Predicting Cost/Performance Trade-offs For Whitney: A Commodity Computing Cluster

Jeffrey C. Becker, Bill Nitzberg and Rob F. Van der W ijngaart[1]

NAS Technical Report NAS-97-023 October 1997

becker@nas.nasa.gov
NAS Parallel Systems Group
NASA Ames Research Center
Mail Stop 258-6
Moffett Field, CA 94035-1000

Abstract

Recent advances in low-end processor and network technology have made it possible to build a "supercomputer" out of commodity components. We develop simple models of the NAS Parallel Benchmarks version 2 (NPB 2) to explore the cost/performance trade-offs involved in building a balanced parallel computer supporting a scientific workload. We develop closed form expressions detailing the number and size of messages sent by each benchmark. Coupling these with measured single processor performance, network latency, and network bandwidth, our models predict benchmark performance to within 30%. A comparison based on total system cost reveals that current commodity technology (200 MHz Pentium Pros with 100baseT Ethernet) is well balanced for the NPBs up to a total system cost of around $1,000,000.

1.MRJ, Inc., NASA Contract NAS2-14303, Moffett Field, CA 94035-1000

1.Background

Traditionally, scientific computing has been done on large, expensive supercomputers. This has been warranted by requirements on processing speed and memory, as well as I/O capacity. Within the last year, CPU speeds of commodity processors have increased to the point where they are interesting to run scientific workloads. For example, 25 MFLOPS was attained running a particular NAS Parallel Benchmark on a 200 Mhz Pentium Pro. By comparison, one node of the IBM SP2 parallel supercomputer at NAS (the Numerical Aerospace Simulation Facility at NASA Ames Research Center) achieves 65 MFLOPS performance. Thus, about 1/3 performance of the SP2 could be achieved at a significantly lower cost per node. Memory and disk prices have dropped along with processor prices, enabling the assembly of a powerful processor node with adequate memory and storage, at a low price. At the time this paper was written (late 1997), a 200 Mhz Pentium Pro system with 128 megabytes of memory, and 4 Gigabytes of disk cost on the order of $2000. In addition, the price of networking hardware for commodity networks such as Fast Ethernet (100baseT) has also dropped, enabling the cost-effective interconnection of these powerful nodes.

Another major factor enabling the construction of scientific computing clusters out of commodity PC components, is the availability of message passing libraries, such as MPI. These are used in application programs to allow the utilization of several nodes, and make the cluster appear as one large system, rather than a collection of disparate components. These libraries use the networking facilities provided by the underlying operating system. Each node runs an independent copy of the OS. Aside from the capabilities provided by the message passing library, and routing information contained in system tables, each node is oblivious to the fact that it is connected to several others to form a large system.

In addition to the cost advantage of commodity PC clusters over traditional supercomputers, there are other benefits as well. First, since they are constructed from commodity parts, these PC clusters are much more easily tailored to fit a given workload. Second, most supercomputer vendors have either gone out of business or been purchased by other companies in the last few years. The ability to harness similar computing power from commodity components constitutes a degree of insurance that such a capability will continue to be available.

We initiated the Whitney project to put together a cluster of commodity personal computer nodes with the goal of supporting a scientific workload using hundreds, and possibly thousands of nodes. Other commodity computing projects such as Beowulf [5] have been started recently, but these comprise on the order of 16 to 64 nodes, and do not address the scale we are interested in. A related goal is the development of scalable system software such as parallel file systems and schedulers to support the scientific workload typically run at our site.

Unfortunately, the parameter space for such a cluster is quite large, comprising at least CPU type, amount of memory per node, network technology/topology, and operating system. Although one of our goals is to evaluate different choices from these components, clearly we could not try every combination. Thus we developed simple models of the NAS Parallel Benchmarks version 2, as these are representative of our target workload. These are parameterized by network latency and bandwidth, single node performance, number of nodes, problem size, and hardware cost, and allow the comparison of the performance of different technologies, e.g., Fast Ethernet vs. Myrinet, at a given cost.

2.The Communication Models

We develop models for the three application benchmarks, BT, SP, and LU. All are derived from the MPI codes specified in version 2 of the NAS Parallel Benchmarks [1]. Unless otherwise specified, we use the following notation for our models:

n - grid dimension size of benchmark

p - number of processors

l - message latency in microseconds (assumed to be constant)

b - message bandwidth in megabytes per second (assumed to be constant)

We will also use a function *msgtime* which is parameterized by *l*, *b* and message size (in bytes), and in fact is simply (*l*/1000000) + (message size)/(*b* * 1048576). However, in the interest of brevity, the *l* and *b* parameters are omitted in the use of the function *msgtime* in the models. Note that all words are eight bytes long, so this will be a factor in all the message lengths.

