DTD 5
ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Future Generation Computer Systems xxx (2004) xxx–xxx

FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/future

A performance prediction framework for scientific applications

Laura Carrington*, Allan Snavely, Nicole Wolter

San Diego Supercomputer Center, University of California, San Diego, CA, USA

Abstract

This work presents the results of ongoing investigations in the development of a performance modeling framework, developed by the Performance Modeling and Characterization (PMaC) Lab at the San Diego Supercomputer Center. The framework is faster than traditional cycle-accurate simulation, more sophisticated than performance estimation based on system peak-performance metrics, and is shown to be effective on benchmarks and scientific applications. This paper focuses on one such functionality by investigating sensitivity studies to further understand observed and anticipated effect of both the architecture and the application in predicted runtime.

Keywords: Performance modeling; Performance prediction; HPC

1. Introduction and motivation

Performance of a parallel application on a High Performance Computing (HPC) machine is resultant from at least factors of algorithm, implementation, the compiler, operating system, underlying processor architecture, and interconnect technologies. Therefore, one might conclude that performance models for scientific applications on complex systems must account for all of the above system and application attributes. This work shows that a framework based on simplicity, including only the major factors in performance, can predict an application's performance with useful accuracy.

This framework is designed to have tools that combine simulation and analytical modeling to automate the entire performance prediction process for an application. The design implements easy to use tools that create an accurate model in a reasonable amount of time for users and centers. In previous work [6,7,22,23], this framework was described and validated to accurately model and improve understanding of the performance for small parallel scientific kernels and applications on different HPC architectures. In this research, the general framework is used to predict the performance of scientific applications on current HPC platforms with improved time cost, creating models in hours. The results were evaluated using sensitivity studies, to further explain the observed performance of the application.

The paper will progress as follows. A recap of the framework is described in Section 2, giving an overview of the different pieces of the framework and

* Corresponding author.
E-mail addresses: lnett@sdsc.edu (L. Carrington), allans@sdsc.edu (A. Snavely), wolter@sdsc.edu (N. Wolter).

0167-739X/$ – see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.future.2004.11.019

how they are used in performance prediction. Section 3 shows results of performance predictions for three different scientific applications. Section 4 illustrates processor and network investigations enabled by the framework on those applications. Section 5 describes background and related work, some of which this research is based on.

2. A performance modeling framework

In the pursuit of rapid, useful, and accurate performance models that can account for complexities of the memory hierarchy and work with all arbitrary applications on all arbitrary machines, the Performance Modeling and Characterization (PMaC) performance modeling framework's design is based on principles of isolation and simplicity. Measuring various performance factors in isolation enables independent performance investigations of each system feature as exhibited in the sensitivity studies of Section 4. The simplicity principle argues that the framework should be based on as *few* parameters as possible while still retaining accuracy. The framework is designed in such a way that it provides the ability to easily add and remove significant factors as needed to sufficiently depict a given application or system. The framework is composed of tools to automate each of the components and steps in the performance prediction of an application. This allows anyone to feed an application through the framework and arrive at a runtime prediction on any HPC system. A detailed description of the framework can be found in Snavely et al. [23].

Based on the hypothesis that a parallel application's performance is often dominated by two major factors: (1) single processor performance and (2) use of the network, the framework was developed to model these factors along with some of the features of modern, highly complex processor. Starting simple and only adding complexity when needed to account for observed performance, the framework consists of a single processor model, combined with a communication model (see Fig. 1). Clearly, there are other factors that can affect performance, but often processor and network performance are sufficient for accurate performance prediction (∼10% error) while adding more factors only increases the complexity of the model with nominal gains (∼1–2%) in accuracy [23].

The single-processor and communication models both use independent *Application Signatures* and *Machine Profiles*, which are combined using *Convolution Methods*. An Application Signature is a summary of the operations to be carried out by an application, including memory and communication access patterns, independent of any particular machine. Application Signatures are collected via traces. For the single-processor model, these are memory traces collected via the MetaSim Tracer [29]. For the communication model, these are MPI traces collected by MPIDtrace [30].

