Architectural Requirements and Scalability of the NAS Parallel Benchmarks

Frederick C. Wong, Richard P. Martin, Remzi H. Arpaci-Dusseau, and David E. Culler

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley

{fredwong, rmartin, remzi, culler}@CS.Berkeley.EDU

Abstract

We present a study of the architectural requirements and scalability of the NAS Parallel Benchmarks. Through direct measurements and simulations, we identify the factors which affect the scalability of benchmark codes on two relevant and distinct platforms; a cluster of workstations and a ccNUMA SGI Origin 2000.

We find that the benefit of increased global cache size is pronounced in certain applications and often offsets the communication cost. By constructing the working set profile of the benchmarks, we are able to visualize the improvement of computational efficiency under constant-problem-size scaling.

We also find that, while the Origin MPI has better point-to-point performance, the cluster MPI layer is more scalable with communication load. However, communication performance within the applications is often much lower than what would be achieved by micro-benchmarks. We show that the communication protocols used by MPI runtime library are influential to the communication performance in applications, and that the benchmark codes have a wide spectrum of communication requirements.

1 Introduction

The NAS Parallel Benchmarks (NPB) are widely used to evaluate parallel machines [19]. To date, every vendor of large parallel machines has presented NPB results, at least with the original "paper and pencil" version 1.0 [6]. Those reports provide a comparison of execution time as a function of the number of processors, from which execution rate, speedup, and efficiency are easily computed. While extremely valuable, these results only provide an understanding of overall delivered performance. The fixed algorithm and standard Message Passing Interface (MPI) [13] programming model of NAS Parallel Benchmarks Version 2 [3] make it possible to use these benchmarks as a basis for an in-depth comparative analysis of parallel architectures. However, the current reports still provide only a crude performance comparison because the only reported result is the total execution time [14, 18]. All other performance metrics are derived from execution time.

When we measured the NAS benchmarks on the Berkeley Network Of Workstations (NOW) [2], we were pleasantly surprised to find that the speedup was as good as that of the Cray T3D, with better per-node performance, and better than that of the IBM SP-2, although with lesser performance per processor. Neither the raw speed of our MPI over Active Messages [8, 11], nor the ratio of processor performance to message performance, provided an adequate accounting of these differences. The lack of a clear explanation motivated us to develop a set of tools to analyze the architectural requirements of the NPB in detail. Given that a single pass through the Class A benchmarks is roughly a *trillion* instructions, traditional simulation techniques were intractable and therefore ruled out. Instead, we employed a *hybrid* method, combining direct measurements from a real machine with parallel trace-driven simulations. Not only does this allow us to understand the performance characteristics of an actual platform, it also shows us how different architectural parameters

affect scaling.

This paper provides a detailed analysis of the architectural factors that determine the scalability of the NAS Parallel Benchmarks on parallel machines. We use the Berkeley NOW cluster and the SGI Origin 2000, two relevant and distinct platforms, as the basis of the study. Starting from the base performance and speedup curves, we break down the benchmarks in terms of their computation and communication costs, to isolate the factors that determine speedup. This analysis shows that for machines with scalable communication performance, improvements in memory system performance due to increasing cache effectiveness compensate for the time spent in communication and the extra computational work, so much so that many applications exhibit perfect or even super-linear speedup for the machine sizes typically used for each class of data set (1 to 128 processors for Class A). This behavior is inherent to the constant problem size (CPS) scaling used in the benchmarks and can be characterized precisely by constructing working set graphs for any given input size.

The main contributions of this work are: (1) a characterization of the complex interactions of software and hardware effects on speedup, (2) a methodology for understanding speedup in the CPS domain, (3) a quantitative analysis of the architectural requirements of the NAS Parallel Benchmarks suite version 2.2, including the first detailed study of NAS benchmarks working set behavior under scaling, and (4) an evaluation on communication efficiency of applications with different MPI communication protocols.

2 Experimental Environment and Methodology

Understanding the performance under scaling of large, parallel codes is a difficult problem [15]. Ideally, one would like to run the benchmarks on real machines, but this option precludes a detailed study of different hardware characteristics such as cache size and other parameters. Simulations are problematic, because they limit the size of the problem and can potentially miss long-term effects. To remedy this situation, we apply a *hybrid* approach; where possible, we use an instrumented communication library in tandem with hardware counters to measure the execution characteristics of the benchmarks on a real machine. When necessary, we trace and simulate the benchmarks, which allows us to vary certain architectural parameters. Details for both of these methods are given below.

2.1 Direct Measurement

In this study, all executions of the NAS benchmarks are performed on a cluster of 36 UltraSPARC Model 170 workstations and on an SGI Origin 2000 system. The runtime measurements present in this paper are an average of 30 runs excluding outliers. On the cluster, each node has a single Myricom network interface card attached to the S-Bus I/O bus. The machines are interconnected with ten 8-port Myrinet [4] switches in a fat-tree like topology. Each node of the cluster is running Solaris 2.5.1 and has 512 KB of L2 cache with 128 MB of main memory. An UltraSPARC I with 512 MB of main memory is used to obtain the single processor runtime. The Origin 2000 machine consists of 64 R10000 processors running at 250 MHz. Each processor has a 4 MB L2 cache. There are 32 nodes in the machine. Each node contains 2 processors and 512 MB of main memory running IRIX 6.5 and MPI 3.0.

Our initial study of the Origin system was based on R10000-195MHz system at Numerical Aerospace Simulation Facility at NASA Ames Research Center. Over the course of investigation, the hardware configuration of the Origin system was upgraded from 195 MHz to 250 MHz. All Origin system measurements are based on the 250 MHz system except in Section 7, where the performance of NAS benchmarks on two processor speeds is discussed.

All the NAS benchmarks communicate via MPI. Our implementation of MPI for the cluster is based on the MPICH (v1.0.12) reference implementation [9]. All ADI calls are mapped to Active Messages [11] operations, and the layer is highly tuned for performance.

To break down the performance of the benchmarks, we add instrumentation code to the MPI layer. At each MPI call, we record a desired set of statistics and at the end of execution, we write the results to disk. In all cases the instrumentation adds less than 10% to overall execution time and in most cases, less than 1%. In this manner, we gather information such as message sizes, destinations, and time stamps of when communication events occur.

To measure instruction count and CPI, we use the performance counters available in the UltraSPARC and R10000 processors. At the beginning of the run, we configure one counter to count cycles, and the other to count instructions. At the start and end of each MPI event, we record both these values; at the end of the run, we can de-construct the amount of time spent and the number of instructions executed inside and outside of MPI routines. All measurements run for the full number

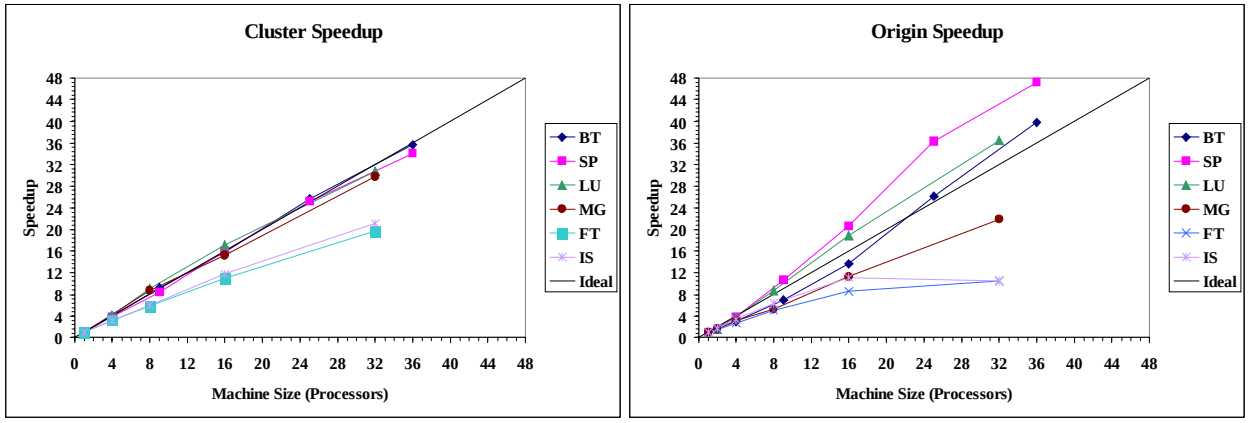


Figure 1 Speedup of NAS Parallel Benchmarks. These figures present the speedup curves for the Berkeley NOW cluster and SGI Origin 2000 on the Class A problem size of the NAS Parallel Benchmarks, version 2.2. The cluster achieves perfect or slightly above perfect speedup for all but two programs. The Origin attains super-linear speedup for several benchmarks, but has a wide spread in speedup behavior.

Benchmark	Cluster (seconds)	Origin (seconds)
BT	4177.8	2074.4
SP	2806.7	1244.6
LU	2469.3	1074.5
MG	89.9	43.1
IS	41.4	29.7
FT	131.1	94.4

Table 1 Single Processor Execution time. This table presents single processor runtimes on the Berkeley NOW cluster and the SGI Origin 2000 for the Class A problem size of the NAS benchmarks, version 2.2.

of iterations specified by the NAS benchmarks Class A problem sizes.