2.1 BT

This code implements an ADI (Alternating Direction Implicit) solution of five nonlinear partial differential equations. The Beam-Warming approximate factorization yields a product of three matrices (one for each physical dimension) which are inverted sequentially. Each matrix in turn contains a large number of block-tridiagonal equa-

tions (with five by five blocks), one for each grid line. The general form of the equation at timestep t is given below. The aforementioned factors are *X*,*Y* and *Z*. *Q* is the flow quantity vector comprising five physical quantities per grid point (density, three momentum components and energy). *R* is a matrix of source terms derived from the solution at the previous timestep. Details may be found in [4].

$$(XYZ)\Delta Q = R$$

Each timestep involves the following three steps. First, the R matrix is computed. Next the *X*, *Y* and *Z* matrices are "inverted." Finally, *Q* is updated. The steps involving interprocessor communication are the computation of the *R* matrix, and "inversion" of the *X*,*Y* and *Z* matrices. These will be captured by the model.

A grid of size n^3 is distributed among the processors using the multi-partition method [3]. Figure 1 shows the multi-partitioning of a grid among nine processors. The grid is partitioned into logical three by three by three cubes of grid sub-blocks, and each processor is responsible for update of *Q*-values on three partitions. Note that the cube has been "sliced" to show the partitioning.

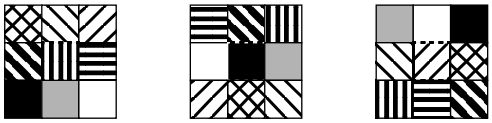


FIGURE 1. Multi-partitioning of n by n by n grid on nine processors: hatch patterns indicate processor ownership

Each grid line must be solved sequentially, but it may be worked on independently from other grid lines. Each processor gets n^2/p grid lines to work on, and $\sqrt{p}$ processors cooperate on each line solve. The line solves are pipelined in the sense that each processor works on several different sets of grid lines, and thus can work on one set of grid lines, while waiting for completion of communication corresponding to another set.

The updating of the *R* matrix involves 13 point difference stencils due to a fourth order artificial dissipation term. Thus, two neighboring points, each consisting of five quantities, will have to be received to update *R* on all the grid points on the six faces of each partition "owned" by each processor. Note that for any direction, a processor can batch together all the data it needs to send to its neighbor. In the multi-partitioning, a processor has the same neighbors in each of the three coordinate directions, except when the processor is on the edge of the grid (where it missing the neighbor that has been "wrapped around" to the opposite edge of the cube). Thus, each communication is scaled by a "batching factor" of $\sqrt{p}-1$.

Each line has a corresponding block tridiagonal matrix solve associated with it. The basic form of the equation used to solve the *X*, *Y* and *Z* matrices is;

$$\begin{aligned} &B_0 u_0 + C_0 u_1 = f_0 \\ &A_j u_{j-1} + B_j u_j + C_j u_{j+1} = f_j \quad (1 \le j \le n-2) \\ &A_{n-1} u_{n-2} + B_{n-1} u_{n-1} = f_{n-1} \end{aligned}$$

where A_i, B_i, C_i are dense five by five matrices. The solution for each of the three factors involves a forward pass in which Gaussian elimination is used to render the block tridiagonal matrix into one in which there are ones on the main diagonal (actually five by five identity matrices), and five by five blocks on the upper diagonal only (zeros elsewhere). Then, a back-substitution pass is done to solve for *u*.

In the forward pass, each matrix row is used to do Gaussian elimination on the next row. Thus, for each point on the interface between two cubes (belonging to neighboring processors), a processor must communicate a five by five matrix of pivots, and a right hand side vector of length five to the processor that owns the neighboring partition. In the backward pass, a *u* vector of length five is communicated to solve for the *u* in the previous row. Each communication is done three times (once for *X*, *Y* and *Z*), and with a multiplicity of $\sqrt{p}-1$ per grid line.

We can now write down a cost model for the communication in one iteration of BT. The total is:

$$\begin{aligned} &6 \cdot msgtime\left((\sqrt{p}-1)\left(\frac{n^2}{p}\right)(2 \cdot 5 \cdot 8)\right) + \\ &3 \cdot (\sqrt{p}-1)msgtime\left(\left(\frac{n^2}{p}\right)(2 \cdot (25+5) \cdot 8)\right) + \\ &3 \cdot (\sqrt{p}-1)msgtime\left(\left(\frac{n^2}{p}\right)(2 \cdot 5 \cdot 8)\right) \end{aligned}$$

where the first line is for the *R* matrix computation, the second line is for the *X*, *Y* and *Z* forward solves, and the last line is for the back-substitutions.

