Penvelope: A New Approach to Rapidly Predicting the Performance of Computationally Intensive Scientific Applications on Parallel Computer Architectures

Daniel M. Pressel
US Army Research Laboratory (ARL), Aberdeen Proving Ground, MD
dmpresse@arl.army.mil

David Cronk
University of Tennessee, Knoxville, TN
cronk@cs.utk.edu

Sameer Suresh Shende
University of Oregon, Eugene, OR
sameer@cs.uoregon.edu

Abstract

A common complaint when dealing with the performance of computationally intensive scientific applications on parallel computers is that programs exist to predict the performance of radar systems, missiles and artillery shells, drugs, etc., but no one knows how to predict the performance of these applications on a parallel computer. Actually, that is not quite true. A more accurate statement is that no one knows how to predict the performance of these applications on a parallel computer in a reasonable amount of time. PENVELOPE is an attempt to remedy this situation. It is an extension to Amdahl's Law/Gustafson's work on scaled speedup that takes into account the cost of interprocessor communication and operating system overhead, yet is simple enough that it was implemented as an Excel spreadsheet.

1. Introduction

A common complaint when dealing with the performance of computationally intensive scientific applications on parallel computers is that programs exist to predict the performance of radar systems, missiles and artillery shells, drugs, etc., but no one knows how to predict the performance of these applications on a parallel computer. Actually, that is not quite true. A more accurate statement is that no one knows how to predict the performance of these applications on a parallel computer in a reasonable amount of time. We are developing a fast model of the performance of these applications on modern parallel computer architectures. As the first step in the process, we have chosen to model pure MPI applications. In general our approach works best for programs using a single communicator. However, it should be possible to use PENVELOPE to model more complicated applications.

The model uses "Back-of-the-Envelope" methods and relies on either measured or predicted (e.g., using the ENVELOPE model developed jointly at the US Army Research Laboratory and the University of Tennessee, Knoxville[1]) run times for single processor runs. It also relies on measured numbers of calls and the amount of data transferred for each of the commonly used MPI-1 calls (e.g., send, receive, and the more commonly used collective operations). This of course means taking the parallel measurements for each number of processors one is interested in. However, the model assumes that these numbers will be system independent, so the measurements only need to be made once per application. In contrast, the serial run times will be needed for each of the systems to be modeled.

In addition to the application specific data, one also needs system specific data concerning peak internode bandwidth, peak interprocessor/internode bandwidth with only a single sender/receiver pair, and the minimum latency for passing a one byte message. All of these numbers of course assume one is using MPI. While this is a lot of information to collect, fortunately it only needs to be collected once per system. Many vendors and some supercomputer sites (e.g., Oak Ridge National Laboratory) routinely publish this information. Once collected, it can be applied to the simulation of the performance of any number of applications.

The application specific and system specific information is then entered into a series of Microsoft Excel spreadsheets that our team has put together (one spreadsheet per application-number of processors pairing). While it can take some time to initially collect this data, the time required for the spreadsheet to perform

Proceedings of the Users Group Conference (DOD_UGC'04)
0-7695-2259-9/04 $ 20.00 **IEEE**

its calculations is negligible on today's PCs. This paper will discuss our experimental results based on the NAS benchmarks[2].

This project was supported by a grant of computer time from the DoD High Performance Computing Modernization Program.

2. The Problem

How does one predict the performance of a parallelized scientific application on a computer? In general, one is not interested in just one computer. Rather, there are likely to be multiple competing systems and the user is trying to decide which system(s) to request access to. Frequently, the system staff is attempting to evaluate competing bids from multiple vendors, possibly for systems that will not be generally available for several more months. Complicating matters further, one may have multiple applications, multiple data sets per application, and almost certainly will want the applications run for a varying numbers of processors.

3. The Traditional Solutions

The most common answer to this problem is to measure the performance. This of course takes time and in many cases requires considerable resources. Additionally, when talking about one of a kind systems and/or systems that are still being developed, this is not even a possibility. The most common solutions to this problem have been:

- Wait until the systems are available and then run your benchmarks.
- Rely on industry standard benchmarks such as Linpack[3], STREAMS[4], or the NPB benchmark suite out of NASA Ames Research Laboratory.
- Extrapolate from runs made on the previous generation of hardware from the same vendor and hope for the best.

