Performance Sensitivity Studies for Strategic Applications

Laura C. Carrington, Xiaofeng Gao, Nicole Wolter, & Allan Snavely

Performance Modeling and Characterization Lab
San Diego Supercomputer Center
University of California
San Diego, CA 92093
{lcarring,xgao,wolter,allans}@sdsc.edu

Roy L. Campbell, Jr.

High Performance Computing Division
Army Research Laboratory
Major Shared Resource Center
Aberdeen Proving Ground, MD 21005
rcampbell@arl.army.mil

Abstract

This paper applies modeling and simulation to key HPCMP systems and applications to determine the degree to which fundamental system attributes affect application performance. Synthetic probes are used to ascertain target system capabilities, while application tracing is used to uncover the memory and communication usage characteristics of target codes. A predictive model subsequently melds system and application data in order to project a time-to-solution for each application and system pair. System attributes are then systematically modified, and the predictive model is again applied, to determine the sensitivity of application performance to key system attributes.

Time-to-solution predictions for five application test cases (AVUS Standard/Large, HYCOM Standard, OVERFLOW2 Standard, and RFCTH2 Standard) from the HPCMP TI-05 Benchmarking Suite were validated against Government-obtained benchmarking data for 10 HPCMP systems (ranging from an SGI Origin 3800 to an IBM Opteron cluster), yielding an average absolute error of 18%. Sensitivity analysis was then applied to AVUS Large and OVERFLOW2 Standard using the DoD baseline system (a 2832 processor IBM p655) as the target system, revealing that both test cases have difficulty staying within mid-tier (L2) and outer-tier cache (L3), and therefore greatly benefit from increased L3 and main memory bandwidths.

1. Introduction.

The performance of a parallel application on an HPC system is a function of the algorithm(s) used with the code, the programming style and prowess of the code author(s), the selected compiler(s) and libraries, and a host of system attributes pertaining to the CPU, OS, I/O, interconnect, and memory. Therefore, one might conclude that performance models for scientific applications on such complex systems must account for all elements of the target system and application; however, this work shows that a predictive framework, which considers only the most vital factors, on average can predict an application's performance within 20%.

This framework uses simple tools to generate reasonably accurate performance predictions within a relatively short period of time for a large number of systems, given a set of codes (representing a target workload) and a set of probe results (which are basic attributes measured by synthetic tests) for systems that are not readily available for applications benchmarking. In previous work [25-27], this framework was described, validated, and used to accurately model the performance of small parallel scientific kernels and applications on different HPC architectures. Only two probes were used: MAPS [29] (which measures the memory bandwidth versus message size for a single processor) and maps_ping [30] (which measures interconnect bandwidth and latency via a simple ping-pong test between two nodes). Predictions were blindly conducted without any knowledge of observed runtimes, using only probe results and basic system data to represent system attributes. Performance insights were further extended by employing sensitivity analysis though the variation of key system parameters.

2. A Performance Modeling Framework.

Isolation (divide and conquer) and simplicity (Occam's Razor) were incorporated into the framework design in order to achieve a rapid, yet accurate, performance model that accounts for complexities in the memory hierarchy and interconnect, while still being applicable to arbitrary applications and systems. Isolation facilitates a dynamic framework which can be increased and decreased in complexity as necessary by simply adding or deleting "terms" within the model. Simplicity ensures that only those "terms" that are absolutely necessary for a reasonable degree of accuracy

are retained. A detailed description of the framework can be found in Snavely et al [26].

Based on the hypothesis that a parallel application's performance is largely defined by its single processor performance and interconnect usage, the framework characterizes the code from these two vantage points, while taking into consideration only the most paramount features of the target system. Starting simply and adding complexity only when necessary to account for observed performance, the framework consists of a single processor model combined with a communication model (Figure 1).