2.2 SP

SP is closely related to BT. It takes the Beam-Warming approximate factorization a step further by applying the diagonalization technique of Pulliam and Chaussee [4]. This applies a similarity transformation, and adds a fourth order numerical viscosity to change the block tridiagonal system into five uncoupled scalar penta-diagonal systems. In addition, the eigenvalues (and hence the matrix coefficients) corresponding to three of the systems are identical. Thus only three of the five systems need to be solved, for five different right hand sides.

The general form of each of the systems is given below.

$$\left(T_x X T_x^{-1} T_y Y T_y^{-1} T_z Z T_z^{-1}\right)\Delta Q = R$$

The *T* matrices are associated with the similarity transformation, and their inverses are known. Only the *X*, *Y* and *Z* matrices (which represent set of independent scalar penta-diagonal equations) need to be numerically inverted. Each timestep of SP uses the same three steps used in each timestep of BT. In addition, the multi-partition method is used just as in BT, and the allocation of grid lines to processors is identical. In fact, the computation of the *R* matrix, and the updating of *Q* is done exactly the same way as in BT. The only difference lies in the inversion of the *X*, *Y* and *Z* matrices.

In SP, each grid line has three scalar penta-diagonal systems of the form above to solve. The basic form of the *X*, *Y* and *Z* matrices is;

$$\begin{aligned}
&c_0 u_0 + d_0 u_1 + e_0 u_2 = f_0 \\
&b_1 u_0 + c_1 u_1 + d_1 u_2 + e_1 u_3 = f_1 \\
&a_i u_{i-2} + b_i u_{i-1} + c_i u_i + d_i u_{i+1} + e_i u_{i+2} \\
&\quad = f_i \qquad (2 \le i \le n-3) \\
&a_{n-2} u_{n-4} + b_{n-2} u_{n-3} + c_{n-2} u_{n-2} \\
&\quad + d_{n-2} u_{n-1} = f_{n-2} \\
&a_{n-1} u_{n-3} + b_{n-1} u_{n-2} + c_{n-1} u_{n-1} \\
&\quad = f_{n-1}
\end{aligned}$$

where a_i, b_i, c_i, d_i, e_i are scalars. The solution for each of the three factors involves a forward pass in which Gaussian elimination is used to render the scalar penta-diagonal matrix into an upper diagonal matrix with two upper diagonals, and ones on the main diagonal. Then, a back-substitution pass is done to solve for *u*.

In the forward pass, Gaussian elimination is used to eliminate the coefficients a_i and b_i on row *i*. Both rows *i-1* and *i-2* are needed to update row *i*. In particular, the transformed upper diagonal coefficients from each row are required. In addition, the pivots corresponding to the three systems (associated with the three unique eigenvalues) are communicated in the same step. Thus, for each point on the interface between two cubes (belonging to neighboring processors), a processor must communicate twelve coefficients, and two right hand side vectors of length five, to its neighbor in the multi-partitioning. In the backward pass, a *u* vector of length five is communicated to solve for *u* in the previous row. Each communication is done three times (once for *X*, *Y* and *Z*), and $\sqrt{p} - 1$ per grid line.

We can now write down a cost model for the communication in one iteration of SP. The total is:

$$\begin{aligned}
&6 \cdot msgtime\left((\sqrt{p}-1)\left(\frac{n^2}{p}\right)(2 \cdot 5 \cdot 8)\right) + \\
&3 \cdot (\sqrt{p}-1) msgtime\left(\left(\frac{n^2}{p}\right)((12+10) \cdot 8)\right) + \\
&3 \cdot (\sqrt{p}-1) msgtime\left(\left(\frac{n^2}{p}\right)(2 \cdot 5 \cdot 8)\right)
\end{aligned}$$

where the first line is for the *R* matrix computation, the second line is for the *X*, *Y* and *Z* forward solves, and the last line is for the back-substitutions.

Note that the SP and BT models are pessimistic in that they assume no overlap between computation and communication.