A Machine Profile is measurements of the rates at which a machine can perform basic operations, including message passing, memory loads and stores, and

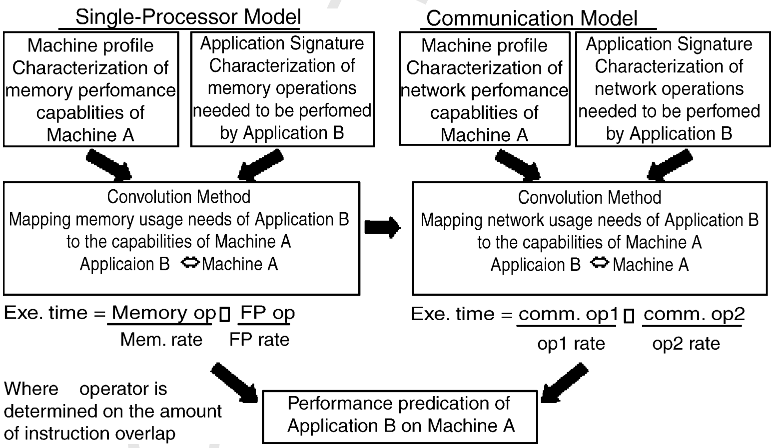

Fig. 1. Performance prediction framework for a parallel application.

floating-point operations, independent of any particular application. This data is collected via low level benchmarks or probes. To arrive at a performance prediction for an application, its Application Signature is mapped to the corresponding performance of the Machine Profile of the machine on which the application is being predicted, by the Convolution Methods. These mappings are automated using the MetaSim Convolver [31] for the single-processor model and Dimemas [30] for the communications model. The convolutions of the Application Signature and Machine Profile result in a predicted runtime, which the application should achieve on the target machine. Comparing a predicted run time with the actual runtime is the method we use for validating the model for that application [1]. Validation of models for three different scientific applications is presented next in Section 3.

3. HPC applications and model verification

In Sections 3.1–3.3, three scientific applications are fed through the framework to predict their performance on four different HPC architectures. Only small benchmarks were run on the target machines to collect the Machine Profiles. These benchmarks only consumed a few CPUs of the target machine but were used in predicting performance of an application running on hundreds of CPUs. The advantage of this is that typically in building large (>1000 CPUs) HPC machines a small prototype will be available long before the full system can be built. The benchmarks can be run on the prototype system and predict the full system before it is built.

3.1. Parallel Ocean Program (POP)

The Parallel Ocean Program (POP) was specifically developed to take advantage of high performance computer architectures. POP has been ported to a wide variety of systems including IBM Power3, and IBM Power4, Compaq Alpha server SC45, and Cray X1. POP is used for eddy-resolving simulations of the world oceans and for climate simulations as the ocean component of coupled climate models. POP is an ocean circulation model that solves the three-dimensional primitive equations for fluid motions on the sphere under hydrostatic and Boussinesq approximations. Spatial derivatives are computed using finite-difference discretizations, formulated to handle any generalized orthogonal grid on a sphere, including dipole, and tripole grids that shift the North Pole singularity into land masses to avoid time step constraints due to grid convergence.

The x1 dataset used in this study is a coarse resolution configuration that is currently being used in coupled climate models. The horizontal resolution is one degree (320×384) and uses a displace-pole grid with the pole of the grid shifted into Greenland and enhanced resolution in the equatorial regions. The vertical coordinate uses 40 vertical levels with smaller grid spacing near the surface to better resolve the surface mixed layer. This configuration does not resolve eddies, and therefore it requires the use of computationally–intensive sub grid parameterizations. This configuration is setup to be identical to the actual production configuration of the Community Climate System Model with the exception that the coupling to full atmosphere, ice and land models have been replaced by analytic surface forcing.

We applied the modeling framework to POP on the x1 dataset. Table 1 shows real versus model-predicted wall-clock execution times for several machines at several processor counts. POP execution times are more typically reported in seconds-per-simulation-day. An error of around 20% is considered acceptable to our user/funding agency for the purpose of getting a general idea of the application's performance on the target machine. The table of results shows that all predictions were below the acceptable limit and some were significantly lower. This confirms that the performance model for POP is robust on all the machines modeled.

3.2. Navy Layered Ocean Model (NLOM)

The Navy's hydrodynamic (iso-pycnal) non-linear primitive equation layered ocean circulation model has been used at NOARL for more than 10 years for simulations of the ocean circulation in the Gulf of Mexico, Caribbean, Pacific, Atlantic, and other seas and oceans. The model retains the free surface and uses semi-implicit time schemes that treat all gravity waves implicitly. NLOM consumes a significant portion of all cycles on the supercomputers run by DoD's High Performance Computing Modernization Program (HPCMP).

Table 1
Real vs. Predicted-by-Model wall-clock times for POP

# of CPUs	Blue Horizon (IBM PWR38-way)			Lemieux (Compaq SC45)			Longhorn (IBM PWR4)			Seaborg (IBM PWR316-way)			Cray X1		
	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time (s)	% Error
16	204.92	214.29	5	125.35	125.75	0	93.94	95.15	1	204.3	200.07	−2	9.21	9.79	6.3
32	115.23	118.25	3	64.02	71.49	11	51.38	53.30	4	108.16	123.10	14			
64	62.64	63.03	−1	35.04	36.55	4	27.46	24.45	−11	54.07	63.19	17			
128	46.77	40.60	−13	22.76	20.35	−11	19.65	15.99	−16	45.27	42.35	−6			

Where %Error = (Predicted − Real)/Real × 100.