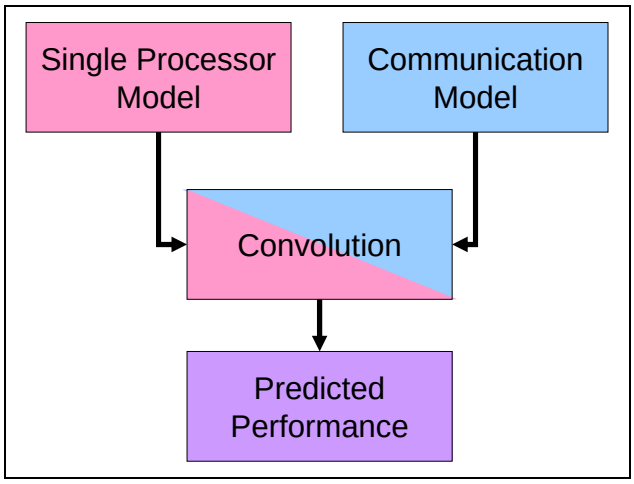

Figure 1. Performance prediction framework for parallel applications.

Clearly, other factors can affect performance, but often single processor and interconnect performance are sufficient for accurate performance prediction (~20% error), while adding more factors increases the complexity of the model with only nominal gains (~1-2%) in accuracy [26].

The single processor and interconnect models both use application signatures and system profiles, which are combined though a convolution method to yield a predicted time-to-solution. An application signature summarizes the operations that will be conducted by the target application and includes memory and communication access patterns, all attributes being irrespective of any particular system. For the single processor model, the application signature is a memory trace collected via the MetaSim Tracer, while for the communication model, the application signature is an MPI trace collected by MPIDtrace (Dimemas). A system profile contains rates at which a target system can perform basic operations, including message passing, memory loads and stores, and floating-point operations, independent of any particular application. Such rates are determined by low-level probes (synthetic benchmarks) to minimize the impact of data gathering and maximize the isolation between the target application and the system profile.

To achieve a performance prediction for a target application on a target system, the operations described in the application signature are simulated according to the rate and structure information found in the system profile via a careful convolution methodology. Convolution is automated by the MetaSim Convolver for the single processor model and Dimemas for the communications model.

The predictions presented in this paper are a byproduct of a recently expanded framework that includes two new model terms:

1. **Data Dependency** – requiring the loops within a target application to be characterized according to the presence of any data dependencies as well as probes to measure the ability of a target system to execute loops containing a notable degree of data dependency
2. **Control Dependency** – requiring loops within a target application to be characterized according to branch intensity as well as probes to measure the ability of a target system to perform loops containing a high level of branch intensity.

While investigating the difference in predicted and observed performance in a previous work, it was discovered that some loops with the same size working set (cacheability) and same memory access pattern had noticeably different performance in terms of instruction execution rate. This disparity was found to be primarily attributable to differences in the data and control dependencies in these seemingly similar loops.

As for data dependency, some loops contain a large number of definition-use dependencies, thereby requiring that an instruction wait for a value to be produced by a previous instruction before the current instruction can be executed. In the meantime, no other independent instructions are available for execution, rendering a notable portion of the pipeline idle and in turn reducing processor efficiency. Other loops on the other hand may have a large number of independent instructions available at any given moment, thereby more efficiently utilizing the ILP (instruction level parallelism) of the target system. To distinguish between these cases, the MetaSim Binary Analyzer was developed to uncover register-carried dependencies via static analysis and to compute, for each floating-point instruction in a loop, the minimum distance to the instructions which generate the current floating-point instruction's data sources. Each loop is subsequently scored by the weighted average of these floating-point definition-use distances. (The weighting comes from dynamic path information acquired by tracing and by giving more weight to more frequently executed instructions.) Probes are then executed to measure the performance of target systems on a range of loops with different **Average Definition-Use Distance** scores. In a final step, the probe data and

the scored MetaSim Binary Analyzer data are combined through an added "term" in the framework's convolution process, thereby incorporating the effects of data dependency into the model.

A very similar approach discovers, via static analysis, the ratio of non-branch to branch instructions and quantifies via probes the effect of different ratios on loop execution for a target system. An extra "term" is again added to the convolution step to meld the application and system data to account for branch dependencies.

Predicted and observed times-to-solution are compared in the Section 4 as a validation of the current model's predictive abilities.

