VERIFYING LARGE-SCALE SYSTEM PERFORMANCE DURING INSTALLATION USING MODELING

Darren J. Kerbyson
Los Alamos National Laboratory
Performance and Architectures Laboratory (PAL), CCS-3
P.O. Box 1663, Los Alamos, NM 87545
djk@lanl.gov

Adolfy Hoisie
Los Alamos National Laboratory
Performance and Architectures Laboratory (PAL), CCS-3
P.O. Box 1663, Los Alamos, NM 87545
hoisie@lanl.gov

Harvey J. Wasserman
Los Alamos National Laboratory
Performance and Architectures Laboratory (PAL), CCS-3
P.O. Box 1663, Los Alamos, NM 87545
hjw@lanl.gov

Abstract In this paper we describe an important use of predictive application performance modeling - the validation of measured performance during a new large-scale system installation. Using a previously-developed and validated performance model for SAGE, a multidimensional, 3D, multi-material hydrodynamics code with adaptive mesh refinement, we were able to help guide the stabilization of the Los Alamos ASCI Q supercomputer. This system was installed in several stages and has a peak processing rate of 20-Teraflops. We review the salient features of an analytical model for SAGE that has been applied to predict its performance on a large class of Tera-scale parallel systems. We describe the methodology applied during system installation and upgrades to establish a baseline for the achievable "real" performance of the system. We also show the effect on overall application performance of certain key subsystems such as PCI bus speed and processor speed. We show that utilization of predictive performance models can be a powerful system debugging tool.

Keywords: Performance Modeling; Terascale Systems; Performance Validation; High Performance Computing.

1. Introduction

Performance modeling is a key approach that can provide information on the expected performance of a workload given a certain architectural configuration. It is useful throughout a system's lifecycle: starting with a design when no hardware is available for measurement, in procurement for the comparison of systems, through to implementation / installation, as well as being able to examine the effects of updating a system over time.

We have previously reported the development and validation of an analytical model that captures the performance and scaling characteristics of an important ASCI (Accelerated Strategic Computing Initiative) application [5]. We have also described one interesting use of the model to predict the effect on runtime of a key algorithmic change to the application enabling a different parallel decomposition method.

In this paper we report another interesting use of this same model. Los Alamos National Laboratory (LANL) has recently installed a Tera-scale computing system called ASCI Q that comprises 2048 compute servers with an interconnect fabric composed of federated switches. ASCI Q has a peak performance of 20-Teraflops. The installation of a system with such a large number of components is subject to a variety of both hardware- and software-related issues that effectively result in a "stabilization period" during which the system's performance may be sub-par. The question is: how does one identify sub-par performance in a large-scale parallel system, especially one that is larger than any previously available for testing. Performance observations made on a newly-installed system do not necessarily represent just the cost of processing the workload but often include temporary idiosyncrasies of the hardware and system software, i.e., bugs, faulty or poorly configured hardware components, and so on.

We report here our experiences using a performance model to validate the measured performance during system integration of ASCI Q. Several sets of measurements of the application performance were made on the system during installation over a period of months. Only after several iterations of hardware refinements and software fixes did the performance of the system achieve the performance predicted by the model. The model did not necessarily reveal precisely what hardware and/or software refinements were needed; however, it was ultimately the only way to determine that no such further refinements were necessary. During this process the model and corresponding system measurements exposed several important performance characteristics associated with the system, such as the effect of PCI bus speed and the effect of processor speed on the overall application performance. This work builds on earlier analysis during the initial stages of the installation of ASCI Q [8].

2. Performance Modeling

Performance modeling is an important tool that can be used by a performance analysist to provide insight into the achievable performance of a system on a given workload. It can be used throughout the life-cycle of a system, or of an application, from first design through to maintenance. For example:

Design: performance modeling can be used to quantify the benefits between alternatives when architectural details are being defined and hence examine trade-offs that arise.

Implementation: when a small system becomes available, perhaps in the form of a prototype, modeling can be used to provide an expected performance for systems of larger size.

Procurement: performance modeling can be used to compare competing systems by a purchaser. Measuring application performance is typically not possible on the systems being proposed due to either the scale of the system required being larger than anything available, or due to the system using next generation components which are not yet available.

Installation: performance modeling can provide an expectation of what level of performance should be achieved and hence verify that the system is correctly installed and configured. As we show in this work, this is an important aspect which to date is not routinely considered.

