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

Abstract

We present a study of the architectural requirements and scalability of the NAS Parallel Benchmarks. Through direct measurements and simulations, we carefully identify the factors which affect the scalability of benchmark codes on two relevant and distinct platforms; a cluster of workstations and a CC-NUMA 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 sets of the benchmarks, we are able to visualize the improvement of computational efficiency through CPS scaling.

We also find that, while the Origin MPI has better point-to-point performance, our cluster MPI layer is more scalable with communication load. Communication performance within the applications is often much lower than what would be expected from micro-benchmarks. We show that the communication protocols used by MPI runtime library are influential to the communication performance in applications.

1 Introduction

The NAS Parallel Benchmarks (NPB) are widely used to evaluate parallel machines [12]. To date, every vendor of large parallel machines has presented NPB results, at least with the original "paper and pencil" version 1.0 [3]. Those reports provide a comparison of execution time as a function 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 MPI [8] programming model of NPB2 [2] 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 truly available independent variable is total execution time [9, 11].

When we measured the NPB2 on the Berkeley NOW [1], 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 [5, 6], 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, but 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 (Version 2) 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 inherent computation, communication, and synchronization 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

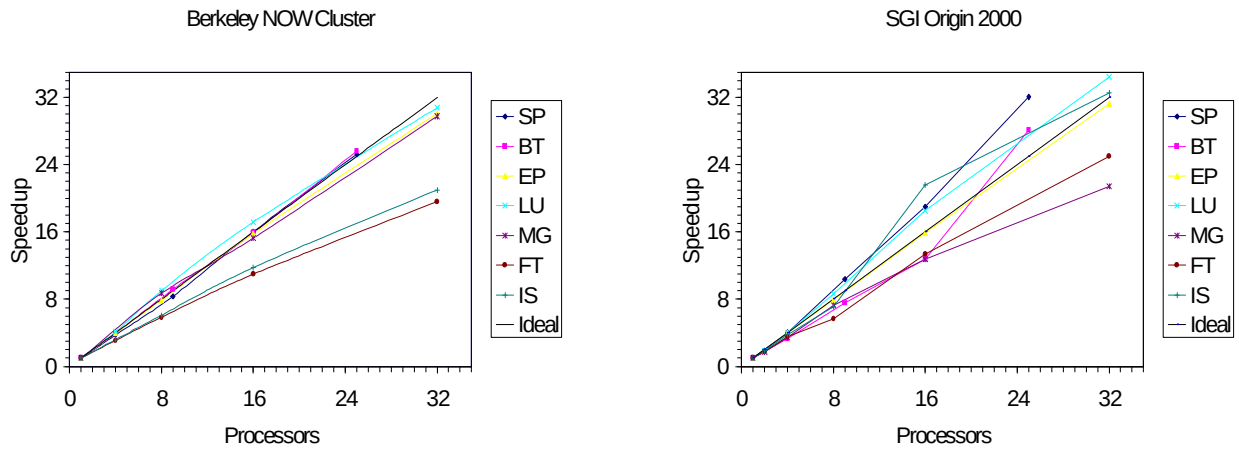


Figure 1 Speedup for NOW and SGI Origin on NPB2. This figure presents the speedup curves for the Berkeley NOW cluster and SGI Origin 2000 on the Class A problem size of the NAS Parallel Benchmarks, version 2.2. The Origin attains super-linear speedup for most of the benchmarks, whereas the NOW achieves linear or slightly above linear speedup for all but two programs.

Table 1: Single-Processor Execution time. This table presents single node runtimes on the Berkeley NOW cluster and the SGI Origin 2000 for the Class A NPB. For most of the programs, the Origin has roughly twice the processor performance.

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 Benchmark suite version 2.2, including the first detailed study of NAS working set behavior under scaling, and (4) an evaluation on communication efficiency of applications with different MPI communication protocols.