3. DoD HPCMP TI-05 Application Test Cases.

Five application test cases from the DoD HPCMP TI-05 Benchmarking Suite were analyzed. Each test case was executed at three different processor counts, ranging from 16 to 384 processors.

AVUS STANDARD & LARGE

AVUS was developed by the Air Force Research Laboratory (AFRL) to determine the fluid flow and turbulence of projectiles and air vehicles. Its standard test case calculates 100 time-steps of fluid flow and turbulence for a wing, flap, and end plates using 7 million cells. Its large test case calculates 150 time-steps of fluid flow and turbulence for an unmanned aerial vehicle using 24 million cells.

HYCOM STANDARD

The Naval Research Laboratory (NRL), Los Alamos National Laboratory (LANL), and the University of Miami developed HYCOM as an upgrade to MICOM (both well-known ocean modeling codes) by enhancing the vertical layer definitions within the model to better capture the underlying science. HYCOM's standard test case models all of the world's oceans as one global body of water at a resolution of one-fourth of a degree when measured at the Equator.

OVERFLOW2 STANDARD

OVERFLOW-2 was developed by NASA Langley and NASA Ames to solve CFD equations on a set of overlapping, adaptive grids, such that the grid resolution near an obstacle is higher than that of other portions of the domain. This approach allows computation of both laminar and turbulent fluid flows over geometrically complex, non-stationary boundaries. The standard test case of OVERFLOW-2 models fluid flowing over five spheres of equal radius and calculates 600 time-steps using 30 million grid points.

RFCTH STANDARD

Sandia National Laboratories (SNL) developed CTH to model complex multidimensional, multiple-material scenarios involving large deformations or strong shock physics. RFCTH is a non-export-controlled version of CTH. The standard test case of RFCTH models a ten-material rod impacting an eight-material plate at an oblique angle, using adaptive mesh refinement with five levels of enhancement.

4. Performance Prediction Results.

Table 1 reveals the average absolute difference between predicted and observed performance for five application test cases and nine production architectures.

Table 1. Average absolute difference and standard deviation for performance predictions as compared to observed performance.

Category	% Difference	
	Absolute AVG	SD
Overall	18%	20%
AVUS_STD	13%	16%
AVUS_LG	21%	21%
HYCOM_STD	17%	20%
OVERFLOW2_STD	13%	19%
RFCTH2_STD	28%	24%
SGI_O3800_400MHz_NUMA	22%	27%
IBM_P3_375MHz_COLONY	13%	16%
HP_SC45_1GHz_QUADRICS	17%	16%
IBM_690_1.3GHz_COLONY	17%	28%
IBM_690_1.7GHz_FEDERATION	22%	8%
LNX_Xeon_3.06GHz_MYRINET	21%	21%
SGI_Altix_1.5GHz_NUMA	26%	28%
IBM_655_1.7GHz_FEDERATION	6%	10%
IBM_Opteron_2.2GHz_MYRINET	23%	28%

For the majority of the application test cases, the framework accuracy ranges from 13% to 21%, with RFCTH2 Standard being the only outlier (28%) due to a few large differences at 64 processors. (It is suspected that an error may exist in the RFCTH2 Standard traces for this processor count.) For the majority of the system architectures, accuracy ranges from 13% to 23%, with the IBM p655 (6%) and the SGI Altix (26%) being the only outliers.

Although absolute difference is reported to avoid sign canceling, it is worthwhile to also examine signed difference (Table 2) to reveal any systematic behavior.

Table 2. Average signed difference for performance predictions as compared to observed performance.

Category	AVG % Difference
Overall	-3%
AVUS_STD	4%
AVUS_LG	-1%
HYCOM_STD	-12%
OVERFLOW2_STD	7%
RFCTH2_STD	1%
SGI_O3800_400MHz_NUMA	3%
IBM_P3_375MHz_COLONY	5%
HP_SC45_1GHz_QUADRICS	-12%
IBM_690_1.3GHz_COLONY	-14%
IBM_690_1.7GHz_FEDERATION	3%
LNX_Xeon_3.06GHz_MYRINET	18%
SGI_Altix_1.5GHz_NUMA	-3%
IBM_655_1.7GHz_FEDERATION	3%
IBM_Opteron_2.2GHz_MYRINET	-8%