2.3 LU

The LU benchmark finds a steady state solution to the same set of partial differential equations as BT and SP, but uses the symmetric successive over-relaxation (SSOR) algorithm. After discretization, the system of linear equations is:

$$K\Delta U = R$$

The matrix *K* has a sparse block-banded structure with seven bands, and five by five blocks. The constituent equation associated with grid point (*i*,*j*,*k*) is given by the following, in which *i*, *j* and *k* range from 2 to *n*-1, the matrices *A*,*B*,*C*,*D*,*E*,*F*,*G* are five by five, and ΔU and *R* are vectors of length five. Note that function values at points on the edge of the grid are kept fixed (boundary conditions).

$$A_{i,j,k}\Delta U_{i,j,k-1} + B_{i,j,k}\Delta U_{i,j-1,k} + C_{i,j,k}\Delta U_{i-1,j,k} + D_{i,j,k}\Delta U_{i,j,k} + E_{i,j,k}\Delta U_{i+1,j,k} + F_{i,j,k}\Delta U_{i,j+1,k} + G_{i,j,k}\Delta U_{i,j,k+1} = R_{i,j,k}$$

The K matrix can be written as the sum of a lower triangular matrix, Y ($Y_{i,j,k}=A_{i,j,k}+B_{i,j,k}+C_{i,j,k}$), D (the diagonal), and an upper triangular matrix Z ($Z_{i,j,k}=E_{i,j,k}+F_{i,j,k}+G_{i,j,k}$). The SSOR scheme solves the system $X\Delta U = R$, where;

$$X = w(2-w)(D+wY)(I+wD^{-1}Z)$$

and w is a specified over-relaxation constant. Thus each SSOR iteration comprises the following steps. Communication is required in all but the last step.

1. Formation of the right hand side R
2. Generation and solution of the lower triangular system
$$(D+wY)\Delta U_1 = R$$
3. Generation and solution of the upper triangular system
$$(I+wD^{-1}Z)\Delta U = \Delta U_1$$
4. Solution update
$$U = U + \frac{1}{w(2-w)}\Delta U$$

The xy plane is evenly partitioned among a number of processors equal to a power of two, and each gets a vertical "pencil" comprised of all the grid points enclosed by its partition, and the z axis.The partitioning of the grid among sixteen processors is illustrated in Figure 2 below.

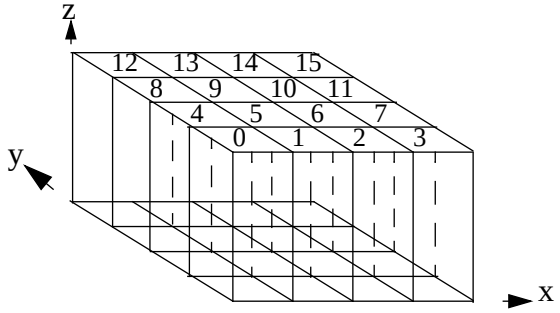


FIGURE 2. Partitioning of 3D grid in LU

The right hand side R matrix update is completed first. Each point is updated using a thirteen point difference stencil, which is required for the numerical fourth order dissipation added as in BT and SP. Thus each processor must receive two planes of data from each of its neighbors. Each plane has $n^2/\sqrt{p}$ points.

For the lower and upper triangular solution phases, each processor works on a plane of data (with fixed z) at a time. In the lower triangular phase, when a processor q is done with a plane, it communicates the east and north edges to its neighbors, which can then work on their respective planes. In addition, in accordance with the recurrence constraint, processor q can then compute the plane above the one it just completed. To illustrate, in the cube shown above, processor 0 starts on its bottom plane first (while other processors wait). The computation proceeds up and across the cube in pipeline fashion as follows. At a given time, processor 0 works on plane $z=k$, while processors 1 and 3 work on plane $z=k$-1, processors 2, 4, and 6 work on plane $z=k$-2, processors 5 and 7 work on plane $z=k$-3, and processor 8 works on plane $z=k$-4.

In the upper triangular phase, the computation proceeds in the opposite direction. When processor q is done with a plane, it communicates the west and south edges to its respective neighbors on these sides, and then the neighbors can work on their planes. Processor q can then work on the plane below the one it just completed. Thus in the cube above, processor 8 starts on its top plane first (while other processors wait). The pipeline then proceeds down and across the cube as follows. At a given time, processor 8 works on plane $z=k$, while processors 5 and 7 work on plane $z=k$+1, processors 2, 4, and 6 work on plane $z=k$+2, processors 1 and 3 work on plane $z=k$+3, and processor 0 works on plane $z=k$+4.

The communication involved in both lower and upper triangular phases is identical. For each plane, $n/\sqrt{p}$ points are communicated in each of two directions. This is done for a total of n-2 planes in each processor's "pencil" partition.