A synthetic benchmark called synNLOM was designed by HPCMP to behave similar to the real NLOM application and has been used to evaluate vendors vying for DoD TI-02 procurements. Even though synNLOM is termed a “benchmark” it is really a representative production problem and runs for more than 1 h on 28 CPUs on an IBM Power3 system. The framework was applied to both synNLOM and “real” NLOM, those results are shown in Tables 2 and 3 below. SynNLOM was run with data from the Gulf of Mexico on 28 and 56 processors NLOM was run with same data on 56 and 112 CPUs.

Tables 2 and 3 show that for both applications the error was below the acceptable limit except for one case where the error was only slightly above the limit. Showing that even for large and complex applications the framework remains accurate in predicting performance of HPC machines.

3.3. Cobalt 60

Cobalt 60 is an unstructured Euler/Navier-Stokes flow solver that is routinely used to provide quick, accurate aerodynamic solutions to complex CFD problems. Cobalt 60 handles arbitrary cell types as well as hybrid grids that give the user added flexibility in their design environment. It is a robust HPC application that solves the compressible Navier-Stokes equations using an unstructured Navier-Stokes solver. It uses Detached Eddy Simulation (DES) which is a combination of Reynolds-averaged Navier-Stokes(RANS) models and Large Eddy Simulation (LES).

Cobalt 60 was modeled for two systems on four different processor counts; the results are seen in Table 4. The results show the error remained below the acceptable limit for all predictions validating the performance model for this application.

For all three applications, the accuracy of the performance models was confirmed by the fact that error was below the acceptable limit for all predictions but one, in which the error was only slightly above the limit. Once a performance model is verified as being accurate, one can investigate different performance factors of the hardware and how they affect the application's overall performance. These sensitivity studies were done on two of the application and described in Section 4.

Table 2
Performance prediction of NLOM application on two machines

# of CPUs	Blue Horizon (IBM PWR38-way)			Lemieux (Compaq SC45)		
	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time(s)	% Error
28	2385.8	2383.0	−0.1	1300.5	1220.3	−6.2
56	1220.4	1211.8	−0.7	809.7	618.7	−23.6

Table 3
Performance prediction of SynNLOM application on two machines

# of CPUs	Blue Horizon (IBM PWR38-way)			Lemieux (Compaq SC45)		
	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time(s)	% Error
56	4432	4611	4.0	2066.1	1816.4	−12.1
112	2356.0	2144.7	−9.0	1226.1	1218.6	−0.6

Table 4
Performance prediction of Cobalt 60 application on two machines

# of CPUs	Blue Horizon (IBM PWR3)			Lemieux (Compaq SC45)		
	Real time (s)	Predicted time (s)	% Error	Real time (s)	Predicted time(s)	% Error
16	1132.3	1145.5	1.2	766.4	786.3	2.5
32	553.8	568.9	2.7	337.0	284.7	−12.4
64	297.8	313.2	4.9	174.8	215.1	18.7
128	181.9	204.8	12.9	110.4	134.9	18.2

4. Performance sensitivity studies

Reporting the accuracy of performance models in terms of model-predicted time versus observed time (as in the Section 3) is a validating step for obtaining confidence in the model. A more interesting, useful, and challenging endeavor is to explain and quantify observed performance differences of an application on different architectures. The model can also be used to play “what if” scenarios, such as “what if the network had twice the bandwidth, how would that affect the application’s performance”. In this work, the performance difference of POP is investigated between two machines, Lemieux (SC45) and Blue Horizon (PWR3).

For example, it is clear from Table 1 in Section 3 that Lemieux (SC45) is faster across-the-board, for POP running the x1 data set, than Blue Horizon (PWR3). The question is why? Lemieux has faster processors (1000 MHz versus 375 MHz) with theoretical peak MFLOPS of 2000 versus 1500 for Blue Horizon. Lemieux also has a lower-latency network (measured ping-pong latency of about 9 μs versus about 20 μs) but Blue Horizon’s network has the higher bandwidth (ping-pong bandwidth measured at about 350 MB/s versus 260 MB/s with the PMaC probes). One can conjecture that POP performance is more sensitive to processor performance and network latency than network bandwidth, but with sensitivity studies we can go one step further and support that conjecture with data.