As was demonstrated in Reference 5, there can be a considerable degree of variability in the delivered levels of performance for each of these benchmarks. So which if any of them should one use? Obviously what is needed is an entirely different approach.

4. Back-of-the-Envelope Calculations

A promising concept for an alternative approach is to use Back-of-the-Envelope calculations. Engineers and scientists have been using this approach for decades if not centuries to predict the outcome of their work prior to committing themselves to a particular design or costly experiment. The key concept is to remove enough of the fluff that the equations can be readily solved while not eliminating any of the details that really matter. Amdahl's law and the work of Gustafson on scaled speedup are prime examples of this approach at work. While in many cases these approaches are good enough, they will fail in two key areas:

- Neither will tell you if the program has been poorly parallelized (e.g., the granularity is too fine).
- Both lack any insights as to the importance of the system interconnect. For all they care, a 1 Byte/second interconnect is just as good as a 1 GB/second interconnect. Similarly, an interconnect with a 1 minute message passing latency is of equal value to one with a 1 microsecond message passing latency.

What is needed is an extension that incorporates both Amdahl's law and the work of Gustafson while making a limited effort to take into account the design of the program and the systems it will run on.

In the general case, this problem may be too complicated to solve using this approach. However, it was felt that if one limited the problem to a commonly occurring case or set of cases, then it might be possible to achieve usable results with the desired degree of effort. The obvious choice was programs parallelized using the more commonly used MPI-1 features with a single communicator. With some effort on the part of the user, it should be possible to extend the model to cover programs with multiple communicators. At this point in the development of PENVELOPE, no attempt has been made to investigate this possibility.

5. General Approach

Based on data collected using TAU[6,7,8], or other similar utilities, the cost for the data communications is estimated. This means that on at least one system an instrumented run must be made for each number of processors being used. As previously mentioned, one will also need information concerning the bandwidth and latency for the system interconnects. Provisions have been made for specifying the cost of the I/O, operating system overhead, and whether or not communications are overlapped with computations. The model assumes that collective communications are never overlapped with computations. Sends and Receives may or may not be overlapped. Currently this is a simple yes or no question, with no provisions for partial overlapping of communications with computations. The cost of the computations is estimated using Amdahl's law for fixed

problem sizes, or Gustafson's work for problems involving scaled speedup.

6. Limitations

There are four main limitations with the model at the present time. The first one is difficult to know how to handle. When discussing an application's performance, one would like to think in terms of there being a single value for the run time. In reality, there will always be a range of values. If this range is small enough (e.g., 1–10%), then it probably does not matter. However, in a recent effort to benchmark some of the systems at the ARL-MSRC, the following degrees of variability were observed.

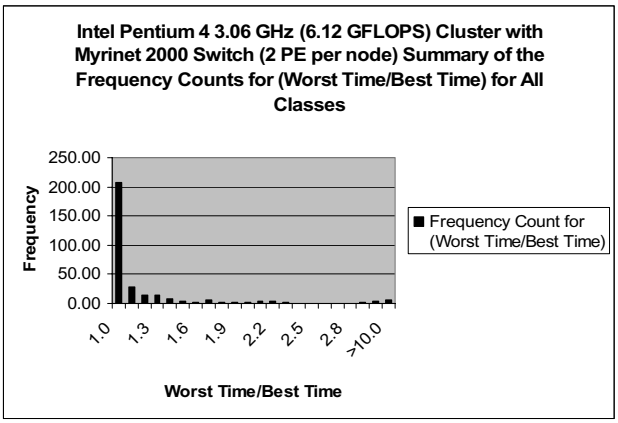

Figure 1. The variability in run times on the Intel Pentium 4 cluster at the ARL-MSRC (0.1 increment binning)

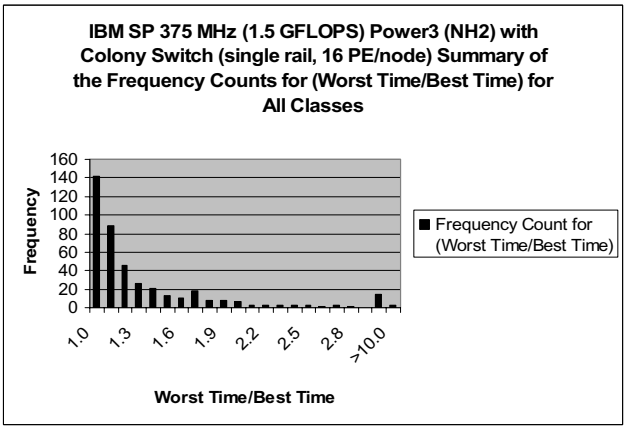

Figure 2. The variability in run times on the IBM SP Power3 (NH2) at the ARL-MSRC (0.1 increment binning)