We can now write out the cost model for communication in one iteration of LU. The total is;

$$4 \cdot msgtime((n^2/\sqrt{p}) \cdot 2 \cdot 5 \cdot 8) + 2n \cdot msgtime((n/\sqrt{p}) \cdot 5 \cdot 8)$$

where the first term is for the right hand side computation, and the second term totals the communication for the lower and upper triangular solution steps. We ignore the pipeline fill time. Note that if the number of processors is an even power of two, e.g., sixteen, then this model is

exact. If the number of processors is an odd power of two, then the model is still approximately correct.

3.The Overall Models

Each of the communication models described above can now be used as a component of an overall model of NAS benchmark performance. In particular we will derive models for class A, B and C sizes of the benchmarks as described in [1,2]. Table 1 shows the grid size in each of the three coordinate directions (the parameter *n* in the communication models), and Table 2 shows the iteration counts for the application benchmarks.

class	BT	LU	SP
A	64	64	64
B	102	102	102
C	162	162	162

TABLE 1. Grid size

class	BT	LU	SP
A	200	250	400
B	200	250	400
C	200	250	400

TABLE 2. Iteration counts

We now give additional parameters needed for the overall models:

f - Mops/s (millions of operations/second) per-processor performance- approximated as single node class A performance

i - the number of iterations from Table 2

m - the total Mops required to complete the benchmark - printed out at the end of each benchmark run

In order to simplify the models, we assume that the benchmarks are completely parallelizable. Although this ignores the serial fraction from Amdahl's law, the approximation is close to reality, especially on the larger classes. We can now write the overall model for the execution time *t* in terms of the parameters and the communication model, *comm*(*n*,*p*,*b*,*l*).

$$t = \frac{m}{fp} + i \cdot comm(n, p, b, l)$$

4.Model Verification

NAS benchmark performance numbers for BT, LU and SP were measured on a four processor PC cluster with all four nodes plugged directly into a Fast Ethernet switch. For all the measurements, *n* is 64 (class A), *p* is 4, *b* is 8 (MB/s) and *l* is 190 (us). These were compared to the model predictions. Results, as well as the parameters *f* and *m*, are given in Table 3. The models predict timings within 20-30%. In addition, the codes were profiled to detail the number and size of each message sent. Referring back to its model, BT has three message sizes (call them a, b and c). Tables 4 and 5 show a comparison between the message size and counts predicted by the model, and measured by the profiling. The error is 6% or less. Similar results are found for LU and SP.

	m - Mops	f - Mops/ s	runtime (model)	runtime (actual)
BT	168289	23.67	1810.09	2439.33
LU	119299	30.90	996.304	1337.43
SP	85006	18.97	1182.63	1645.81

TABLE 3. Model timing comparison (seconds)

message	model	actual	error (%)
a	81920	81920	0
b	245760	261360	6
c	40960	43560	6

TABLE 4. BT message size comparison (bytes)

message	model	actual	error
a	4800	4848	1
b	2400	2412	0.05
c	2400	2412	0.05

TABLE 5. BT message count comparison

5.Using The Models For Prediction

5.1 Performance Prediction

Equipped with validated models, we now explore the effects of varying different technology components. The nodes of the system on which the predictions are based are single processor 200 Mhz Pentium Pro PCs, each with an ASUS dual processor motherboard, 128 Megabytes of

memory, Natoma PCI chipset, and 2.5 Gigabyte IDE hard drive. All performance predictions to follow are for class C benchmarks.

Figures 3, 4 and 5 show the effects of successive increases in processor performance on BT, LU and SP respectively. In each case, the top curve has the performance of our base system on the benchmark. It is clear from the graphs that increases in processor speed on the order of a factor of two have a dramatic effect on benchmark performance. As technology changes, such increases are expected every year to 18 months. Thus the curves show what we can expect over the next few years.

Figure 6 shows the effect of network bandwidth on BT performance. The top curve is for typical 10baseT bandwidth. The next curve represents the performance of current Fast Ethernet. This first jump has the largest effect. Increasing bandwidth beyond that of Fast Ethernet (and leaving processor speed and network latency constant) does not have much effect on performance. Similar graphs can be observed for LU and SP.

Figure 7 shows that varying latency from the 10baseT value has even less effect than varying bandwidth (while holding other factors constant) on LU performance. Similar curves can be observed for BT and SP.

5.2 Cost/performance Prediction

We can also use the models to predict cost/performance (Mops/s/$) for various technologies. In particular, we are interested in comparing network technologies for Whitney, our cluster of Pentium Pro nodes. We will compare Fast Ethernet to Myricom's Myrinet technology. Fast Ethernet is more of a commodity than Myrinet, and is cheaper. However, Myrinet is capable of gigabit/s speeds. A comparison of the two technologies using our most recent price quotes, and MPI ping-pong tests is given in Table 6.