With a model that can accurately predict application performance based on properties of the code and the machine, precise modeling experiments can be carried out, such as those represented in Fig. 2 with details in Table 5. The model is used to perturb the Power3-based, Colony switch Blue Horizon (BH), system into the Alpha SC45-based, Quadrics switch (TCS), system by replacing components one by one and doing a prediction of the new hypothetical machine with each new component. The base system of BH is perturbed by changing network bandwidth, network latency, and processor “performance”, to finally arrive at a machine that represents the SC45. Note that processor “performance” captures the improved performance of the floating-point rate as well as the memory bandwidth. Fig. 2 represents a series of cases modeling the perturbation of BH to TCS, going from left to right. The bar

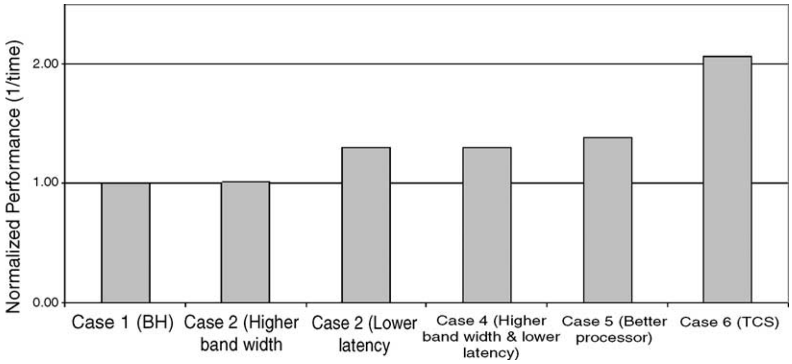

Fig. 2. Sensitivity study of POP on 16 CPUs.

for each case represents the performance of POP on 16 processors normalized to the performance of BH.

- *Case 1* is the base case normalized to the performance of BH.
- *Case 2* models the effect of reducing the bandwidth of BH's network to that of a single rail of the Quadrics switch. There is no discernable performance effect to the POP application, at this size, in changing in peak network bandwidth from 350 MB/s to 260 MBs.
- *Case 3* models the effect of reducing network latency of the Colony switch to that of the Quadrics switch. There is a significant performance improvement noted by switching the 20 μs latency of the Colony switch to 9 μs latency of the Quadrics switch. This is because the barotropic calculations in POP at this size are latency sensitive.
- *Case 4* uses Quadrics latency and bandwidth for completeness.
- *Case 5* models the Colony switch latencies and bandwidths but replace the Pwr3 processors and local memory subsystem with that of the Alpha SC45. There is a substantial improvement in performance due mainly to the faster memory subsystem of the Alpha. The Alpha can load stride-1 data from its L2 cache at about twice the rate of the Power3 and this benefits POP significantly.
- *Case 6* shows the values of TCS performance, processor and memory subsystem speed, network bandwidth and latency, as a ratio to BH's values.

The higher level point from the above exercise is that the model can quantify the performance impact of each machine hardware component. One can carry out this exercise for any size POP problem as well as for NLOM, Cobalt 60, or any application modeled via the framework.

As an abstraction from a specific architecture comparison study such as the above, one can use the model to generate a machine-independent performance sensitivity study. As an example, Fig. 3 indicates the performance impact on a 128 CPU POP run for quadrupling the speed of the CPU and memory subsystem (lumped together, we call this processor), quadrupling network bandwidth, cutting network latency by 4, and

Table 5
Model parameters for POP used in Fig. 2

Case number	Prediction (s)	CPU-memory subsystem ratio	NW ping–pong BW (MB/s)	SMP Node BW (MB/s)	NW ping–pong latency (μs)	SMP Node latency (μs)
1	42.07	1.00	350	370	19	19
2	41.71	1.00	269	552	19	19
3	32.46	1.00	269	552	5	4.6
4	32.43	1.00	350	370	5	4.6
5	30.41	1.69	350	370	19	19
6	20.35	1.69	269	552	5	4.6

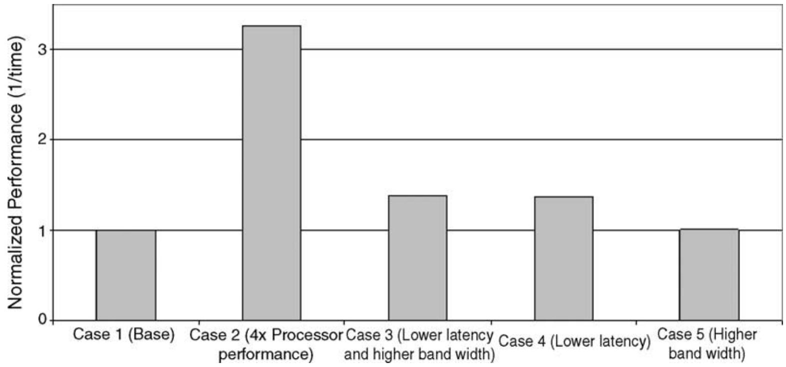

Fig. 3. POP Performance sensitivity study for 128 CPUs.