2.2 Simulation

While the instrumented MPI layer can give us usage characteristics and breakdowns of where time is spent in the code, it can not give us the working sets for the benchmarks. Counters provide miss rates for the particular cache design of the machine (512 KB on the cluster and 4 MB on the Origin), but we need to know how the miss rate changes with cache size.

To solve this problem, we employ the Shade simulation environment [5] on the cluster. Shade provides a SPARC-based virtual machine to an application program. We use Shade in a novel manner, running independent instances of the simulator on each of the workstations of our cluster. Inside this "virtual cluster", communication between processes of a parallel program takes place on the real Myrinet network. We have written a Shade analyzer that outputs the data cache address trace of one process on the simulated machine.

All benchmarks are traced for a single time-step. Other experiments have revealed that behavior across time steps is nearly identical for these benchmarks. After a trace is produced, we use the Dinero cache simulator [10] to simulate the desired cache configurations.

3 Speedup

Figure 1 and Table 1 show the speedup and single processor execution time, respectively, on the cluster and the Origin on the Class A problem size of the NAS

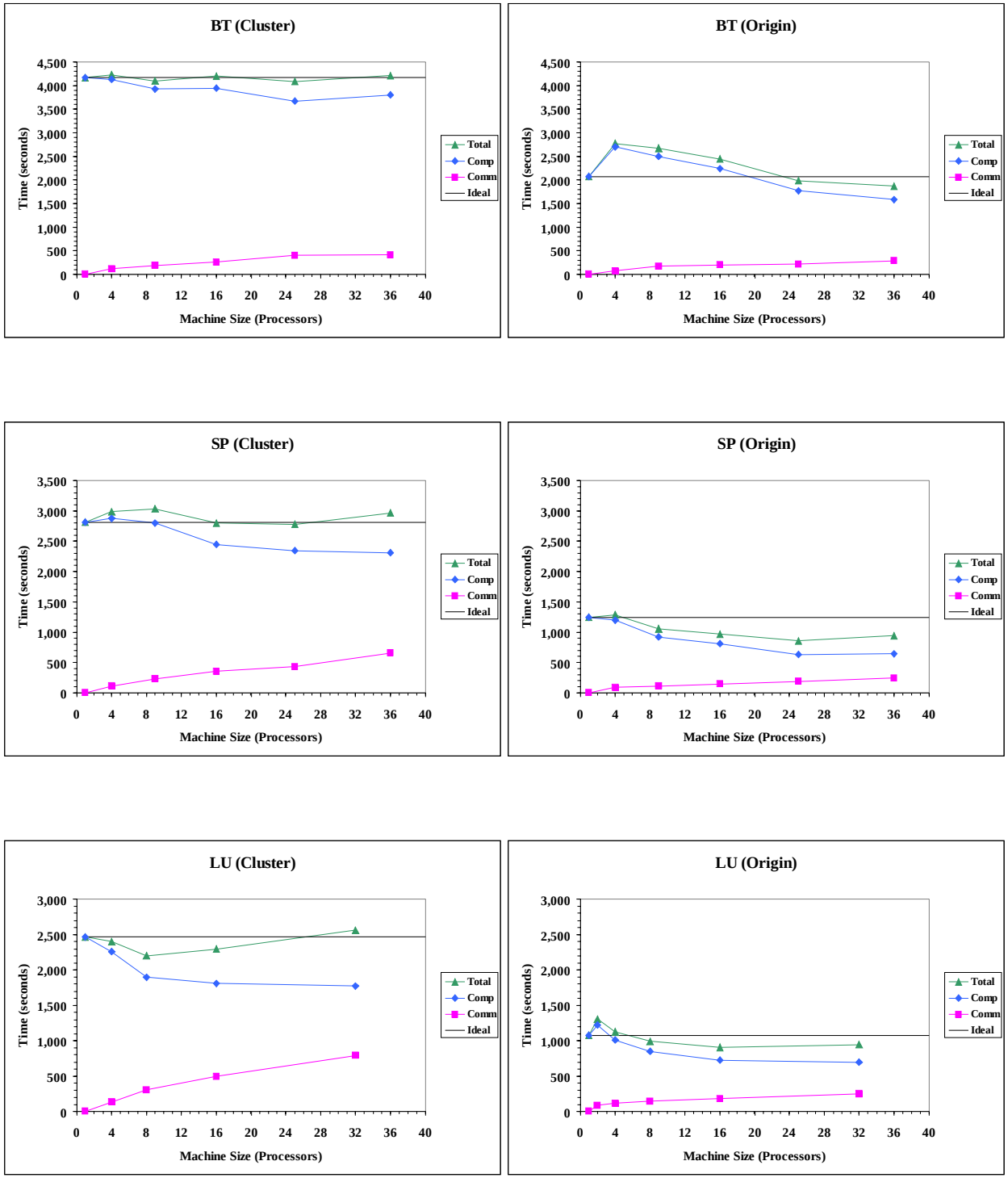


Figure 2 Time Breakdown for the NAS Benchmarks. These figures break down the total execution time, summed across all processors, for the NAS benchmarks, on the cluster and Origin. The communication and computation time are shown as separate (non-cumulative) lines; total time (the sum of communication and computation time) is presented as well.

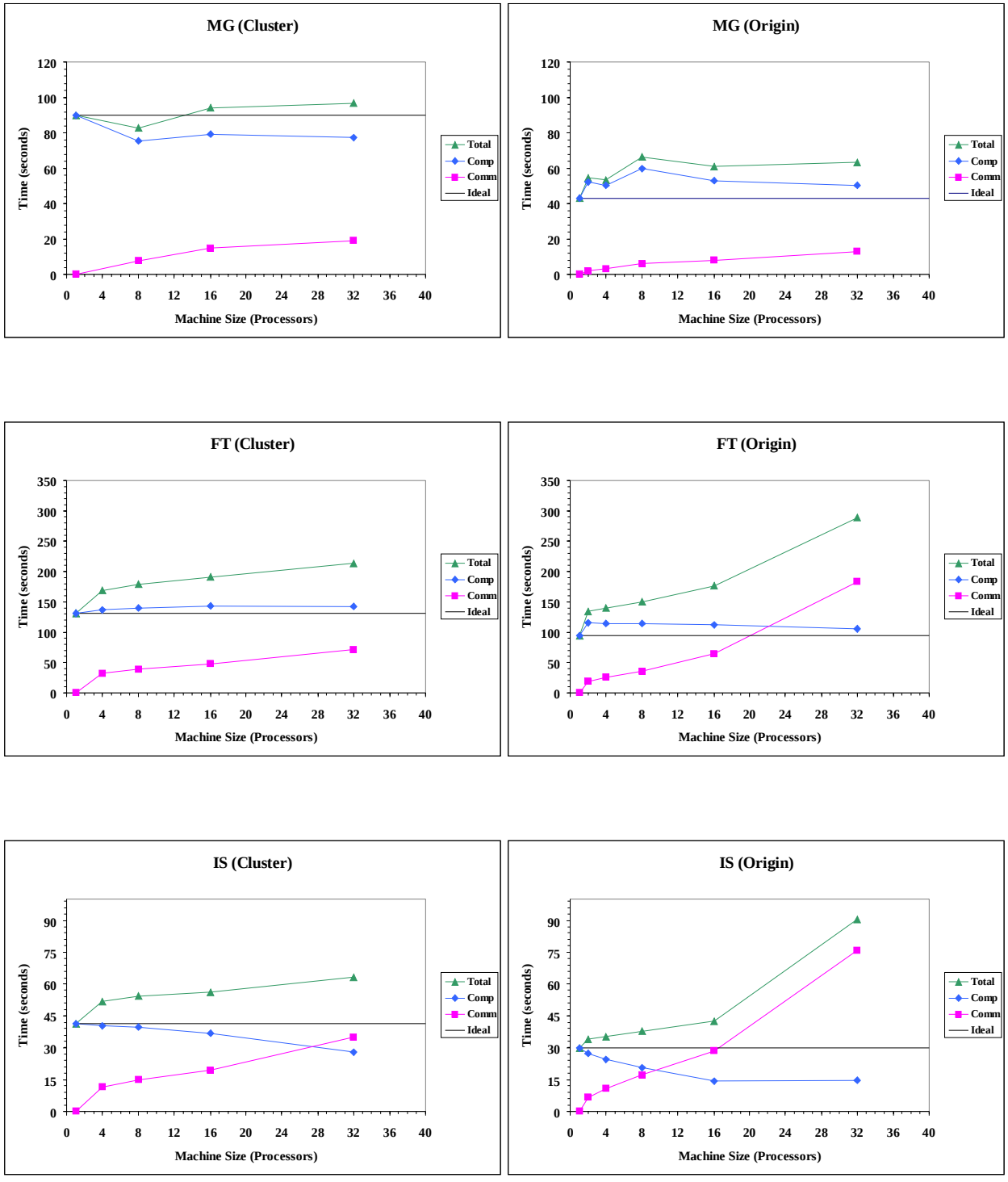


Figure 2 Time Breakdown for the NAS Benchmarks (continued). These figures break down the total execution time, summed across all processors, for the NAS benchmarks, on the cluster and Origin. The communication and computation time are shown as separate (non-cumulative) lines; total time (the sum of communication and computation time) is presented as well.

benchmarks. (Although we have also run Class B, the Class A problem size is the most useful basis for this study because Class B can not be run on a single processor of most parallel machines before 1998.) Observe that the cluster obtains near perfect speedup (*i.e.* slope = 1) for the benchmarks besides FT and IS, where it obtains about 2/3 efficiency. Indeed, for a few of the benchmarks, speedup is slightly super-linear for certain machine sizes. The behavior is far more complex on the Origin. The performance of several benchmarks is substantially super-linear in this range, while the performance of FT, IS and MG falls off to 2/3 or even 1/3 efficiency. The reason, as we shall see, has to do with cache effects as well as communication effects. This behavior would not appear in systems of a generation ago; it only occurs with the large second-level caches that are present in today's machines.

