://www.netlib.org/parkbench/>.