various combinations of these four-fold hardware improvements. Case 1 represents the base performance of POP run on Blue Horizon. Case 2 illustrates the performance effects that POP would see if the processor on Blue Horizon were to get a performance increase of four-fold. Case 3 represents Blue Horizon with a complete network upgrade with four-fold improvements to the network latency and bandwidth. Case 4 shows the performance effects of a network improvement localized to just a four-fold improvement in latency and Case 5 shows similar affects for improvements in just bandwidth. It is understood that given the gap between memory and floating-point performance on a processor that increasing both these components by an even factor of four is not realistic. But the results of Case 2 can show if processor performance is a significant factor worthy of further studies to split the individual components of memory and floating-point performance.

At this size, POP is quite sensitive to processor, (faster processor and memory subsystem) seen in the Case 2 results, and somewhat sensitive to latency (Case 4) because of the communications-bound, small-messages, barotropic portion of the calculation and fairly insensitive to bandwidth (Case 5). The higher-level impact is that performance models enable "what-if" examinations for implications of improving the target machine in various dimensions. Thus, purchasing upgrades or future machines to run this application would benefit the application most by focusing resources on better processors and lower latency networks.

Fig. 4 illustrates a similar study done on the application synNLOM, but this study provides "zoom in" on the processor performance factor for synNLOM. In the above results for POP, the processor improvements show modeled execution time decreases from having

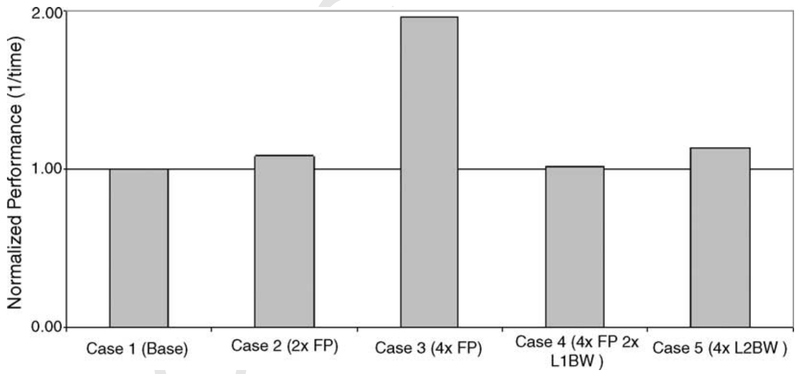

Fig. 4. synNLOM Performance sensitivity study for 28 CPUs.

a four-times better processor (Case 2) with respect to MHz (implying four-fold improvement to floating-point issue rate) but also implicit in “four-times better processor” is quadruple bandwidth and 1/4th latency to all levels of the memory hierarchy (unfortunately this may be hard or expensive to achieve architecturally!). Fig. 4 shows how much better a processor would perform relative to the Power 3 processor for synNLOM if it had Case 2: 2× issue rate; Case 3: 4× issue rate; Case 4: 2× issue rate and 2× faster L2 cache; Case 5: base issue rate of 4 × 375 MHz but 4× faster L2 cache. From the results in Fig. 4, it appears that SynLOM at this size is compute-bound between communication events and would benefit significantly from a faster processor clock, even without improving L2 cache. Not shown but discoverable via the model is that synNLOM is somewhat more network bandwidth sensitive than POP because it sends less frequent, larger messages.

The third example using application Cobalt 60, modeled performance sensitivity of 32 CPU Cobalt 60 to faster network and faster node, shown in Fig. 5. This study was conducted in a way similar to Fig. 4 with four-fold increases to processor performance, network latency, and network bandwidth. Case 1 represents the base performance of Cobalt 60 on Blue Horizon. Case 2 represents the performance increase of four-fold to the processor both floating-point rate and memory bandwidth. Case 3 illustrates the performance increases due to both network bandwidth and latency. Case 4 represents performance increase due to improved network latency and Case 5 shows performance increases due to improved network bandwidth. Case 2 shows Cobalt 60’s sensitivity to improvements in the processor performance at this size, this remains true at larger processor counts. Cases 3–5 illustrate how network performance upgrades would not benefit this application. Further studies could be performed to determine which component of the processor, memory or floating-point rate, have the most influence in application performance.

5. Background and related work

Methods for performance evaluations can be broken down into two areas [25]: structural models and functional and analytical models. Structural models use descriptions of individual system components and their interactions, such as detailed simulation models. The second area, functional and analytical models, separates the performance factors of a system to create a mathematical model.