4 Where the Time Goes

The first step in understanding NAS benchmarks behavior is to isolate the components of application execution time. Of course, these are parallel programs, so we need to work with the time spent on all processors. The curves labeled "*Total*" in Figure 2 show the sum of the execution time over all processors of the NAS benchmarks on our two machines, as a function of number of processors. By this metric, a perfect speedup corresponds to a horizontal line (labeled "*Ideal*") with a *y*-intercept of the single processor execution time. For example, on the cluster, BT and SP follow the "*Ideal*" closely, whereas LU and MG drop below (*i.e.* super-linear speedup) for moderate machine sizes, and IS and FT rise above (*i.e.* sub-linear speedup). The Origin curves show even greater variation.

To understand this behavior, we isolate components of the execution time by instrumenting portions of the program. Although we have obtained detailed breakdowns, here we consider only the overall time spent inside and outside the MPI library. It is hard, in practice, to distinguish between the inherent load imbalance of the parallel program, the synchronization cost of a communication event, and the actual communication cost. The curves labeled "*Communication*" in Figure 2 show the sum of time spent in MPI for communication and synchronization (including send, receive, and wait time), and the curves labeled "*Computation*" show the sum of time spent outside the MPI library as the processor count increases. In general, we see that communication time grows with processor count, but often this is compensated by improvements in computational efficiency. Super-linear speedup is observed when the decrease in total computation time is more than the increase in communication cost.

MG and BT are computation bound. Communication is less than 20% of the total execution time at 32 and 36 processors configuration respectively. The benchmark BT experiences a modest linear improvement in computational efficiency with larger machine size. MG, on the other hand, has roughly constant efficiency beyond a few processors and the change relative to one processor is opposite on two machines. In SP and LU, changes in communication and computation cost roughly balance. Although communication time occupies one third of the execution time in both benchmarks, it is offset by the decrease in computation time.

IS and FT are communication bound. FT shows no gain in computational efficiency when run on more processors. In fact, the computation time increases slightly. The increase in communication time is so significant that it becomes the dominant factor in the overall speedup of the benchmark. Although the computation efficiency of the benchmark IS improves, the increases in communication time dominate the overall performance of the benchmark.

With the CPS scaling rule, the total amount of work (*i.e.* total number of computational operations to solve a fixed problem) remains the same regardless the number of processors. Therefore, the computation time on all processors should remain constant if the computational efficiency is unchanged. On the other hand, as more processors are added to solve the same problem, communication time increases, as does the number of computational operations due to redundant work. Nonetheless, many benchmarks show perfect or super-linear speedup. The extra time spent in communicating and synchronizing is more than compensated for by the improvement in computational efficiency. Using hardware counters, we have concluded that this reduction in computation time corresponds to a reduction in miss rate and in CPI.

In the following sections, we will investigate the factors that govern the speedup of the NAS benchmarks. In particular, Section 5 examines the change in computational efficiency caused by a more effective memory system as the total amount of cache increases. In Section 6, we examine the communication behavior of the NAS benchmarks.

5 Working Sets

To gain better insight into the memory access characteristics of the benchmarks under scaling, we obtained a per-processor memory address trace for each application

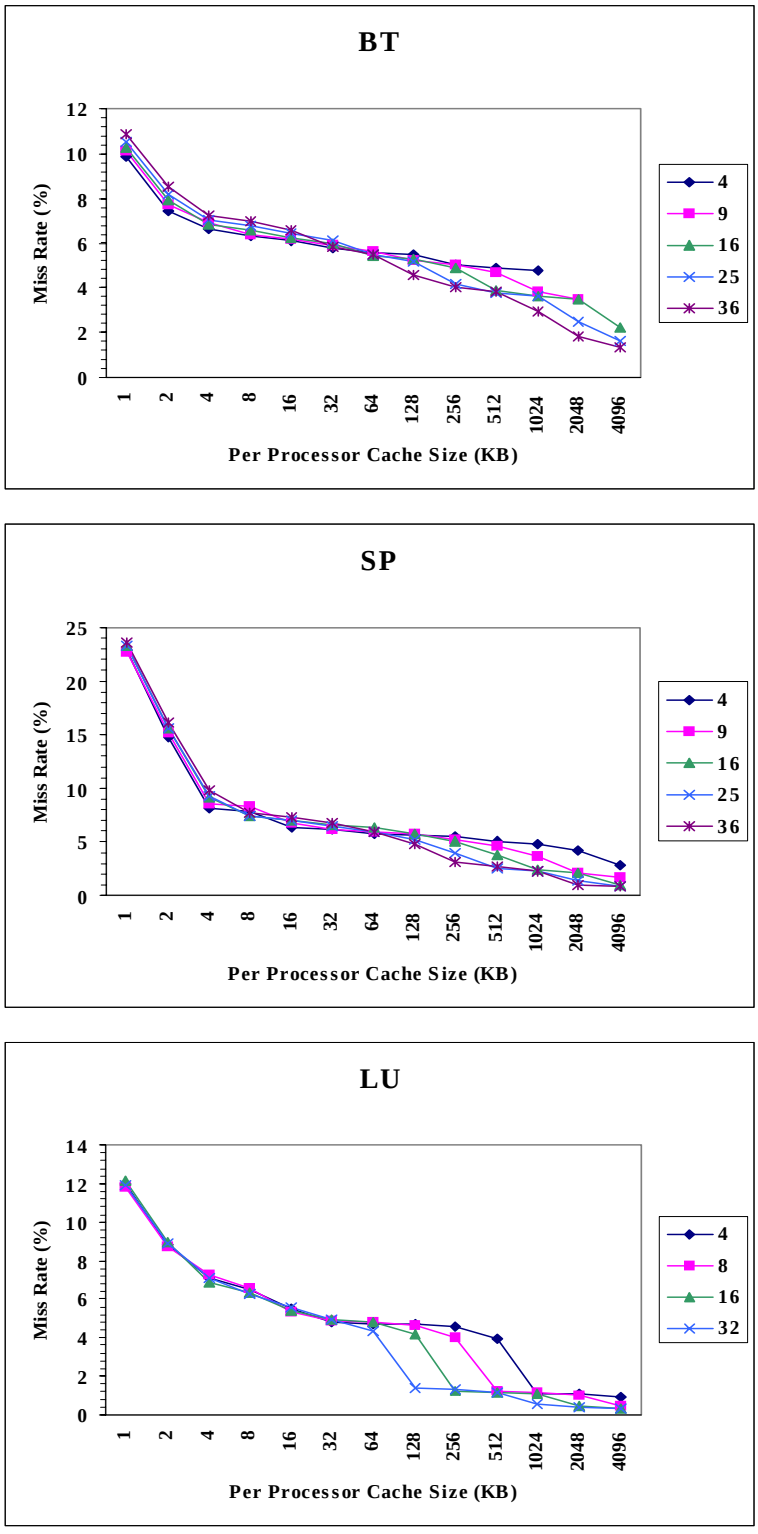


Figure 3 Working Set Profiles for the NAS Benchmarks. The working sets of the Class A problem size for BT (*top*), SP (*middle*) and LU (*bottom*) are presented. Each curve corresponds to a particular machine size (*e.g.* curve labeled "*4*" corresponds to a 4-processor machine). In all cases, at large cache sizes, increasing the number of processors working on the problem decreases the per-processor miss rate by a noticeable amount. At smaller cache size, scaling either makes no difference in cache performance (LU), or increases the miss rate (SP, BT).