Only negligible systematic error exists overall, meaning that a prediction is equally likely to be too fast or too slow. The HP SC45 and the IBM p690 (with a Colony interconnect), however, stand out as being consistently under-predicted (i.e., the predicted time is often faster than the observed time). The I/O subsystems for these systems are relatively weak compared to other architectures, and I/O is not considered in the current model, possibly explaining the bias. Future work will investigate the addition of I/O modeling to the framework. The Xeon stands out as being systematically over-predicted (i.e., the predicted time is often slower than the observed time). This anomaly is still under investigation, although it is thought that the probes to characterize the system may not have been executed under optimal conditions.

Nevertheless, even when considering outliers, the results strongly support the notion that about 80% of observed performance (for arbitrary combinations of systems and applications) can be predicted by a simple model.

5. Performance Sensitivity Studies.

Having developed confidence in the predictive model from the data provided in the previous section, the framework is now used to predict the performance improvement for a target system given a specific set of hardware modifications.

Using the application signatures for AVUS Large and OVERFLOW2 Standard and the system profile for the DoD baseline system (a 2832 processor IBM p655), sensitivity was analyzed with respect to 10 different modified hardware scenarios (Table 3).

Table 3. Modified hardware scenarios.

Scenario	Description
Case 1	Reduced interconnect latency by 2
Case 2	Increased interconnect bandwidth (BW) by 2
Case 3	Increased FLOP rate by 2
Case 4	Increased L1 BW by 2
Case 5	Increased L1 and L2 BWs by 2
Case 6	Increased L1, L2, and L3 BWs by 2
Case 7a	Increased L1, L2, L3, and main memory (MM) BWs by 2
Case 7b	Increased L1, L2, L3, MM, and on-node BWs by 2
Case 8a	Increased MM BW by 2
Case 8b	Increased MM and on-node BW by 2

Sensitivity results for AVUS Large and OVERFLOW Standard are plotted in Figures 2 and 5, respectively, in relative performance units (i.e., new performance divided by baseline performance). Since the data is not easily discerned in one plot, concentric rings are extracted and enlarged in Figures 3-4 for AVUS Large and Figures 6-8 for OVERFLOW2 Standard.

Assuming for the sake of discussion that a notable relative increase in performance from one case to another is at least 5%, the plots indicate that for both codes the interconnect latency and BW, FLOP rate, and L1 BW have about the same effect on performance (per code). The L1 and L2 BWs combined have a slightly bigger impact, which is still not "notable" relative to the first four cases. L1, L2, and L3 BWs combined have a "notable" impact relative to the previous five cases. L1, L2, L3, and MM BWs combined provide the largest impact of all eight cases, and MM BW alone provides the second largest impact of all eight cases. Therefore, both codes are having difficulty staying within the mid-tier (L2) and outer-tier (L3) cache, as both greatly benefit from L3 and MM BW increases.

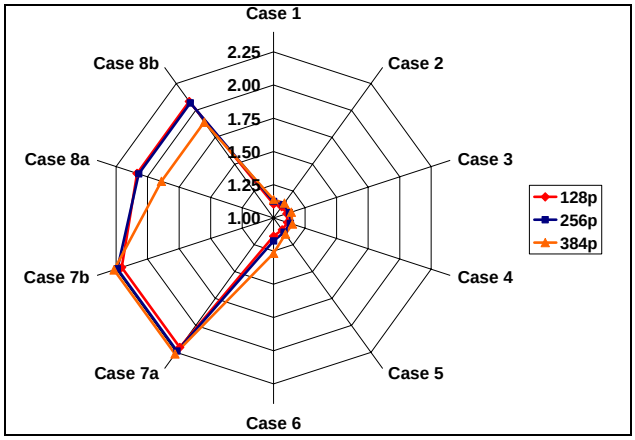

Figure 2. Sensitivity results for AVUS Large.