The use of detailed or cycle-accurate simulators in performance evaluation has been used by many researchers [2,3,5,17,26]. Detailed simulators are normally built by manufactures during the design stage of an architecture to aid in the design. For parallel machines, two simulators might be used, one for the processor and one for the network. These simulators have the advantage of automating performance prediction from the user’s standpoint. The disadvantage is that these simulators are proprietary and often not available to HPC users and Centers. Also, because they capture all the behaviors of the processors, simulations can take on an upwards of 1,000,000 times longer, than the real runtime of the application [14]. This means, to simu-

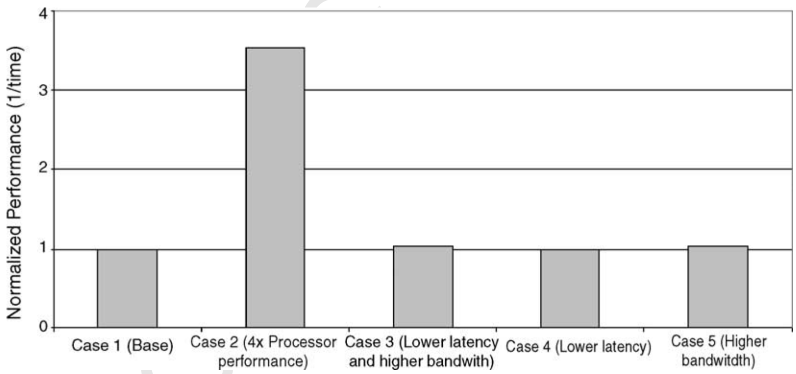

Fig. 5. Cobalt 60 Performance sensitivity study for 32 CPUs.

late 1 h of an application it could take approximately 114 years of CPU time. Direct execution methods are commonly used to accelerate architectural simulations [9] but they still can have large slowdowns. To avoid these large computational costs, cycle-accurate simulators are usually only used to simulate a few seconds of an application. This causes a modeling dilemma, for most scientific applications the complete behavior cannot be captured in a few seconds of a production run. Applications rarely spend all their time in one routine and their behavior may change as the application progresses through its simulation (in some cases the actual physics of the problem being solved changes).

Cycle-accurate simulators are limited to only work in modeling the behavior of the processor for which they were developed, so they are not applicable to other architectures. In addition, the accuracy of cycle-accurate simulation can be questionable. Gibson et al. [10] showed that simulators that model many architectural features have many possible sources for error, resulting in complex simulators that produce greater than 50% error. This work suggested that simple simulators are sometimes more accurate than complex ones.

In the second area of performance evaluation, functional and analytical models, the performance of an application on the target machine can be described by a complex mathematical equation. When the equation is fed with the proper input values to describe the target machine, the calculation yields a wall clock time for that application on the target machine. Various flavors of these methods for developing these models have been researched. Below is a brief summary of some of this work but due to space limitations it is not meant to be inclusive of all.

Saavedra and Smith [18–20] proposed applications modeling as a collection of independent Abstract FORTRAN Machine tasks. Each abstract task was measured on the target machine and then a linear model was used to predict execution time. In order to include the effects of memory system, they measured miss penalties and miss rates to include in the total overhead. These simple models worked well on the simpler processors and shallower memory-hierarchies of the mid 1990s. The models now need to be improved to account for increases in the complexity of parallel architectures including processors, memory subsystems, and interconnects.

For parallel system predictions, Mendes and Reed [15,16] proposed a cross-platform approach. Traces were used to record the explicit communications among nodes and to build a directed graph based on the trace. Sub-graph isomorphism was then used to study trace stability and to transform the trace for different machine specifications. This approach has merit and needs to be integrated into a full system for applications tracing and modeling of deep memory hierarchies in order to be practically useful today.

Simon and Wierun [21] proposed to use a Concurrent Task Graph to model applications. A Concurrent Task Graph is a directed acyclic graph whose edges represent the dependence relationship between nodes. In order to predict the execution time, it was proposed to have different models to compute the communication overhead, (FCFS queue for SMP and Bandwidth Latency model for MPI) with models for performance between communications events. As above, these simple models worked better in the mid 1990s than today.

Crovella and LeBlanc [8] proposed complete, orthogonal and meaningful methods to classify all the possible overheads in parallel computation environments and to predict the algorithm performance based on the overhead analysis. Our work adopts their useful nomenclature.

Xu et al. [27] proposed a semi-empirical multiprocessor performance prediction scheme. For a given application and machine specification, the application first is instantiated to thread graphs which reveal all the possible communications (implicit or explicit) during the computation. They then measured the delay of all the possible communication on the target machine to compute the elapsed time of communication in the thread graph. For the execution time, of each segment in the thread graph between communications, they use partial measurement and loop iteration estimation to predict the execution time. The general idea of prediction from partial measurement is adopted here.