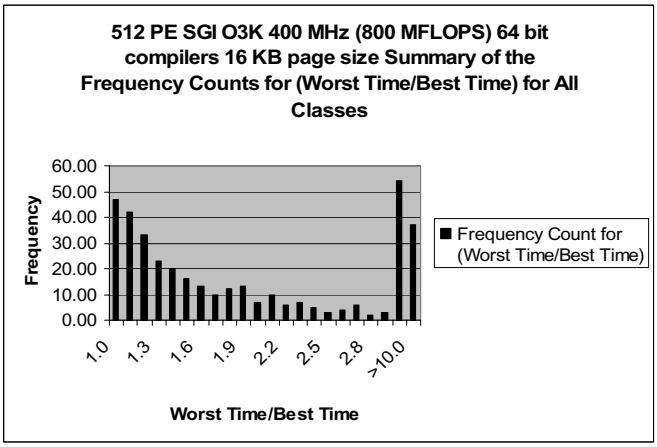

Figure 3. The variability in run times on the SGI Origin 3000 at the ARL-MSRC (0.1 increment binning)

The second problem is that many systems have insufficient memory bandwidth in a node to peg the interface on all of the processors at the same time. Therefore, when going from a partially filled node to a fully utilized node, there may be a significant decrease in the per processor level of performance. The third problem is superlinear speedup, where as the processor count increases the amount of work per processor decreases to the point that the working set fits in cache. A side effect of this is that once the program enters the region of superlinear speedup, the demands on the memory system can drop markedly. In many cases, this will eliminate the limitations discussed in the second problem.

The fourth problem is related to the second and third problems. What value should one use for the serial runtime? For large problems, it may not even be possible to run the application on a single processor. Assuming that it is possible to run the application on a single processor, is it best to use the measured serial run time, the measured run time for a small number of processors* that number of processors (this might eliminate some of the errors associated with problems 2 and 3), or use a model such as ENVELOPE to estimate the serial performance based on more appropriate assumptions for per processor memory bandwidth and cache hit rates when using N processors. Currently for 4-9 processors, the measured serial run time was used in our experiments whenever it was available. For larger numbers of processors, 4* the four processor run time was used consistently. In some cases, a better choice would have been to use the minimum of these two values, while the best solution is likely to be modeling the serial performance.

7. Results

In an attempt to jump start the process, data from[9, 10, and 11] were used to calculate the communication costs. Hardware information came from vendor websites, vendor presentations at Supercomputer 2003 and other conferences, from numerous publications out of Oak Ridge National Laboratory and the Ohio Supercomputer Center/Ohio State University. At the present time, we are assuming that the BT, CG, EP, FT, and LU benchmarks overlap computation with communication, while the MG and SP benchmarks do not overlap computation with communication. It now appears as though on some systems the MG benchmark may be able to overlap computation with communication (probably due to the buffering of messages). Benchmark data for the class W, A, and B benchmarks were collected for four systems at the US Army Research Laboratory–Major Shared Resource Center (ARL MSRC). This data was supplemented with results published by the NAS group at NASA Ames Research Laboratory and roughly 40 other sites on the web, along with correspondence with some of the vendors and two other supercomputing sites.

Based on this data, the run time was estimated for a large number of combinations of system, number of processors (in some cases out to 64 processors), and benchmarks for the class A data sets. A smaller number of combinations were estimated for the class B and W data sets due to the more limited amount of information available for modeling these data sets. When sufficient information existed, the predicted run times were compared to the measured runtimes (using the best times when more than one measurement was made). Similarly, the predicted run times using Amdahl's law was compared to the measured run times. Figures 4 and 5 show these results. In many cases there was little difference between the two sets of predictions due to the overlapping of communication with computation. Where there was a strong degree of superlinear speedup, neither model worked well, but PENVELOPE appears to be worse. In cases where the communication costs were large, PENVELOPE is significantly more accurate than using Amdahl's law by itself. In a number of cases where neither cache effects nor communications costs were large, Amdahl's law tended to under estimate the run time by 10–30 % while PENVELOPE would over estimate the run time by a similar amount.