	100baseT	**Myrinet**
bandwidth (MB/s)	8	98
latency (us)	190	18
switch cost per port ($)	285	123.12
interface card cost ($)	100	1280
cable cost ($)	10	133

TABLE 6. Comparison of Fast Ethernet (100baseT) and Myrinet

The network architecture we assume for the comparison has each node going into a switch, and each switch is possibly connected to other switches. For each node going into a switch, we assume an inter-switch link of equal performance, i.e., the same technology as the node to switch link. In this simple topology, each node uses a single interface card. As described earlier, our nodes are 200 Mhz Pentium Pro based PCs. Our latest price for these is $2380.

In the figures to follow, we will compare the cost/performance of Fast Ethernet and Myrinet. These graphs will compare Fast Ethernet and Myrinet performance at a given system cost, and compare this to a system with an ideal network with infinite bandwidth and no latency at this cost (for this system, we assume the network is free, and hence, we use the base node cost of $2380). Figure 8 shows how many nodes can be bought if the cluster network is based on Fast Ethernet or Myrinet. As indicated in Table 3, Myrinet is somewhat more expensive than Fast Ethernet. Figure 8 can be used for cross-referencing when interpreting the cost/performance comparisons to follow.

Figures 9, 10 and 11 compare Fast Ethernet and Myrinet to ideal network performance at a given cost for BT, LU and SP respectively. The closer the performance of a given network is to ideal at a given cost, the more balanced it is for the application at that cost. For all three applications, Fast Ethernet provides a more balanced system up to a certain cost range, after which Myrinet is better. For BT, systems up to about $5 million in cost are better off with Fast Ethernet. Referring to Figure 8, this includes systems up to about 1600 nodes in size. For LU, the crossover point drops to about $3 million or systems up to 1000 nodes in size. For SP, the crossover point drops to about $1 million, or systems up to about 400 nodes in size. Note that because Myrinet is more expensive than Fast Ethernet, the equivalent Myrinet-based system at the crossover point has fewer nodes. These graphs clearly show that network performance is not the only consideration in specifying a system. Both the application, and the budgeted cost (system size) are also very important.

Earlier, we showed that for BT, LU and SP, processor speed was the most important factor (compared to network bandwidth and latency) affecting system performance. Thus, as a final experiment, we wanted to see how Fast Ethernet compared to Myrinet if processor performance was twice what it is today. Figure 12 shows that the crossover point for BT is approximately 40% of that of Figure 9, when processor performance is doubled. Similar effects can be seen for LU and SP. Once again, this points to the fact that for current 200Mhz Pentium Pros and small system sizes, Myrinet does not provide as much balance as

Fast Ethernet. Doubling the processor performance brings Myrinet into balance at smaller system sizes.

6.Summary and Future Work

We have developed mathematical models for the communication involved in each of the NAS application benchmarks, BT, LU and SP. We used these as a component in simple models of NAS application performance in which the code was assumed completely parallelizable (no Amdahl's serial fraction). These models were verified against actual performance of the benchmarks on a four processor PC cluster, and we showed that the model is accurate to within 30 percent.

The models were then used to make several interesting predictions. First, for these applications, processor speed affects performance much more than network bandwidth and latency. In fact the latter two factors only have an effect when jumping from 10baseT to 100baseT speeds. Further increases have little effect. Second, we showed that cost is an important consideration when deciding what sort of system to buy. In particular, we showed that even though Myrinet has superior performance characteristics compared to Fast Ethernet, for current Pentium Pro processor based systems, it does not provide better performance until the system gets large enough. In the case of BT, the crossover point is about 1600 nodes. Given current processor speeds, Myrinet does not provide for a balanced system until there are enough nodes. As expected, if processor speed doubles, our models predicted that the Fast Ethernet/Myrinet crossover point is cut to less than half.

In future work, we plan to develop communication models for the other NAS benchmarks, the kernels CG, EP, FT, IS and MG. We also plan to investigate other network technologies such as fiber channel, and Gigabit Ethernet. Finally, we plan to incorporate aspects of SMP systems into our models.

7.Acknowledgments

The authors would like to thank Maurice Yarrow for several helpful discussions about the benchmarks and models. We'd also like to thank Samuel Fineberg and Chris Kuszmaul for their comments and suggestions regarding early drafts of this work.