Upgrade or Maintenance: performance modeling can quantify the impact on possible upgrades prior to the changes being implemented.

Although the examples listed above are considered in terms of a system, most are equally applicable to application codes. For instance from a software perspective in the early development of an application it may be appropriate to compare alternative design strategies and to understand their impact on performance in advance of implementation.

A performance model should also mirror the development of the application and/or system. As details are refined through implementations, the performance model should also be refined. In this way, performance models can be used throughout the life-cycle. At Los Alamos we have used performance models in many of these ways: in the early design of systems, during the procurement of ASCI purple (expected to be a 100-Tflop system to be installed in 2004/5), to explore possible optimizations in code prior to implementation [5], to consider the impact on possible improvements in sub-system performance [6], and to compare large-scale systems (e.g. the Earth Simulator compared to ASCI Q [7]).

In general, there are two main components that contribute to a performance model:

System Characteristics - This includes computational aspects (processor clocks, functional units etc.), the memory hierarchy (cache configuration, memory bus speeds etc.), node configuration (processors per node, shared resources etc.), inter-processor communication (latency, bandwidth, topology), and I/O capabilities.

Workload Characteristics - This includes the application mix, their processing flow, their data structures, their use of and mapping to system resources, their frequency or use, and their potential for resource contention etc.

For modularity and model re-use, the characteristics of the system should ideally be described, and values obtained, independently of any application. Similarly the description of the characteristics of any workload should be described independently of specific systems. Thus, once a model for particular system has been developed, it can be used to examine performance on a multitude of applications. Similarly, application performance can be compared across systems without any alteration to the application model. This modular approach of hardware and software model separation has been taken in a number of modeling activities include the PACE system [11] for high performance computing, and also INDY [12] for E-commerce based applications.

Many of the performance modeling approaches currently being developed can be classified into two categories: those that use a trace of the application (collected from an application run), and those that generate a trace the application activities during its execution. We term these two approaches as *re-play* and *pre-play* respectively.

Replay approaches take an application trace as their input and effectively replay the trace within a modeling environment which combines the trace-events with the characteristics of the system(s) being studied. Since the trace input is usually specific to a certain problem size and processor count, replay approaches cannot be used to fully explore the impact of scalability. Replay approaches include Dimemas [3], the Trace Model Evaluator at Los Alamos, and also the approach taken at San Diego Supercomputing Center [1].

Pre-play approaches use an abstract representation of the application and effectively generate a trace of events that should occur during the application execution. These events are again combined with the characteristics of the system within the modeling environment. Pre-play approaches tend to encapsulate the scaling behavior of the applications with their models parameterized in the same way as the application. Thus allowing a higher degree of performance scenarios to be studied. Pre-play approaches include PACE [11] and INDY [12].

The approach we take is application centric and fits into the pre-play category. It involves the understanding the processing flow in the application, the key data structures, how they use and are mapped to the available resources, and also its scaling behavior. An analytical performance model of the application is constructed from this understanding. The aim is to keep the model of the application as general as possible but parameterized in terms of the application's key characteristics.

Our view is that a model should provide insight into the performance of the application on available as well as future systems with reasonable accuracy. Hardware characteristics should not be part of the application model but rather be specified as a separate component. For instance a model for inter-processor communication may be parameterized in terms of the message size. The actual model may take the form of a simple linear analytical expression in terms of latency and bandwidth, or be more complex. Hardware characteristics may use measurements made by micro-benchmarks, or specified by other means.

An application performance model needs to be validated against measurements made on one or more systems for several configurations (or data sets). Once a model has been validated it can be used to explore performance and to provide insight into new performance scenarios. An overview of a performance model for SAGE and its validation results are given in Section 4.

3. The Alpha-Server ES45 Supercomputing System

ASCI Q consists of 2048 AlphaServer ES45 nodes. Each node contains four 1.25-GHz Alpha EV68 processors internally connected using two 4-GB/s memory buses to 16 GB of main memory. Each processor has an 8-MB unified level-2 cache, and a 64-KB L1 data cache. The Alpha processor has a peak performance of 2 floating point operations per cycle. Thus the Q machine has a peak performance of 20-Tflop.

Nodes are interconnected using the Quadrics QsNet high-performance network. This network boasts high-performance communication with a typical MPI latency of 5μs and a peak throughput of 340-MB/s in one direction (detailed measured performance data are discussed in Section 5). The Quadrics network contains two components - the Elan network interface card (NIC), and the Elite switch. The Elan/Elite components are used to construct a quaternary fat-tree topology - an example is shown in Figure 1. A quaternary fat-tree of dimension n is composed of 4^n processing nodes and $n.4^{n-1}$ switches interconnected as a delta network. Each Elite switch contains an internal 16x8 full crossbar. A detailed description of the Quadrics network can be found in [14].