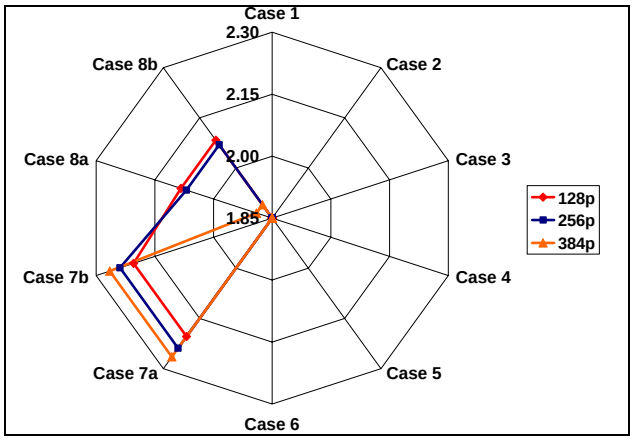

Figure 3. Sensitivity results for AVUS Large (1.85-2.3).

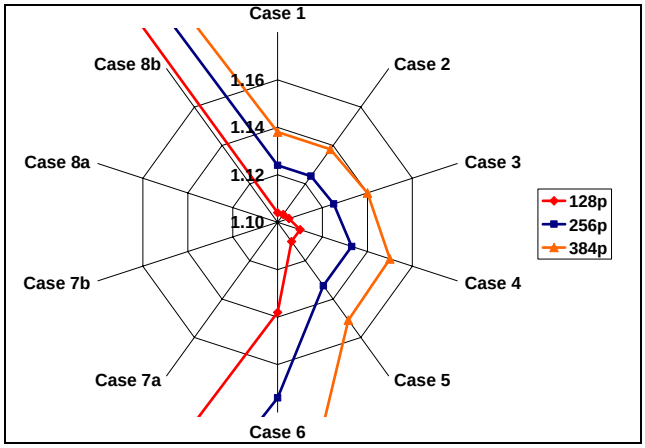

Figure 4. Sensitivity results for AVUS Large (1.1-1.16).

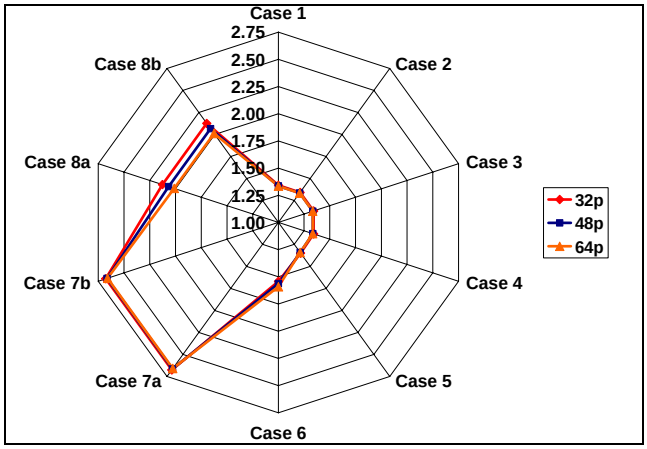

Figure 5. Sensitivity results for OVERFLOW2 Standard.

As the processor count increases, AVUS Large benefits from hardware modifications increasingly for Cases 1-7 and decreasingly for Case 8 (MM BW). OVERFLOW Standard on the other hand decreasingly benefits from hardware modifications for all cases as the processor count increases. Table 4 summarizes the maximum effect of the tested processor count range on performance. Only Cases 6 and 8 (L1/L2/L3 BW and MM BW) for AVUS Large and Case 8 (MM BW) for OVERFLOW2 Standard are profoundly affected by processor count.

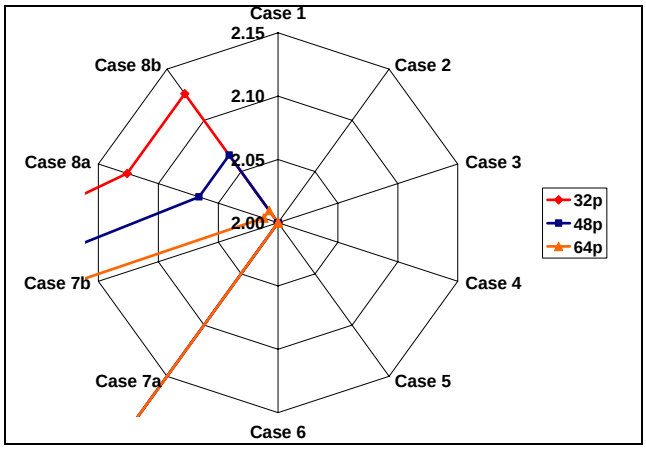

Figure 6. Sensitivity results for OVERFLOW2 Standard (2-2.15).