8.References

[1] Bailey, D.; Harris, T; Saphir, W.; van der Wijngaart, R.; Woo, A.; Yarrow, M.: "The NAS Parallel Benchmarks 2.0", Report NAS-95-020, NASA Ames Research Center, 1995.

[2] Bailey, D.; Barszcz, E.; Barton, J.; Browning, D.; Carter, R.; Dagum, L.; Fatoohi, R.; Fineberg S.; Frederickson, P.; Lasinski, T.; Schreiber, R.; Simon, H.; Venkatakrishnan, V.; Weeratunga, S.: "The NAS Parallel Benchmarks", Report RNR-94-007, NASA Ames Research Center, 1994.

[3] Naik, N.H.; Naik, V.K.; Nicoules, M.: "Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics," International Journal of High Speed Computing, Vol. 5, No. 1, pp.1-50, 1993.

[4] Pulliam, T.H.: "Notes on Solution Methods in Computational Fluid Dynamics", based on notes from: von K'arm'an Institute For Fluid Mechanics Lecture Series, Numerical Techniques For Viscous Flow Computation in Turbomachinery Bindings, von K'arm'an Institute, Rhode-St-Genese, Belgium, 1985.

[5] Warren, M.S.; Becker, D.J.; Goda, M.P.; Salmon, J.K.; Sterling, T.:"Parallel Supercomputing with Commodity Components," In H. R. Arabnia, editor, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), pages 1372-1381, 1997.

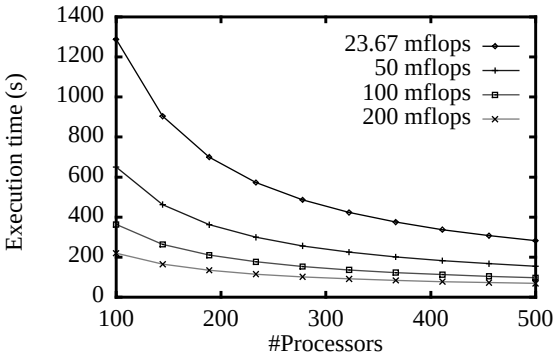


FIGURE 3. Effect of cpu speed on BT performance

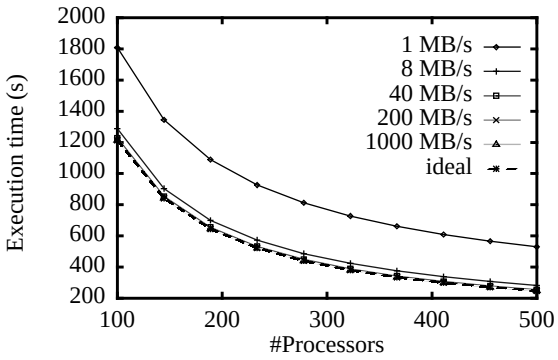


FIGURE 6. Effect of Network Bandwidth on BT performance (ideal is infinite bandwidth)

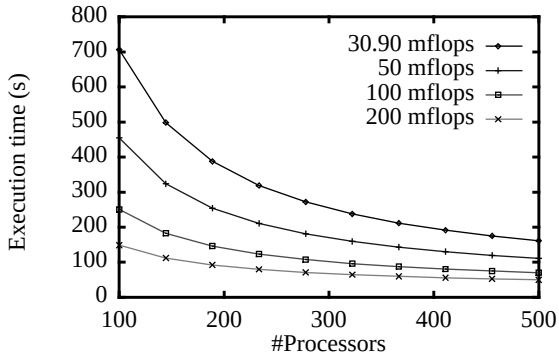


FIGURE 4. Effect of cpu speed on LU performance

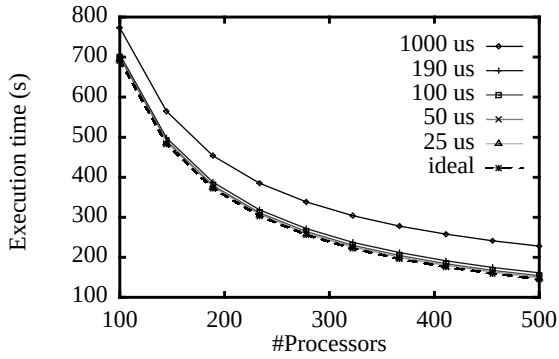


FIGURE 7. Effect of Network Latency on LU performance (ideal is no latency)