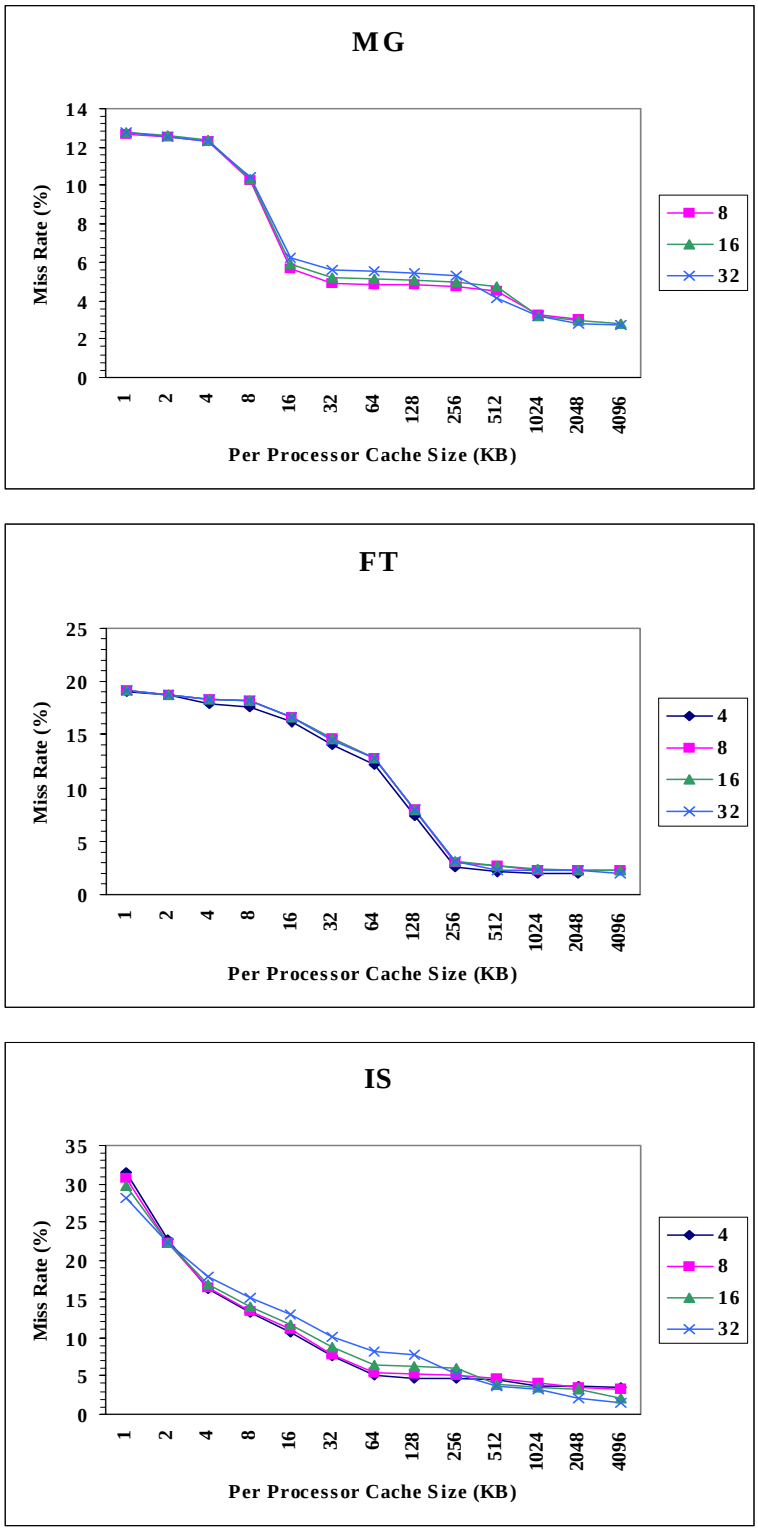


Figure 3 Working Set Profiles for the NAS Benchmarks (continued). The working sets of the Class A problem size for MG (*top*), FT (*middle*), and IS (*bottom*) are presented. For both MG and FT, the per-processor miss rate does not change significantly with any size of the cache when scales. IS has a slight improvement in cache miss rate at large cache sizes, but the miss rate at 2 KB to 256 KB per-processor cache size increases when scaled.

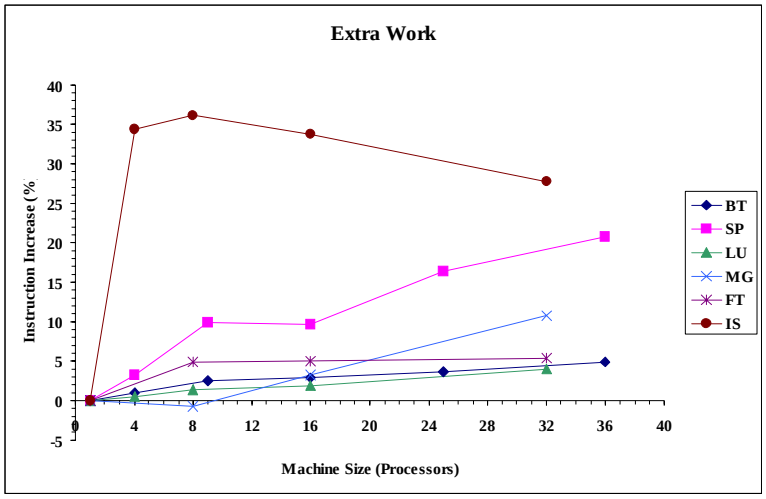


Figure 4 Extra Work. This figure shows the percentage increase in computational instructions (relative to a single processor) of the NAS benchmarks on the cluster with scales.

at each machine size of interest. We then ran the trace for one processor through a cache simulator for a range of cache size. For all simulations, the caches are: fully associative with LRU replacement, 64-byte blocks, write back, and write allocate.

The curves labeled "4" of Figure 3 show the data cache miss rate in a 4-processor machine, as a function of cache size for size between 1 KB and 4 MB. To demonstrate the effect of scaling on miss rates, let us concentrate on LU, where the effect is quite pronounced. For LU, we see the smooth decrease in miss rate (following the general rule that doubling the cache eliminates a constant fraction of the misses [1]) out to 32 KB, down from 12% to roughly 4%. The miss rate is flat to 256 KB, and then it drops from above 4% to below 1% and levels off. These "knees" of the working set correspond to that described in [16] to shared address space programs and measured for SPLASH-2 [20]. The key observation is that with CPS scaling, the working set curve is different for each machine size. With eight processors, the first knee in LU starts at 128 KB, with 16 processors it is at 64 KB, and with 32 processors it is at 32 KB. In all cases, the sharp drop occurs as the amount of global cache (*i.e.* the sum of the local caches) reaches 4 MB for the benchmark LU.

The memory access requirements of LU on the cluster is seen by drawing a vertical line at the per-processor cache size of 512 KB. The miss rate drops significantly from 4 to 8-processor system on the cluster and flattens out with larger configurations. This change is reflected in the change of the total computation time in Figure 2. On the Origin system with 4 MB of L2 cache, this first working set knee is captured by the cache on a single processor. There is an increase in computation on two processors due to other factors and a decrease for large configurations as the second working set fits in the global cache. As algorithms are tuned to be more cache friendly, like LU [21], this phenomenon will be more pronounced.

Other benchmarks, experience different levels of "boost" as global cache size increases. For example, the benchmark BT, SP and IS have moderate improvements in efficiency, whereas FT and MG have no significant change.

Interestingly, at the small local cache size for typical of early parallel machines, the miss rate for the most benchmarks increases with the number of processors, so there is no such improvement in computational efficiency. In particular, for the benchmarks SP, BT, and MG, machines with small caches would only see an increase in miss rate under scaling.

For all benchmarks, the amount of work increases as processors are added. Figure 4 shows the percentage increase in computational instructions on the cluster, relative to the single processor case. Most benchmarks experience moderate growth (5 to 10%) in instructions with scales, whereas IS and SP have significant increase in extra work. The load on the memory system is expected to increase with the instruction load.

The important point in examining cache effects is that they can have significant influence on the scalability of benchmarks under CPS scaling. While not a novel result, the increase in memory system efficiency due to

Benchmark	Isend	Send	AlltoAll	AllReduce	Barrier	Total Volume (MB)
BT	9600	0	0	0	0	1072.9
SP	19200	0	0	0	0	1876.2
LU	0	126000	0	2	0	459.1
*MG	0	4464	0	4	0	104.4
IS	0	0	20	10	0	320.2
**FT	0	0	7	6	0	896.0

Table 2 NAS Benchmarks Baseline Communication Characteristics. This table shows the baseline communication characteristics for the NAS Benchmarks codes on Class A problem sizes on 4-processor. The table shows the number of messages sent breakdown by type, and the total number of bytes sent. *The results for MG are for the 8-processor case. **FT uses Reduce instead of AllReduce.

cache effects is often overlooked or, in the case with the NAS benchmarks, are often dismissed because it is assumed that the working sets far exceed the cache size. However, our work demonstrates that with the combination of large caches (1 to 4 MB per processor) and more cache-friendly codes, cache effects can play a significant role in the scalability of a machine under CPS scaling rules. Indeed, in the case of the NAS benchmarks, the cache "boost" can mask poor performance in other areas, such as communication.

6 Communication Performance

From the breakdowns of execution times in Figure 2, we know that the benchmarks spend a significant amount of time in communication. In some benchmarks, like FT and IS, the increase in communication time primarily determines the scalability of the benchmarks. In this section, we further investigate the communication load the benchmarks place on the architecture, as well as the sensitivity of the benchmarks to the underlying communication protocol of MPI.