2 Speedup

Figure 1 and Table 1 show the speedup and single-processor execution time, respectively, that we measured on the Berkeley NOW cluster and the SGI Origin 2000 on the class A problem sizes of the NPB 2.2. Although we have also run Class B, the class A problem size is the most useful basis for this study because Class B cannot be run on a single processor of most parallel machines before 1998. Observe that the NOW obtains near perfect speedup for the benchmarks besides FT and IS, where it obtains about 2/3 efficiency. Indeed, for a few of the benchmarks, speedup is

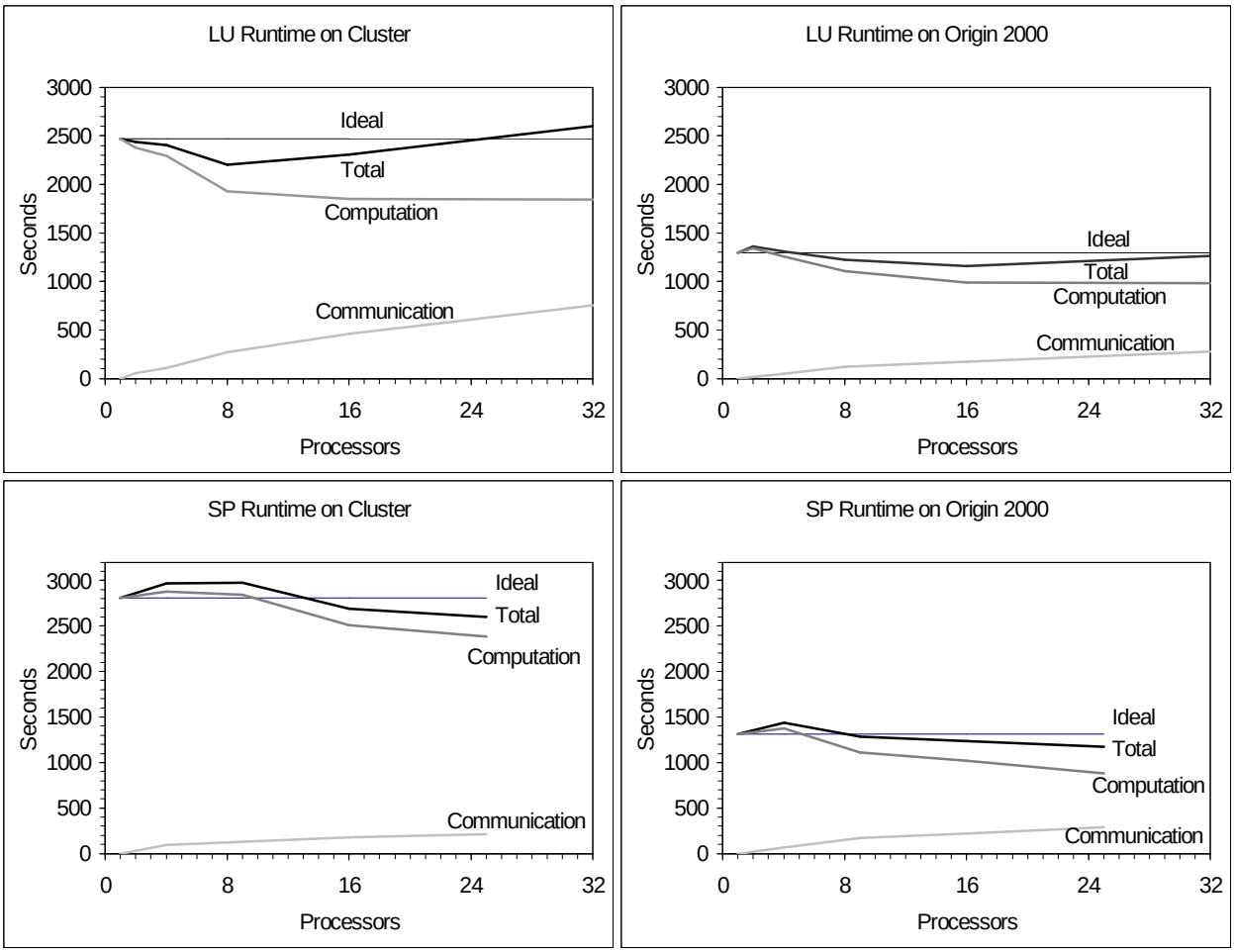


Figure 2 Time Breakdown for the LU and SP. This figure breaks down the total execution time, summed across all processors, for both LU and SP, on both the NOW and Origin. The communication and computation time are shown as separate (non-cumulative) lines; total time (the sum of communication and computation time) is presented as well. Note that where the total time decreases, super-linear speedup is observed.

slightly super-linear for certain machine sizes. The behavior is far more complex on the SGI Origin. The performance of several of the benchmarks is substantially super-linear in this range, while the performance of FT and MG falls off to 2/3 efficiency. The reason, as we shall see, has to do with cache effects. These effects are most pronounced on the Origin, even though the NPB use message passing as the communication paradigm, as opposed to shared memory. The cache effects are also pronounced on the larger class B sizes as well. Moreover, 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.