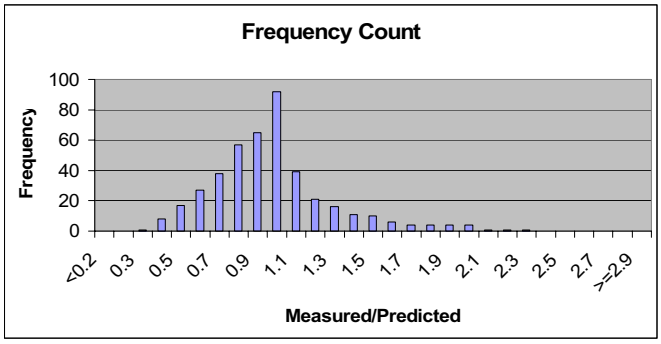

Figure 4. The accuracy of PENVELOPE in estimating the run time for selected runs of the NAS benchmark suite (0.1 increment binning)

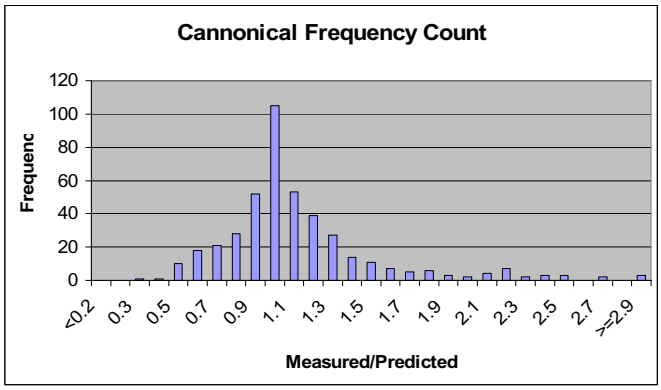

Figure 5. The accuracy of Amdahl's law in estimating the run time for selected runs of the NAS benchmark suite (0.1 increment binning)

8. Conclusions

PENVELOPE is a work in progress. As such it shows significant promise in achieving its goals. However, at the present time, some of those goals are only partially achieved. It is hoped that continued work will rapidly improve the quality of the predictions.

References

1. Pressel, Daniel M., "ENVELOPE: A New Approach to Estimating the Delivered Performance of High Performance Processors." *ARL-TR-2671*, US Army Research Laboratory, February 2002.

2. The NAS Benchmark (NPB) home page can be found at http://www.nas.nasa.gov.

3. Dongara, J., "Linpack Benchmark-Parallel" table for the Linpack Benchmark." published electronically at http://www.netlib.org.

4. McCalpin, J., "Equivalent MFLOPS" table for the STREAM Benchmark." published electronically at http://www.cs.virginia.edu/stream.

5. Pressel, Daniel M. and Jelani Clay, “Benchmarking the Benchmarks.” *ARL-TR-2805*, US Army Research Laboratory, September 2002.

6. Shende, S., et al., “Portable Profiling and Tracing for Parallel Scientific Applications Using C++.” *Proceedings of ACM SIGMETRICS Symposium on Parallel and Distributed Tools (SPDT ’98)*, August 1998, pp. 134–135.

7. “TAU Portable Profiling.” University of Oregon, Published electronically at http://www.cs.uoregon.edu/research/paracomp/tau.

8. Maloney, Allen D. and Sameer Shende, “Performance Technology for Complex Parallel and Distributed Systems.” “*Quality of Parallel and Distributed Programs and Systems,*” (Eds. Peter Kacsuk and Gabriele Kotsis), Nova Science Publishers, Inc., NY, 2003, pp. 25–41.

9. Wong, Frederick C., et al., “Architectural Requirements and Scalability of the NAS Parallel Benchmarks.” published in the conference proceedings for SC’99, November 1999.

10. Lobosco, Marcelo, Vitor Santos Costa, and Claudio L. de Amorim, “Performance Evaluation of Fast Ethernet, Giganet and Myrinet on a Cluster”.

11. Tabe, Theodore B. and Quentin F. Stout, “The Use of the MPI Communication Library in the NAS Parallel Benchmarks.” *CSE-TR-386-99*, University of Michigan as a Computer Science and Engineering Technical Report, 1999.