6.1 Communication Scaling

Table 2 shows the baseline communication characteristics of the NAS benchmarks on 4 processors. All other communication scaling is relative to this base case. The table shows the total number of messages sent by all processors (collective operation that performs on all processors is counted as one message). The "*Total Volume*" is the total number of bytes sent by all processors. At four processors, the benchmarks send few messages and the size of the messages are quite large; some of the benchmarks send messages that are megabytes in size. BT and SP use the non-blocking `MPI_Isend` primitive, whereas LU and MG use the traditional blocking `MPI_Send`. The number of collective communication operations is also low. Surprisingly, no benchmark uses barriers in the benchmarked portion of the code. Most benchmarks use reduce operations, but the reduce operations do not contribute to overall performance significantly. Only FT and IS communicate primarily with collective communication; both do a series of all-to-all exchanges.

Figure 5 shows how the communication characteristics of the NAS benchmarks change with machine sizes. Figure 5 (*top*) plots the change in total message count per processor as a function of the machine sizes, normalized to the message count on 4-processor. Figure 5 (*middle*) shows the byte count per processor. Finally, Figure 5 (*bottom*) shows the resulting average message size per processor. Within the realm of interest, there is an order of magnitude difference in the average size of a message for each benchmark. Interestingly, the smallest messages on most benchmarks (except MG) are still on the order of 1000 bytes, which is a substantially larger grain than found in many other parallel benchmarks [12, 20].

Because of their all-to-all pattern, for both IS and FT, the normalized per-processor message count growth linear with machine size, so the total number of point-to-point messages increases as the square of the number of processors. Since the total byte volume remains constant and therefore the bytes per processor decreases as $O(1/P)$ and the message size decreases as $O(1/(P^2))$. Although for the range of processors studied the abso-

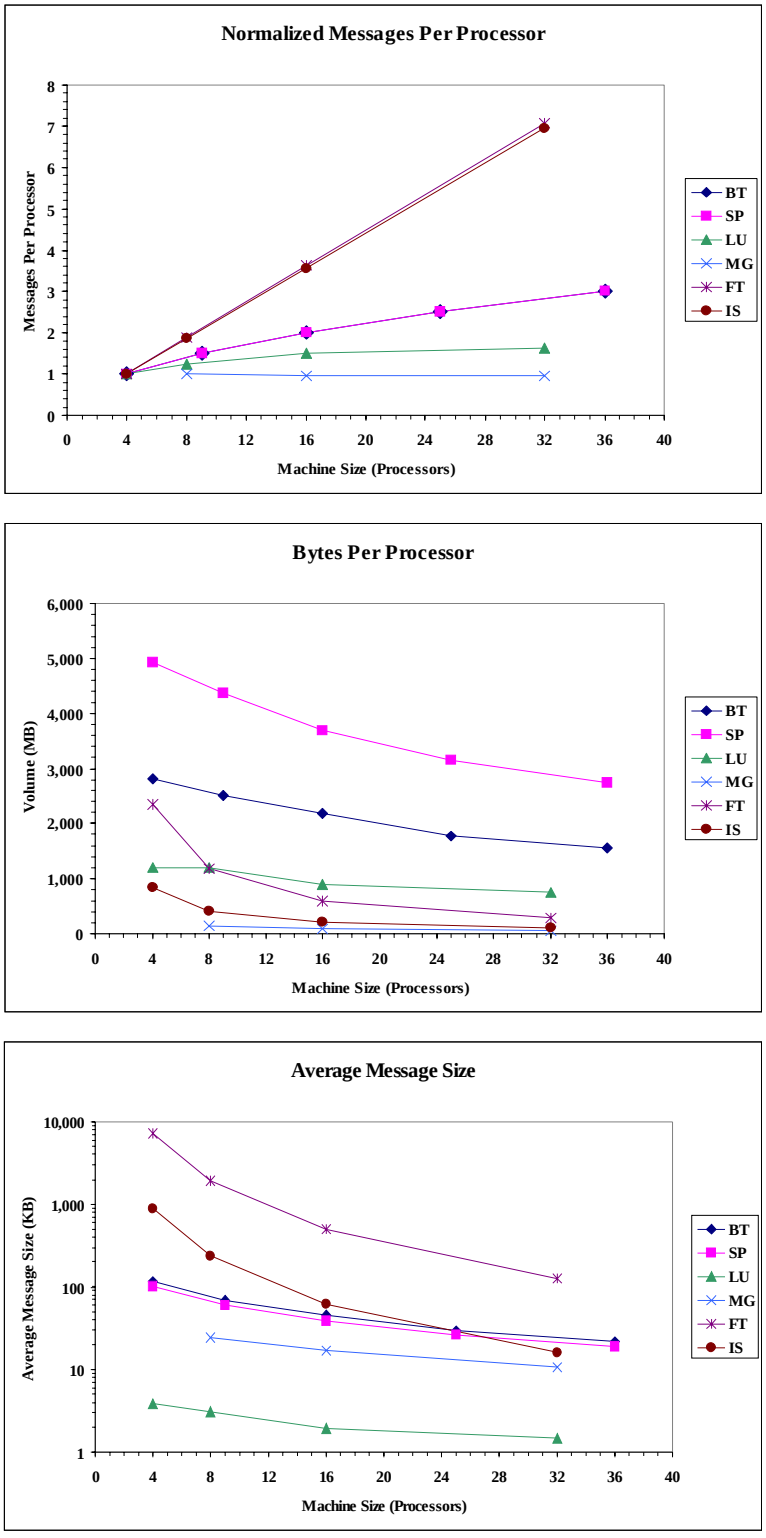


Figure 5 Message Scaling. This figure shows how the number of messages sent normalized to the 4 processor case (*top*), number of bytes sent (*middle*) and average size of a message (*bottom*) scale as a function of processors.

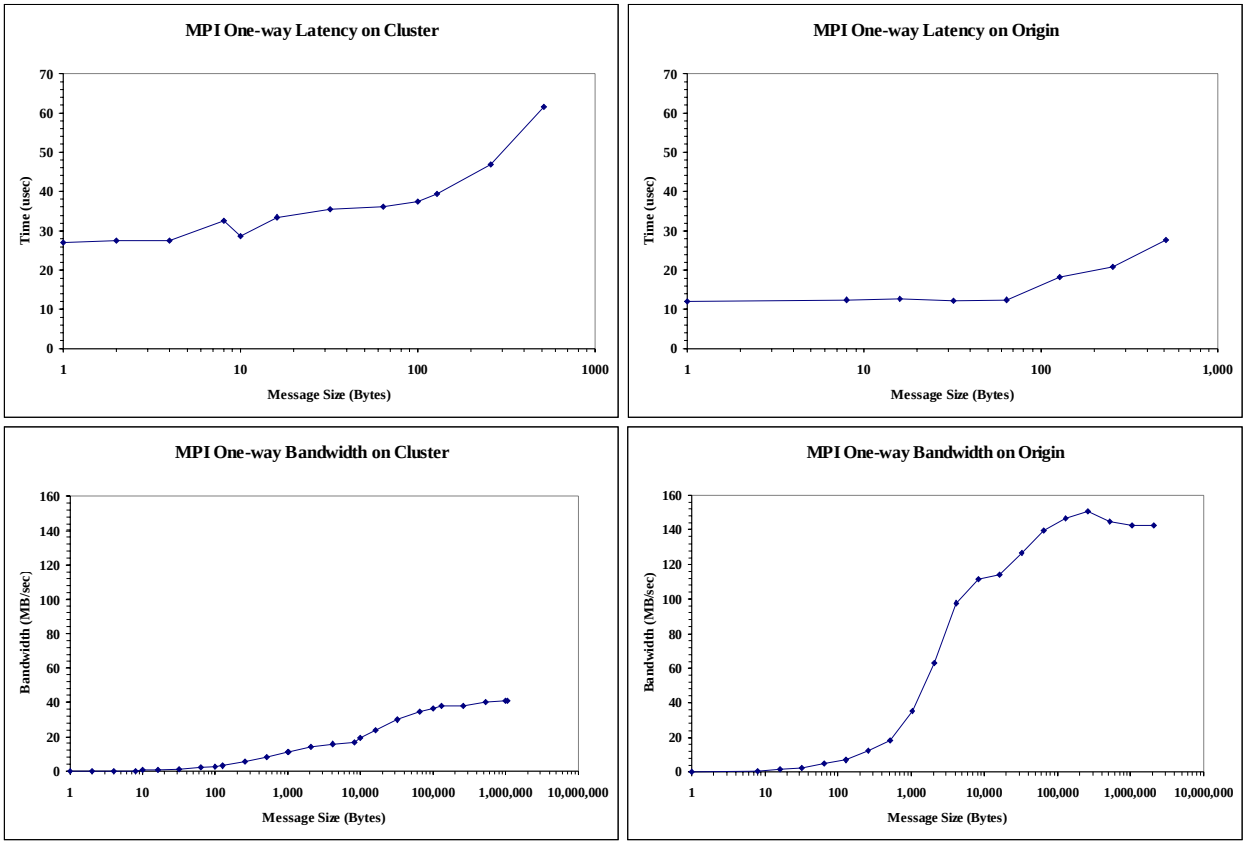


Figure 6 MPI Performance. These figures show the performance of MPI on both platforms using Dongarra's echo test. Top figures show the one-way latency of small messages. Bottom figures show the one-way bandwidth with message size up to 1 MB.

lute number of messages remains relatively small, the squaring of the message count and resulting decrease in message size has important implications for machine designers. For machines of up to 100's of processors, efficient transfer mechanisms must exist for messages ranging from a few hundred bytes to megabytes in size.