3 Where the Time Goes

The first step in understanding NPB 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 by all the processors. The NPB are so well balanced (within 5% across processors) and have good enough efficiency that we adopt an unusual perspective. We will examine the sum of the execution time across all of the processors, as a function of the number of processors. The curves labeled “Total” in Figure 2 show this for the LU and SP benchmarks on our two machines. (The full paper has the remaining applications.) By this metric, ideal speedup corresponds to a horizontal line, with a y-axis value of the single processor execution time. Indeed, in the four examples of Figure 2 there are regimes where the total time curve decreases!

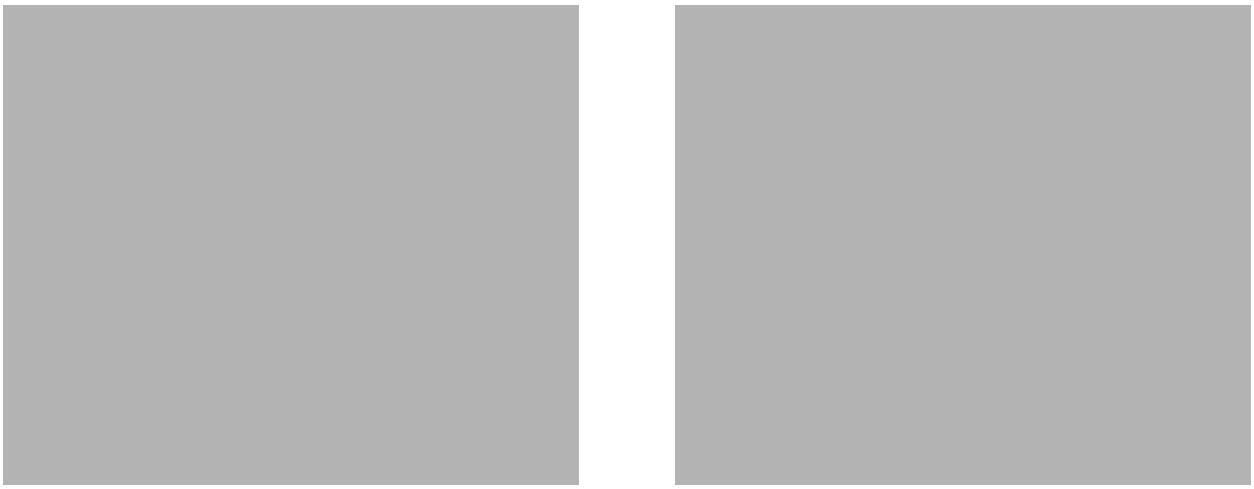


Figure 3 Working Set Profiles for LU and SP. The working sets of the class A input for LU (left) and SP (right) are presented. In both cases, at large cache sizes, increasing the number of processors working on the problem decreases the per-processor miss rate by a noticeable amount. At smaller cache sizes, scaling either makes no difference in cache performance (LU), or increases the miss rate (SP). The traces were collected with the Shade instruction-set simulator, and processed with the Dinero cache simulator.