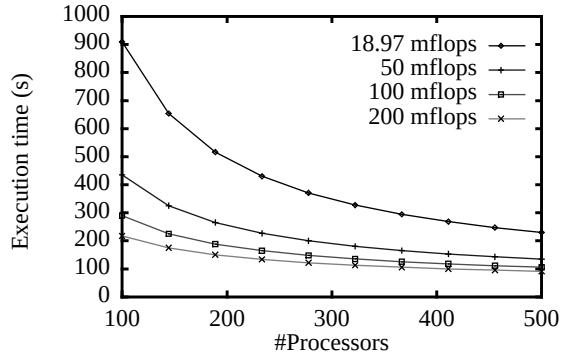


FIGURE 5. Effect of cpu speed on SP performance

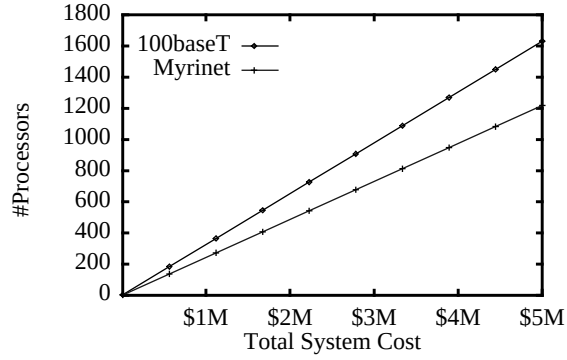


FIGURE 8. System size versus cost

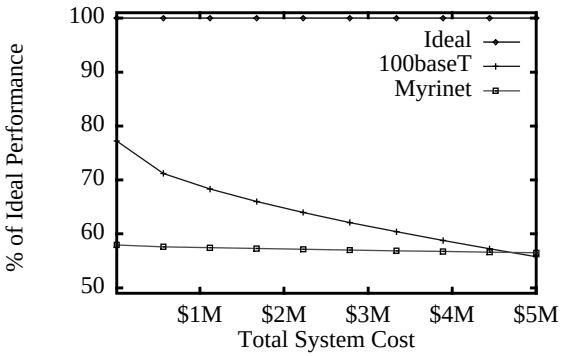


FIGURE 9. Projected BT performance (assuming 23.67 MFLOPS per node): Fast Ethernet vs. Myrinet vs. ideal

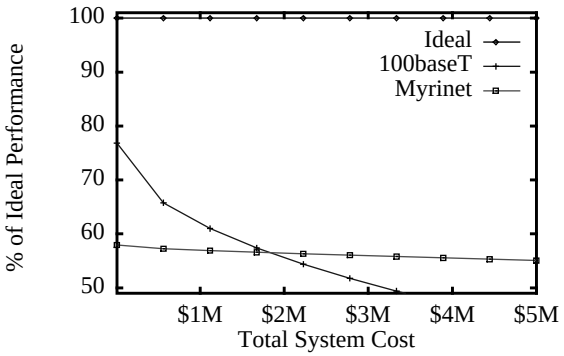


FIGURE 12. Projected BT performance (assuming 47.34 MFLOPS per node): Fast Ethernet vs. Myrinet vs. ideal

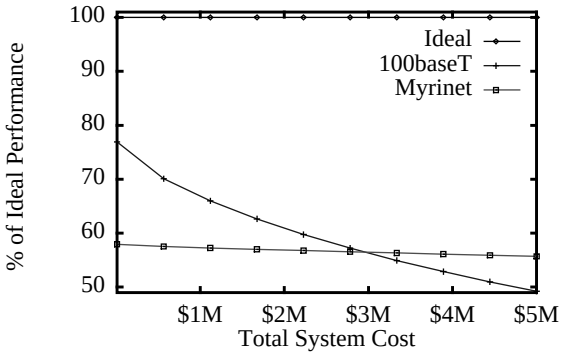


FIGURE 10. Projected LU performance (assuming 30.90 MFLOPS per node): Fast Ethernet vs. Myrinet vs. ideal

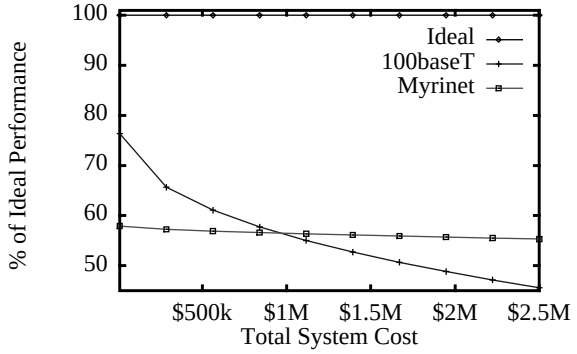


FIGURE 11. Projected SP performance (assuming 18.97 MFLOPS per node): Fast Ethernet vs. Myrinet vs. ideal