For BT and SP, the total amount of communication follows surface to volume ratio as we scale the number of processors. Unlike the other benchmarks, LU and MG use finer-grained communication. However, the message sizes of these benchmarks span an order-of-magnitude range as well. MG, in paticular, sends messages ranging from 8 bytes to 100 KB. For the range of processors of interest, the scaling of communication along these lines does not unduly limit speedup. The spatial decomposition keeps communication in the nearest-neighbor regime for these benchmarks as well.

In summary, the NAS benchmarks place a wide variety of communication loads on the system, ranging from nearest-neighbor point-to-point exchange to coarse-grained all-to-all communication. In general, the communication load increases when scale.

6.2 Cluster Sensitivity to Communication Protocol

The message characteristics imply that total communication costs should increase under CPS as we scale the machine size. Figure 2 shows that indeed, total communication costs rise, however, there are sizable differences in how each platform handles the increased communication load.

Figure 6 plots the MPI one-way latency and bandwidth on both platforms using Dongarra's echo test [7]. One-way latency is half of the message round-trip time and one-way bandwidth is the reciprocal of the latency. The startup cost is 27 µsec and 13 µsec with a maximum bandwidth of 41 MB/sec and 150 MB/sec on the cluster and Origin respectively.

Using the micro-benchmarks results and message characteristics of the NAS benchmarks, we can construct the expected communication cost. For each message, we accumulate the micro-benchmark latency at the message size. This gives us the predicted communication time of the NAS benchmarks. Communication efficiency is the ratio predicted to measured communication

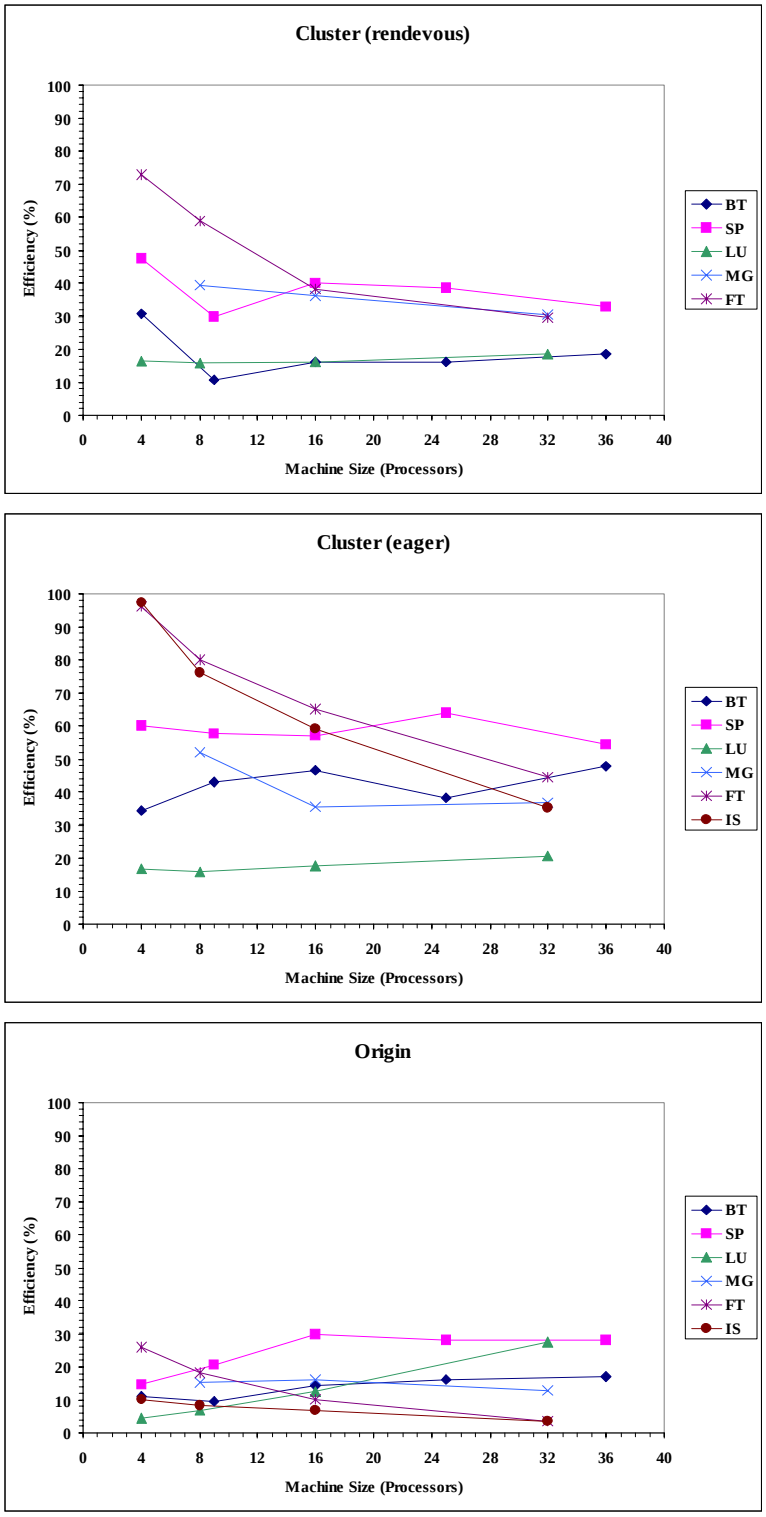


Figure 7 Communication Efficiency These figures show the percentage of predicted bandwidth delivered by the MPI layer over the total time spent in sends and waiting for each application as a function of the number of processors. Top figure shows the communication efficiency using the rendevous protocol in the MPI layer on the cluster, and the middle figure shows the communication efficiency using the eager protocol. The bottom figure shows the communication efficiency on the Origin 2000 machine.

time. Figure 7 graphs the communication efficiency on both platforms. Notice that on the cluster (*rendevous*), most benchmarks have an efficiency of one half and drop slightly with scale. FT, on the other hand, starts with high efficiency (75%) but falls off sharply with scale. The communication efficiency of all benchmarks on the Origin is below 35%. Although the figures show that the cluster platform handles the load better than the Origin, both show that the delivered performance is well below the micro-benchmark performance.

One possible cause of this anomaly is the implementation of the MPI layer and how it interacts with the underlying architecture. The evaluation of NAS benchmarks drove the development of MPI layer on the cluster, because the total time predicted by combining the micro-benchmark performance with the message profile data was significantly less than the time actually spent in communication. Further investigation on the cluster revealed that the source of the problem was the internal protocol of the MPI layer. Our initial implementation of MPI used a conservative *rendevous* protocol. Since we were using low-latency Active Messages as a building block, using a rendevous protocol simplified both the receive code and the message format. The short Round Trip Time (RTT) is easily amortized by a large impending bulk transfer. Under micro-benchmarking conditions, this design does deliver near optimal performance. In practice, however, queueing delays at the source MPI-to-network interface exacerbate the RTT on real applications and it resulting efficiency is as shown in Figure 7 (*rendevous*). Figure 8 (a) plots the histogram of measured round-trip times for the protocol message during a run of the BT benchmark on 36-processor. Although in micro-benchmark tests, the RTT is about 50 μsec, the actual mean cost is 5 ms! The variance is also quite high indicating that prediction of round-trip times would be difficult. Because of the large average message size of the NAS benchmarks, the protocol messages often experience long queueing delay which surface at the application level in the form of low communication efficiency.

We revised the MPI implementation to use a more aggressive *eager* protocol. This significantly increased the complexity to re-sequence out-of-order messages and has slightly worse micro-benchmark performance, but the communication performance in the context of real applications improved by as much as 100% in certain benchmarks as shown in Figure 7 (*eager*). The use of the eager protocol increases the utilization of the outgoing channel by reducing queueing delays. Figure 8 (b) plots the histogram of measured communication time for 45 KB message of BT running on 36-processor using the eager protocol. The figure shows that the cost of 45 KB message is reduced significantly with smaller variance.

All of the benchmarks, except LU, achieve higher communication efficiency with the eager protocol. The benchmarks FT and IS at 4-processor achieve near full efficiency. The communication efficiency falls off with larger configurations as the network becomes saturated. For the benchmarks BT, SP and MG, the communication efficiency is limited by point-to-point communication, as they primarily use `MPI_Send` and `MPI_Isend`. The improvement of communication efficiency using the eager protocol is most effective in these benchmarks. In particular, the communication efficiency of the benchmark BT increases from 20% to 40%, whereas in SP, the efficiency improves from 40% to 60%.