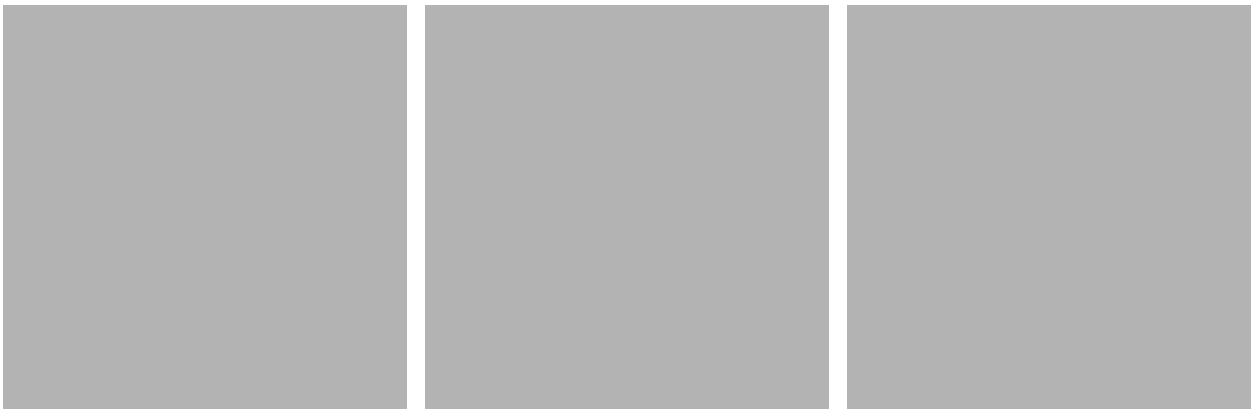
We can isolate components of the execution time by instrumenting portions of the program. Although we have obtained detailed breakdowns, here we consider only the overall time spent inside the MPI library and outside. The lowest curves (labeled "Communication") in Figure 2 show the total time spent in MPI for communication and synchronization (including send, receive, and wait time). The full paper will give a detailed assessment of the communication characteristics, which shows that the message frequency increases by more than an order of magnitude, the message sizes decrease, and the total communication volume increases slowly or is constant, depending on the application. (The communication is also well balanced across destinations.) The main point is that the time spent in communicating and synchronizing increases with the number of processors. The total amount of computation also increases due to redundant work. Nonetheless, the speedup is perfect. The extra time spent in communicating and synchronizing is more than compensated for by the improvement in computational efficiency as the number of processors increases. The curves labeled "Computation" in Figure 2 show the time spent in computation (outside the MPI library) as the processor count increases. This time decreases substantially in most cases. Using hardware counters, we have shown that this reduction in computational time corresponds to a reduction in miss rate and in CPI. All benchmarks except FT exhibit similar behavior.

Notice that these benchmarks are not embarrassingly parallel; LU spends more than 25% of its execution time in communication on 32 processors. The difference in communication costs between the Origin and Cluster is also interesting. Despite it's much greater raw communication performance, the Origin spends more total time in communication on SP than does NOW. We have explored protocol design trade-offs within the MPI layer on NOW and demonstrated that these can yield over a 100% performance improvement in communication performance on certain NPB applications.

4 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 at each machine size of interest. We then ran the trace for one processor through a cache simulator for a range of cache sizes.