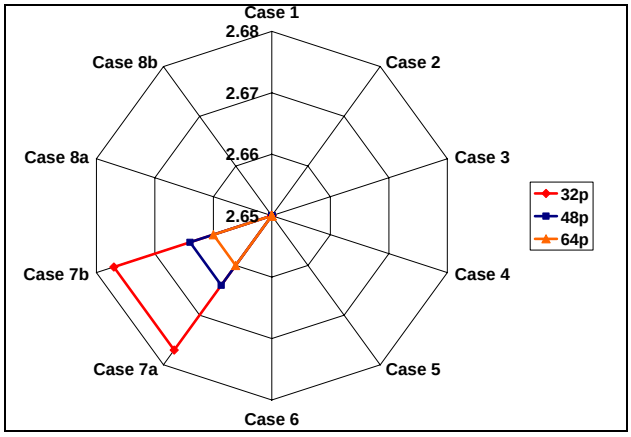

Figure 7. Sensitivity results for OVERFLOW2 Standard (2.65-2.68).

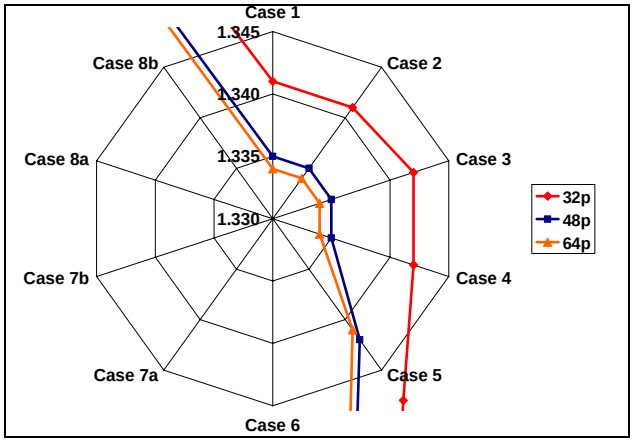

Figure 8. Sensitivity results for OVERFLOW2 Standard (1.33-1.345).

Figure 9 contains a comparison of the average sensitivity results for AVUS Large and OVERLFLOW2 Standard. In general, OVERFLOW Standard is more sensitive than AVUS Large to all hardware modification scenarios except for Case 8 (MM BW), in which case the response of both test cases is similar.

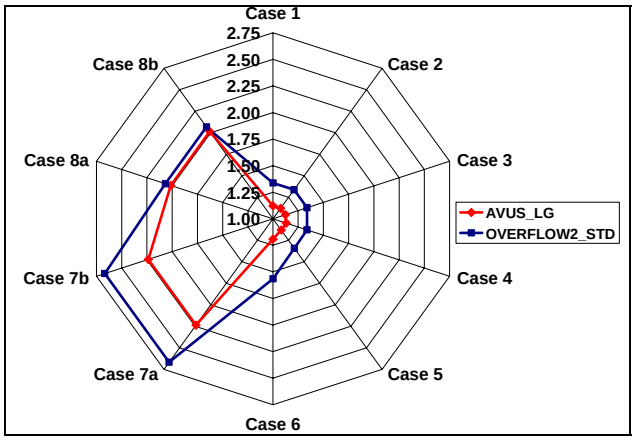

Figure 9. Average sensitivity results for AVUS Large and OVERFLOW2 Standard.

Table 4. Maximum effect of processor count for tested range.