The almost unchanged in efficiency of the benchmark LU is presumably caused by the inherent load imbalance of the program. An investigation shows that the benchmark experiences approximately a 25% load imbalance across the 32 processors, which suggests that the improvement in point-to-point communication is hidden from the large amount of synchronization time.

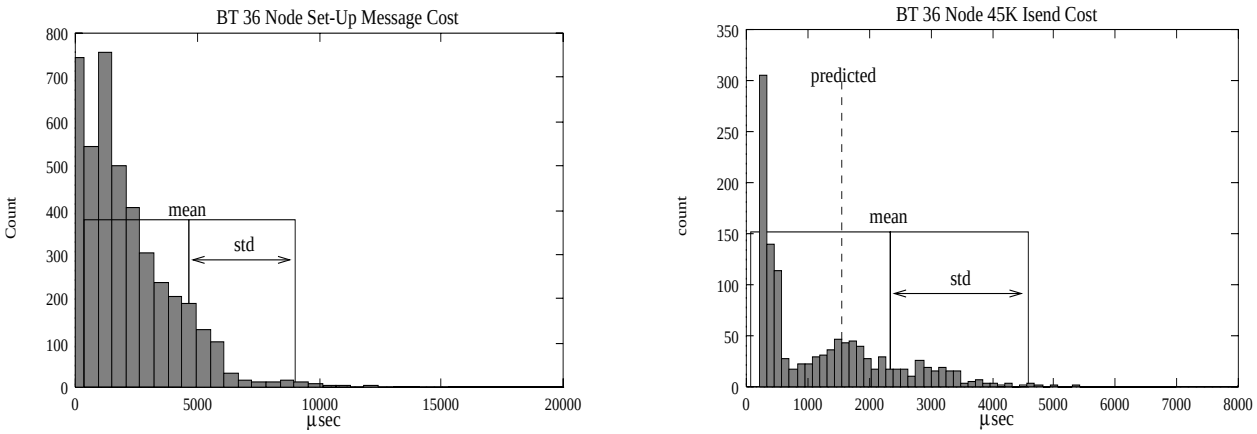
7 Origin Sensitivity to Processor Speed

Our initial study on the Origin system was based on R10000-195MHz processor. When the Origin system at NASA Ames was upgraded to use a R10000-250MHz over the course of our investigation, the change in behavior was quite illuminating. In this section, we study the differences in performance of the NAS benchmarks on these two systems.

The only difference in the two systems is a 25% increase in processor speed. Since on-chip cache latency and memory bus speed is closely tied to the processor speed, the performance of the memory system and the MPI will change as well. First, we use micro-benchmarks to capture the differences in these two systems that experience by user applications.

Figure 9 shows the performance of the memory systems using the memory stride benchmark [17]. The benchmark shows that there are approximately 25% decrease in latency for both L1 and L2 cache, and an approximately 20% improvement in latency to the main memory.

Figure 10 shows the one-way point-to-point bandwidth of the MPI on the 195 MHz system using Dongarra's echo test. Despite the improvement in processor speed and the memory system, the micro-benchmark obtains roughly the same maximum bandwidth with only 10% decrease in latency for small messages. The



(a) **(b)**

Figure 8 Message Cost with Different MPI Internal Protocols. These figures show the change in message costs with different MPI layers. Figure (a) plots the histogram of observed round-trip times for the initial MPI layer's set-up message for 1 node of the BT benchmark on 36-processor. Figure (b) plots time taken by `MPI_Isend` using the eager protocol for a 45 KB message. The mean and standard deviation are shown for both graphs. The tails of the distributions (60,000 µsec and 20,000 µsec , respectively) have been omitted for clarity.

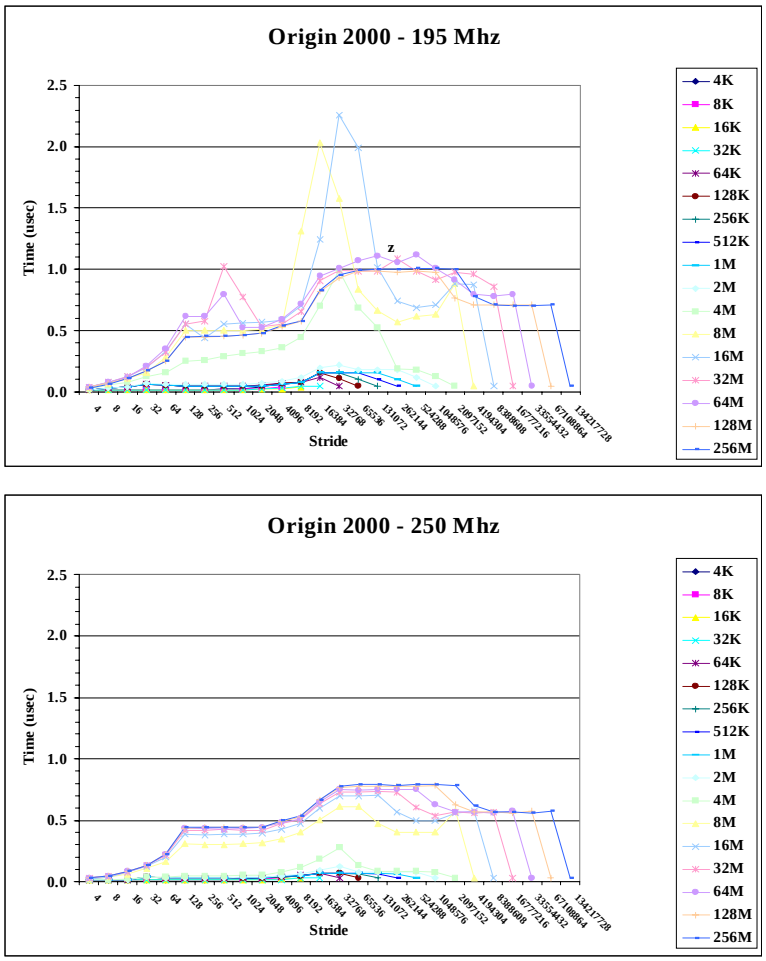


Figure 9 Memory System Characteristics. These figures show the memory system characteristics of the Origin 2000 machines with 195 MHz and 250 MHz processors.

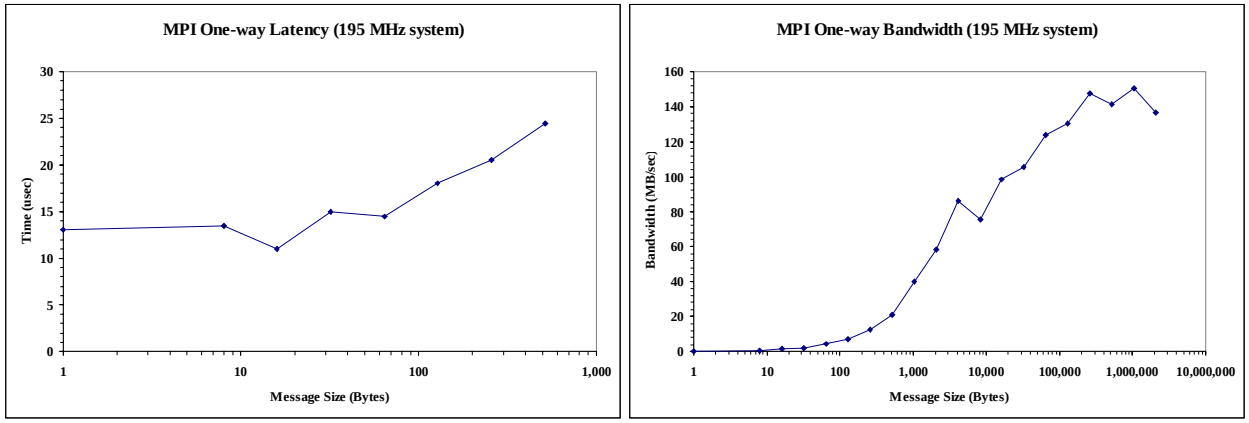


Figure 10 MPI Performance. These figures show the MPI point-to-point performance of the 195 MHz Origin system using Dongarra's echo test.

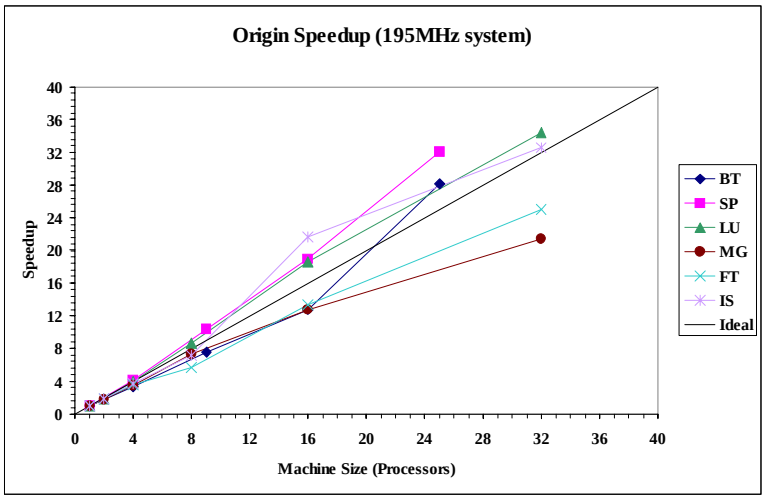


Figure 11 NAS Benchmarks Performance on 195MHz Origin machine. This figure shows the speedup of the NAS Benchmarks on the Origin 2000 195 MHz machine.