The top curve of Figure 3 shows the data cache miss rate (with fully associative caches to capture the true working set) on one of four processors as a function of cache size for sizes between 1 KB and 4 MB. For LU, we see the smooth decrease in miss rate (following the general rule that doubling the cache eliminates one third of the misses) out to 32 KB. 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 [10] to shared address space programs and measured for



(a) **(b)** **(c)**

Figure 4 Message Scaling. This figure shows how the number of messages sent normalized to the 4 processor case (left), number of bytes sent (center) and average size of a message (right) scale as a function of processors.

SPLASH-2 [13]. The key observation is that with CPS scaling, the working set curve is different for each machine size. With eight processors, the 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) expands from 2 MB to 4 MB. As algorithms are tuned to be more cache friendly, like LU [14], this phenomenon will be more pronounced.

The important point in examining cache effects is they can have significant results on the scalability of benchmarks under CPS scaling. While not a novel result, the increase in memory-system efficiency due to cache-effect is often overlooked or, in the case with the NPB, 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 (1-4MB per node) and more cache-friendly codes [14] cache effects can play a significant role in the scalability of a machine under CPS rules. Indeed, in the case of the NPB the cache "boost" can mask poor performance in other areas, such as communication.

5 Communication Performance

Figure 4 shows the communication scaling of the NPB. Figure 4 (a) plots the change in total message count per processor as a function of the number of processors, normalized to the message count on 4 processors. The figure shows that the number of messages for all programs increases dramatically as the machine scales. Figure 4 (b) shows the byte count per processor as a function of scaling. The byte count per processor for all benchmarks decreases. Finally, Figure 4 (c) shows the resulting average size message per processor.

Within the realm of interest, there is an order of magnitude difference in the average size of a message. Interestingly, the smallest messages are still on the order of 1000 bytes, which is a substantially larger grain than found in many other parallel benchmarks [7, 13].

For LU, and SP, the amount of total communication follows surface to volume ratio as we scale the number of processors. For example, in SP, each processor owns $\sqrt{p}$ cubes of space in the physical domain. Since the number of messages is constant for each face of the cube, the number of messages per processor grows with $\sqrt{p}$, and thus the total number of messages increases as $O(p\sqrt{p})$. The total volume of communication grows with $O(\sqrt{p})$, and thus the volume per-processor decreases as $O(\sqrt{p})$. 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.

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. Combining the message characteristics with microbenchmark performance, we find that the NOW platform handles the load better than the Origin, i.e. the total time spent in com-

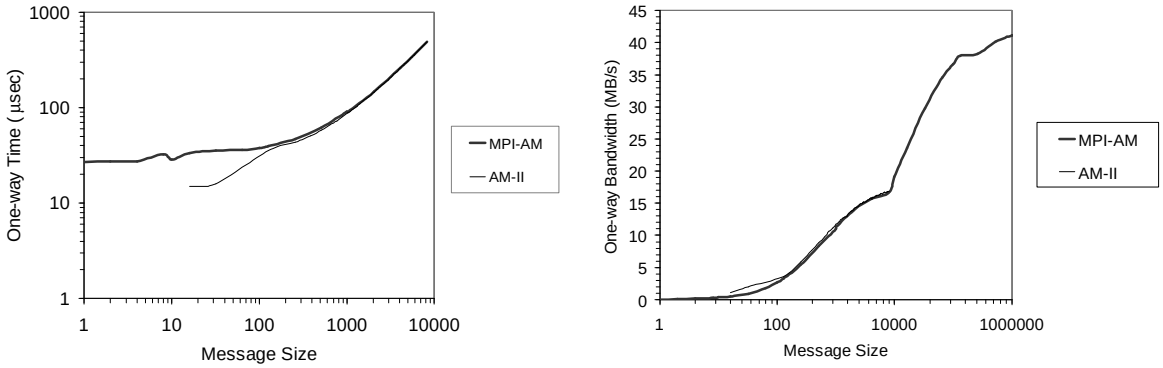
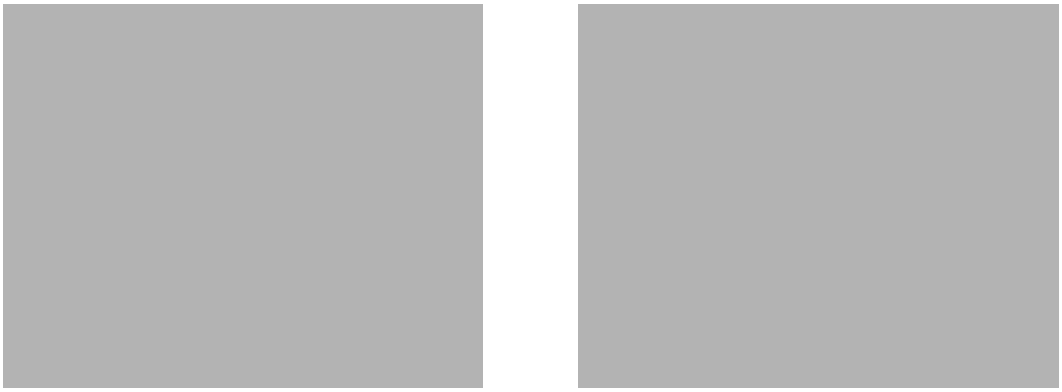