Abandah and Davidson [1] and Boyd et al. [4] proposed hierarchical modeling methods for parallel machines that is kindred in spirit to our work, and was effective on machines in the early and mid 1990s.

A group of expert performance modelers at Los Alamos have been perfecting the analytical model of two applications important to their workload for years [11,12,13,28]. These models are quite accurate in their predictions, although the methods for creating them are time consuming and not necessarily easily done by non-expert user [24]. Also, the models require input related to the applications data set that is not automated.

6. Conclusions

The performance prediction framework has been proven effective for creating models of complex scientific applications. Performance predictions and sensitivity studies were exhibited and shown to be useful in determining which architectural features will best benefit a workload. It is reasonable to make procurement decisions based on the computational demands of the target workload. As a trivial example, if one's workload required more resources for the synLOM application and less for POP, one would be willing to spend more money to improve network bandwidth. It is reasonable to tune current systems and influence the implementation of near-future systems informed by the computational demands of the target workload with the performance information from the application models. It is also reasonable to design future systems based on the quantified performance implications of hardware features for characterized workloads.

Acknowledgments

This work was sponsored in part by the Department of Energy Office of Science through SciDAC award "High-End Computer System Performance: Science and Engineering". This work was sponsored in part by a grant from the Department of Defense High Performance Computing Modernization Program (HPCMP) and the National Security Agency. This research was supported in part by NSF cooperative agreement ACI-9619020 through computing resources provided by the National Partnership for Advanced Computational Infrastructure at the San Diego Supercomputer Center. Computer time was provided by the Pittsburgh Supercomputer Center, the Texas Advanced Computing Center, and the National Energy Research Scientific Computing Center. We would also like to acknowledge the European Center for Parallelism of Barcelona, Technical University of Barcelona (CEPBA) for their continued support of their profiling and simulation tools.

References

[1] G. Abandah, E.S. Davidson, Modeling the communication performance of the IBM SP2, in: Proceedings of the International Parallel Processing Symposium, April, 1996, pp. 249–257.

[2] R.S. Ballansc, J.A. Cocke, H.G. Kolsky, The Lookahead Unit, Planning a Computer System, McGraw-Hill, New York, 1962.

[3] L.T. Boland, G.D. Granito, A.V. Marcotte, B.V. Messina, J.W. Smith, The IBM system 360/Model9: storage system, IBM J. Res. Dev. 11 (1967) 54–79.

[4] E.L. Boyd, W. Azeem, H.H. Lee, T.P. Shih, S.H. Hung, E.S. Davidson, A hierarchical approach to modeling and improving the performance of scientific applications on the KSR1, in: Proceedings of the 1994 International Conference on Parallel Processing, vol. 3, 1994, pp. 188–192.

[5] D. Burger, T.M. Austin, S. Bennett, Evaluating Future Microprocessors: The Simplescalar Tool Set, Technical Report CS-TR-1996-1308, University of Wisconsin-Madison, 1996.

[6] L. Carrington, A. Snavely, X. Gao, N. Wolter, Performance Prediction Framework for Scientific Applications ICCS Performance Modeling Workshop, Melbourne, June, 2003.

[7] L. Carrington, N. Wolter, A. Snavely, A framework for Application Performance Prediction to enable Scalability understanding, Scaling to New Heights Workshop, Pittsburgh, May, 2002.

[8] M.E. Crovella, T.J. LeBlanc, Parallel Performance Prediction using Lost Cycles Analysis, SuperComputing 1994 (1994) 600–609.

[9] B. Falsafi, D.A. Wood, Modeling Cost/Performance of a Parallel Computer Simulator, ACM Transact. Model. Comput. Simul. 7 (1) (1997) 104–130.

[10] J. Gibson, R. Kunz, D. Ofelt, M. Horowitz, J. Hennessy, M. Heinrich, FLASH vs. (Simulated) FLASH: closing the Simulation Loop, in: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), November, 2000, pp. 49–58.

[11] A. Hosie, L. Olaf, H. Wasserman, Performance analysis of Wavefront Algorithms on Very-Large Scale Distributed Systems, Springer's Lect. Notes Control Inf. Sci. 249 (1999) 171.

[12] A. Hosie, L. Olaf, H. Wasserman, Scalability analysis of Multidimensional Wavefront Algorithms on Large-Scale SMP Clusters, in: Proceedings of Frontiers of Massively Parallel Computing'99, Annapolis, MD, February, 1999.

[13] D.J. Kerbyson, A. Hoisie, H.J. Wasserman, Modeling the performance of Large-Scale Systems, Keynote paper, UK Performance Engineering Workshop (UKPEW03) July, 2003.