Test Case	Case	% Difference in Performance
AVUS Large	1	3.1%
	2	3.1%
	3	3.2%
	4	3.6%
	5	3.7%
	6	11.2%
	7a	2.8%
	7b	2.8%
	8a	10.2%
	8b	10.2%
OVERFLOW2 Standard	1	0.5%
	2	0.5%
	3	0.6%
	4	0.6%
	5	0.5%
	6	3.2%
	7a	0.6%
	7b	0.6%
	8a	5.7%
	8b	5.7%

6. Conclusions.

The PMaC framework presented here has been refined over time to deliver fast performance prediction with a reasonable degree of accuracy (~20%). Through a series of sensitivity studies, this framework revealed that AVUS Large and OVERFLOW2 Standard benefit greatly from bandwidth increases in outer-tier cache (L3) and main memory, thereby providing a focal point for code enhancements as well as guidance for hardware purchases.

Acknowledgements

This work was sponsored in part by the Department of Defense High Performance Computing Modernization Program Office through a contract award entitled “HPC Benchmarking” and the Department of Energy Office of Science through a SciDAC contract award entitled “High-End Computer System Performance: Science and Engineering”. The application benchmarking data used in this study was obtained by the following cast of many (listed alphabetically): Mr. Wendell Anderson (NRL), Mr. Robert W. Alter (ERDC-CSC), Dr. Paul M. Bennett (ERDC-CSC), Dr. Sam B. Cable (ERDC-CSC), Dr. John E. Cazes (TACC), Ms. Christine Cuicchi (NAVO), Ms. Carrie L. Leach (ERDC-CSC), Mr. Mitch Murphy (MHPCC), Dr. Thomas C. Oppe (ERDC-CSC), Mr. Daniel M. Pressel (ARL), Mr. Daniel S. Schornak (ASC-CSC), and Dr. William A. Ward, Jr. (ERDC-CSC). Application traces were gathered by (besides the authors) Mr. Mike Timmerman (Instrumental) and Ms. Cynthia Bailey Lee. The European Center for Parallelism of Barcelona (CEPBA) is acknowledged for continued support of their profiling and simulation tools.

A generous grant of computer time by the DoD High Performance Computing Modernization Program at the ARL, ASC, ERDC, and NAVO Major Shared Resource Centers, the MHPCC Allocated Distributed Center, and the NRL Dedicated Distributed Center was vital to the successful completion of this work. Additional computer time was graciously provided by the Pittsburgh Supercomputer Center via an NRAC award and the San Diego Supercomputer Center.

References

1. Svobodova, L. Computer System Performance Measurement and Evaluation Methods: Analysis and Applications. Elsevier, N.Y., 1976.
2. Ballansc, R.S., Cocke, J.A., and Kolsky, H.G. “The Lookahead Unit”, Planning a Computer System, McGraw-Hill, New York, 1962.
3. Boland, L.T., Granito, G.D., Marcotte, A.V., Messina, B.V. and Smith, J.W. “The IBM system 360/Model9:Storage System”, *IBM J. Res. And Develop.*, 11, pp.54-79 (1967).
4. Tjaden, G.S. and Flynn, M.J. “Detection and Parallel Execution of Independent Instructions”, *IEEE Trans. Comptrs.*, C-19, pp889-895 (1970).
5. Murphey, J.O. and Wade, R.M. “The IBM 360/195”, *Datamation*, 16:4, pp. 72-79 (1970).
6. Burger, D., Austin, T.M., and Bennett, S. “Evaluating future microprocessors: The simplescalar tool set” *Tech. Rep. CS-*