195 MHz system has a one-way latency of 12.99 µsec and achieves the maximum bandwidth of 150 MB/sec at 1 MB message size. The 250 MHz system (Figure 6) has a one-way latency of 11.87 µsec and obtains the maximum bandwidth of 150 MB/sec at 256 KB message size.

7.1 NAS Benchmarks Performance

Figure 11 shows the speedup of NAS benchmarks on the older Origin. Single processor performance of all benchmarks on the newer Origin are increased by 20% except FT, where a 40% improvement is observed. Most benchmarks have about 10 to 15% improvement at larger configurations, except FT and IS, where the runtimes at 32-processor are higher on the 250 MHz system. Most benchmarks have roughly the same speedup except FT and IS. The speedup of the benchmarks FT and IS drops from 2/3 efficiency and super-linear respectively, down to about 1/3 of efficiency.

7.2 Sensitivity to Workload

All measurements present in this paper are done in a dedicated environment, *i.e.*, only one program is run at any given time. On Origin machine, this meant running at odd hours when the load was very low. We found that even when there is no time-sharing between workloads,

the execution time of the benchmarks are significantly higher in a multi-workload environment than in a dedicated environment. For example, the average runtime of benchmark IS with 8-processor is 4.71 seconds and the average runtime of the benchmark SP with 36-processor is 26.36 seconds in dedicated mode. When the two benchmarks are run together using only 44 out of 64 available processors, the execution time of the benchmarks increases to 6.18 seconds and 65.28 seconds respectively, the execution time profile of IS shows that the communication time increase of average 1.4 seconds per processor, whereas the communication time of SP increases by a factor of 5. These unexpected results suggest that the communication of the benchmarks interfere, which probably also leads to higher synchronization cost.

8 Conclusion

A detailed analysis of the architectural requirements of the NAS benchmarks shows that while several of the benchmarks perform a non-trivial amount of communication, a reasonably scalable communication system can in principle handle the communication load. The dominant factor in base performance and in scalability is the sequential node architecture, including the CPU, caches, and the local memory system. What is particularly important is how the node architecture interacts with the application requirements under CPS scaling.

For communication, we found that even though the applications are carefully designed to perform coarse-grained communication, the efficiency of communication is lower than expected. Interestingly, the Origin, in spite of the availability of fine-grained shared memory for data transport, achieves fairly low communication efficiency, in some cases spending more time in communication than the cluster.

One result of our work is a word of caution with regards to common assumptions about machine architecture and scalability. One may be tempted to judge the communication ability of a machine based on speedup of the NAS benchmarks: good speedup implies good communication and conversely poor speedup implies poor communication. However, the NAS benchmarks are not necessarily defined by the scalability of the communication system. For example, the Origin has super-linear speedups, but relatively poor communication scalability. Rather, one must examine both the computation and communication scaling of a parallel machine in order to judge a machine's effectiveness in these areas.

Understanding the scaling and performance characteristics of large parallel machines is a difficult problem. The NAS Parallel Benchmarks are a critical step towards this goal, providing a set of common benchmarks for comparison among platforms. However, the current output of the benchmarks is only execution time under scaling (plotted as time, speedup, or efficiency), which does not reveal the complexities of the benchmarks on different processor counts. Lightweight instrumentation should be added to the standard MPI libraries, and minimally should report the time spent in computation versus communication. This simple breakdown would give users better insight as to the nature of processor versus network performance for a given machine.

Acknowledgments

We would like to thank William Saphir of Ernest Orlando Lawrence Berkeley National Laboratory and Mary A. Hultquist of NASA Ames Research Center for their help in obtaining the account on the Origin 2000 in the Numerical Aerospace Simulation Facility at NASA Ames Research Center. Also, we would like to thank Dan Lenoski of SGI and Mark Straka of NCSA for helping us in understanding issues with the R10000 performance counters. And finally, we would like to thank Shirley Chiu, Alan Mainwaring and David Wu for their many helpful comments and discussions on this work. This research has been supported in part by the DARPA (F30602-95-C0014), the NSF (CDA 94-01156), the DOE (ASCI 2DJB705), and the California MICRO program.

Reference

[1] Agarwal, A., Horowitz, M., and Hennessy, J., “An Analytical Cache Model”. *ACM Trans. on Comp. Sys., Vol.7, no.2*, May 1989, pp. 184-215.

[2] T. E. Anderson, D. E. Culler, D. A. Patterson, and the NOW Team. A Case for NOW (Networks of Workstations). *IEEE Micro*, February, 1995.

[3] David H. Bailey, T. Harris, Rob Van der Wigngaart, William Saphir, Alex Woo, and Maurice Yarrow. The NAS Parallel Benchmarks 2.0. *Technical Report NAS-95-010, NASA Ames Research Center*, 1995.

[4] Nanette J. Boden and Danny Cohen and Robert E. Felderman and Alan E. Kulawik and Charles L. Seitz and Jakov N. Seizovic and Wen-King Su. Myrinet -- A Gigabet-per-Second Local-Area Network. *IEEE Micro, Volume 15 number 1*, Feb. 1995. pp.29-38.

[5] Bob Cmelik and Doug Keppel. Shade: A Fast Instruction-set Simulator for Execution Profiling, In *Proceedings of SIGNETRICS 94*, pp. 128-137.

[6] Leonardo Dagum, David H. Bailey, Eric Barszcz and Horst D. Simon. NAS Parallel Benchmarks Results. *Technical Report RNR-93-016, NASA Ames Research Center*, 1993.

[7] Dongarra, and T. Dunnigan. Message Passing Performance of Various Computers. *University of Tennessee Technical Report CS-95-299*, May 1995.

[8] T. von Eicken, D. Culler, S. Goldstein, and K. Schauser, ``Active Messages: a Mechanism for Integrated Communication and Computation", In *Proceedings of the 19th International Symposium on Computer Architecture*, May 1992, Gold Coast, Qld., Australia, pp.256-266.

[9] W. Gropp and E. Lusk and N. Doss and A. Skjellum. A high-performance, portable implementation of the (MPI) message passing interface standard. *Parallel Computing* 22(6):789-828, September 1996.

[10] Mark Hill. The Dinero Cache Simulator. Aug. 1995, http://www.cs.wisc.edu/~larus/warts.html

[11] A. Mainwaring. Active Message Application Programming Interface and Communication Subsystem Organization. *University of California at Berkeley, Computer Science Department, Technical Report UCB CSD-96-918*, October 1996.

[12] Richard P. Martin, Amin M. Vahdat, David E. Culler, and Thomas E. Anderson. The Effects of Latency, Overhead and Bandwidth in a Cluster Architecture. In *Proceedings of the 24th International Symposium on Computer Architecture*, June 1997.

[13] Message Passing Interface Forum. The MPI Message Passing Interface Standard. *Technical Report, University of Tennessee*, Knoxville, April 1994.

[14] NASA Ames Research Center. NPB 2 Detailed Results, 1997. http://science.nas.nasa.gov/Software/NPB/NPB2Results.

[15] Steven K. Reinhardt and Mark D. Hill and James R. Larus and Alvin R. Lebeck and James C. Lewis and David A. Wood. The Wisconsin Wind Tunnel: Virtual Prototyping of Parallel Computers, In *SIGMETRICS 93*. May, 1993.

[16] Edward Rothberg, Jaswinder Pal Singh, and Anoop Gupta. Working Sets, Cache Sizes and Node Granularity Issues for Large Scale Multiprocessors. In *Proceedings of the 20th International Symposium on Computer Architecture*, pages 14-25, May 1993.

[17] Saavedra-Barrera, R.H., CPU Performance Evaluation and Execution Time Prediction Using Narrow Spectrum Benchmarking, Ph.D. Thesis, *UC Berkeley, Technical Report No. UCB/CSD 92/684*, February 1992.

[18] William Saphir, Alex Woo, and Maurice Yarrow. NAS Parallel Benchmark 2.1 Results. *Technical Report NAS-96-010, NASA Ames Research Center*, 1996.

[19] Elisabeth Wechsler. NAS Parallel Benchmarks Set The Industry Standard for MPP Performance. *NAS News*, Jan - Feb, Volume2, Number 8, 1995. http://science.nas.nasa.gov/Pubs/NAnews/98/01/Benchmark.html

[20] Steven Cameron Woo, Moriwoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24--36, June 1995.

[21] Maurice Yarrow and Rob Van der Wijngaart. Communication Improvement for the LU NAS Parallel Benchmark: A Model for Efficient Parallel Relaxation Schemes. *Technical Report NAS-97-032, NASA Ames Research Center*, November 1997.