[14] J. Lo, S. Egger, J. Emer, H. Levy, R. Stamm, D. Tullsen, Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading, ACM Transact. Comput. Syst. (August 1997).

[15] C.L. Mendes, D.A. Reed, Integrated Compilation and Scalability Analysis for Parallel Systems, IEEE PACT (1998).

[16] C.L. Mendes, D.A. Reed, Performance stability and prediction, in: IEEE/USP International Workshop on High Performance Computing, 1994.

[17] J.O. Murphey, R.M. Wade, The IBM 360/195, Datamation 16 (4) (1970) 72–79.

[18] R.H. Saavedra, A.J. Smith, Measuring Cache and TLB performance and their effect on Benchmark run times, IEEE Transact. Comput. 44 (10) (1995) 1223–1235.
[19] R.H. Saavedra, A.J. Smith, Analysis of Benchmark characteristics and Benchmark performance prediction, TOCS14 4 (1996) 344–384.
[20] R.H. Saavedra, A.J. Smith, Performance characterization of Optimizing Compilers, TSE21 7 (1995) 615–628.
[21] J. Simon, J. Wierun, Accurate performance prediction for Massively Parallel Systems and its applications, Euro-Par 2 (1996) 675–688.
[22] A. Snavely, N. Wolter, L. Carrington, Modeling application performance by Convolving Machine Signatures with Application Profiles, in: IEEE Fourth Annual Workshop on Workload Characterization, Austin, December, 2001.
[23] A. Snavely, N. Wolter, L. Carrington, R. Badia, J. Labarta, A. Purkasthaya, A framework to enable Performance Modeling and Prediction, Supercomputing (2002).
[24] A. Spooner, D. Kerbyson, Identification of performance characteristics from Multi-view Trace Analysis, in: Proceedings of the International Conference on Computational Science (ICCS), Part 3, vol. 2659, 2003, pp. 936–945.
[25] L. Svobodova, Computer System Performance Measurement and Evaluation Methods: Analysis and Applications, Elsevier, New York, 1976.
[26] G.S. Tjaden, M.J. Flynn, Detection, Parallel execution of independent instructions, IEEE Trans. Comput. C-19 (1970) 889–895.
[27] Z. Xu, X. Zhang, L. Sun, Semi-empirical Multiprocessor Performance Predictions, JPDC 39 (1996) 14–28.
[28] L. Yong, L.M. Olaf, H. Wasserman, Development and validation of a Hierarchical Memory Model incorporating CPU- and Memory-Operation Overlap, in: Proceedings of the First International Workshop on Software and Performance Santa Fe, NM, 1996, pp. 152–163.
[29] http://www.sdsc.edu/pmac/MetaSim/Tracer/tracer.html.
[30] http://www.cepba.upc.es/.
[31] http://www.sdsc.edu/pmac/MetaSim/Convolver/convolver.html.

Dr. Carrington works in the Performance Modeling and Characterization (PMaC) Lab at the San Diego Supercomputer Center. This work involves developing a framework to model and predict the performance of scientific applications on High Performance Computing (HPC) systems. Funded in part by a DoD grant to predict the performance of key DoD applications on a number of different HPC systems.

Dr. Snavely is an expert in high performance computing. He has contributed to the development of a number of strategies for working around the Von Neumann bottleneck to deliver fast time-to-solution for scientific applications. These include fundamental studies in modeling to understand the factors that effect performance; also architectural innovations including multithreaded computing, and computing with field-programmable gate arrays (FPGAs). Snavely's current research involves the design and optimization of complex systems (including supercomputers and computing Grids) drawing on principles of economics and statistics and leveraging reconfigurability. Snavely is leader of the Performance Modeling and Characterization Laboratory (PMaC) at the San Diego Supercomputing Center (SDSC), charged with understanding and addressing factors that affect performance on large supercomputers and Grids. Current grants include a DoD effort for developing accurate performance models for applications on HPC systems, one to participate in the Performance Evaluation Research Center (PERC) involving DoE labs and universities for which he is technical lead for modeling and analysis, an NSF STI grant to design Grid Benchmarks, and an NSF NGS grant to measure and model the performance of deep memory hierarchies. Snavely is a founding member of the Grid Benchmarking Research Group in the Global Grid Forum with a charge to develop meaningful metrics for Grid performance.

Nicole Wolter works for the Performance Modeling and Characterization Lab at the San Diego Supercomputer Center. Current research involves the performance impact of memory access patterns across different HPC platforms, and Performance modeling of scientific applications on HPC systems. Nicole has her Bachelor of Science Degree from San Diego State University in Computer Science.