Figure 5 MPI Performance These figures show the performance of our MPI implementation in a dedicated environment. Left figure shows the one-way latency of small to medium messages using Dongarra's echo test [4]. Right figure shows the one-way bandwidth with message size up to 1 MB. Active Messages latency and bandwidth are shown for comparison.



(a) **(b)**

Figure 6 Communication Efficiency These figures shows the percentage of micro-benchmark peak bandwidth delivered by the MPI layer vs. the total time spent in sends and waiting for each application as a function of the number of processors on the cluster. Left figure shows the communication efficiency using the three-way handshake protocol in the MPI layer. Right figure shows the communication efficiency using the eager protocol in the MPI layer.

munication is closer to the predicted value. however, both show a significant drop from the microbenchmark performance.

The cause of this anomaly is presumably the implementation of the MPI layer and how it interacts with the underlying architecture. The evaluation of NPB drove the development of MPI layer on the NOW, because the total time predicted by combining the microbenchmark performance with the message profile data was significantly less than the time actually spent in communication (Figure 5 shows the microbenchmark performance of our MPI implementation using Active Messages, and figure 6 (a) shows the communication efficiencies of the NPB codes using our original MPI implementation). Further investigation revealed that the source of the problem was the internal protocol of the MPI layer. Our initial implementation of MPI used a conservative *three-way handshake* protocol. Since we were using low-latency Active Messages as a building block, using a handshake 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. We revised the MPI implementation to use a more aggressive *eager* protocol. This has significantly increased code complexity to resequence out-of-order messages and has slightly worse microbenchmark performance, but communication perfor-

mance in the context of real applications improved by as much as 100% in certain benchmarks. Figure 6 (b) shows the MPI communication efficiency of NPB2 using the eager protocol. On the Origin we are using vendor supplied MPI libraries and so are unable to find the source of the Origin performance anomaly.

6 Conclusion

A detailed analysis of the architectural requirements of the NPB 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 NOW.

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 NPB: good speedup implies good communication and conversely poor speedup implies poor communication. However, an important lesson from examining communication performance of the two platforms is that the scalability results of the NPB 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 begin to 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.

Although constant problem size scaling certainly appeals to common sense (a faster machine solves the given problem more quickly), it may not represent how machines are used in practice. An alternate view (that faster machines are for solving larger problems), generally complicates the performance comparison. However, it can reduce the interactions caused by changes in per-processor load as the machine scales. Thus, we are currently exploring alternative scaling rules for the NPB. Our preliminary results use a memory-constrained scaling paradigm, which more clearly distinguishes the NOW and Origin architectures. These results will be presented in the final version of the paper.

Reference

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

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

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

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

[12] 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

[13] 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.

[14] 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.