TR-1996-1308, University of Wisconsin-Madison, 1996.
7. Lo, J., Egger, S., Emer, J., Levy, H., Stamm, R., and Tullsen, D., "Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading", *ACM Transactions on Computer Systems*, August 1997.
8. Falsafi, B. and Wood, D.A "Modeling Cost/Performance of a Parallel Computer Simulator", *ACM Transactions on Modeling and Computer Simulation*, 7:1, pp. 104-130 (1997).
9. Gibson, J., Kunz, R., Ofelt, D., Horowitz, M., Hennessy, J., and Heinrich, M. "FLASH vs. (Simulated) FLASH: Closing the Simulation Loop", *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 49-58, November 2000.
10. Saavedra, R.H., Smith, A.J., "Measuring Cache and TLB Performance and Their Effect on Benchmark Run Times", *IEEE Transactions on Computers* 44(10) pp1223-1235 1995
11. Saavedra, R.H., Smith, A.J. "Analysis of Benchmark Characteristics and Benchmark Performance Prediction", *TOCS14*(4) pp344-384,1996
12. Saavedra, R.H., Smith, A.J. "Performance Characterization of Optimizing Compilers", *TSE21*(7) pp615-628,1995
13. Mendes, C.L., Reed, D.A. "Integrated Compilation and Scalability Analysis for Parallel Systems", *IEEE PACT* 1998
14. Mendes, C.L., Reed, D.A. "Performance Stability and Prediction", *IEEE / USP International Workshop on High Performance Computing*, 1994
15. Simon, J., Wierun, J. "Accurate Performance Prediction for Massively Parallel Systems and its Applications", *Euro-Par*, Vol II pp675-688, 1996
16. Crovella, M.E., LeBlanc, T.J. "Parallel Performance Prediction Using Lost Cycles Analysis", S*C 1994* pp600-609
17. Xu, Z., Zhang, X., Sun, L. "Semi-empirical Multiprocessor Performance Predictions", *JPDC* 39, pp 14-28, 1996
18. Abandah, G., Davidson, E.S. "Modeling the Communication Performance of the IBM SP2", *Proceedings Int'l Parallel Processing Symposium*, pp. 249-257, April 1996
19. Boyd, E.L., Azeem, W., Lee, H.H., Shih, T.P., Hung, S.H., and Davidson, E.S. "A Hierarchical Approach to Modeling and Improving the Performance of Scientific Applications on the KSR1", *Proceedings of the 1994 International Conference on Parallel Processing*, Vol. III, pp. 188-192, August 1994
20. Hosie, A., Olaf, L., Wasserman, H. "Performance Analysis of Wavefront Algorithms on Very-Large Scale Distributed Systems", *Springer's "Lecture Notes in Control and Information Sciences"*, 249, p. 171 (1999).
21. Hosie, A., Olaf, L., Wasserman, H. "Scalability Analysis of Multidimensional Wavefront Algorithms on Large-Scale SMP Clusters", *Proceedings of Frontiers of Massively Parallel Computing '99*, Annapolis, MD, February 1999.
22. Kerbyson, D.J., Hoisie, A., and Wasserman, H.J., "Modeling the Performance of Large-Scale Systems", *Keynote paper, UK Performance Engineering Workshop (UKPEW03)*, July 2003, and in IEE Software, Inst. Electrical Engineers, August 2003.
23. Yong, L., Olaf, L.M., Wasserman, H. "Development and Validation of a Hierarchical Memory Model Incorporating CPU- and Memory-Operation Overlap", *Proceedings of the First International Workshop on Software and Performance*, Santa Fe, NM, pp. 152-163 (1996).
24. Spooner, A. and Kerbyson, D. "Identification of Performance Characteristics from Multi-view Trace Analysis", Proc. Of Int. Conf. On Computational Science (ICCS) part 3, 2659, pp. 936-945 (2003).
25. Carrington, L., Wolter, N., and Snavely, A. "A Framework for Application Performance Prediction to Enable Scalability Understanding", *Scaling to New Heights Workshop*, Pittsburgh, May 2002.
26. Snavely, A., Wolter, N., Carrington, L., Badia, R., Labarta, J., Purkasthaya, A. "A Framework to Enable Performance Modeling and Prediction", *Supercomputing 2002*.
27. Snavely, A., Wolter, N., and Carrington, L., "Modeling Application Performance by Convolving Machine Signatures with Application Profiles", *IEEE 4th Annual Workshop on Workload Characterization*, Austin, Dec. 2, 2001.
28. Davidson, E. and Kumar, B. "Computer System Design Using a Hierarchical Approach to Performance Evaluation", Communications of the ACM, 23:9, pp. 511-521 (1980).
29. http://www.sdsc.edu/PMaC/MAPs/
30. http://www.sdsc.edu/PMaC/Benchmark/maps_ping/
31. http://www.darpa.mil/ipto/programs/hpcs/