In order to implement a fat-tree network a single NIC is used per node in addition to a number of Elite switch boxes. The Elite switches are packaged

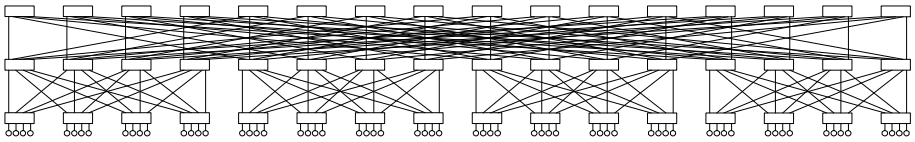

Figure 1. Network topology for a dimension 3 quaternary fat-tree network with 64 nodes.

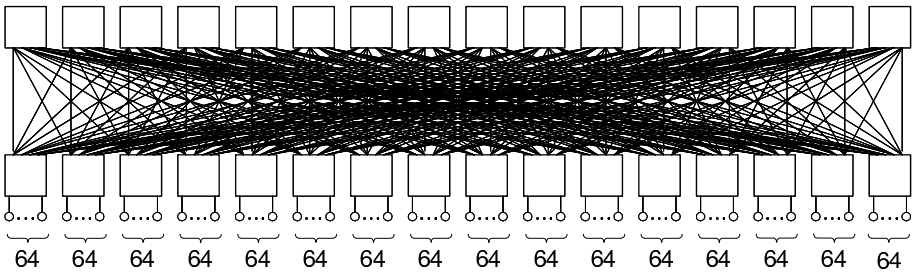

Figure 2. Interconnection of a federated Quadrics network for a dimension 5 fat-tree.

in 128-way boxes which by themselves implement a dimension 3 fat-tree. A 1024-node system, as shown in Figure 2, requires two rows of 16 switch boxes.

Using multiple independent networks, also known as "rails" is an emerging technique to overcome bandwidth limitations and to enhance fault tolerance [2]. The Q machine contains two rails, i.e. two Elan cards on separate PCI interfaces per node, and two complete sets of Elite switches.

4. The Application and the Model

The application used to analyze the performance of ASCI Q is SAGE (SAIC's Adaptive Grid Eulerian hydrocode). It is a multidimensional (1D, 2D, and 3D), multimaterial, Eulerian hydrodynamics code with adaptive mesh refinement (AMR) consisting of 100,000+ lines of Fortran 90 code using MPI for interprocessor communications. It comes from the LANL Crestone project, whose goal is the investigation of continuous adaptive Eulerian techniques to stockpile stewardship problems. SAGE has also been applied to a variety of problems in many areas of science and engineering including: water shock, energy coupling, cratering and ground shock, stemming and containment, early time front end design, explosively generated air blast, and hydrodynamic instability problems [16]. SAGE represents a large class of production ASCI applications at Los Alamos that routinely run on 1,000's of processors for months at a time.

A detailed description of SAGE, the adaptive mesh processing, and the characteristics of its parallel scaling were described previously in which we devel-

oped and validated the performance model [5]. Table 1 gives a summary of the validation results in terms of average and maximum prediction errors across all processor configurations measured. It can be seen that the model is highly accurate with an average prediction error of 5% and maximum of 11% being typical across all machines.

Table 1. SAGE performance model validation results.

System	Configurations tested	Processors tested (maximum)	Error (maximum)	Error (average)
ASCI Blue (SGI O2K)	13	5040	12.6	4.4
ASCI Red (Intel Tflops)	13	3072	10.5	5.4
ASCI White (IBM SP3)	19	4096	11.1	5.1
Compaq AlphaServer ES40	10	464	11.6	4.7
Cray T3E	17	1450	11.9	4.1

5. Use of the SAGE model to validate system performance

The model is parametric in terms of certain basic system-related features such as the sequential processing time and the communication network performance; these had to be obtained via measurements on a small system and are listed in Table 2 and Table 3 respectively. The SAGE model is based on weak scaling in which the global problem size grows proportionally with the number of processors. The subgrid size remains constant at approximately 13,500 cells per subgrid.

Table 2. Measured ES45 Performance Parameters for the SAGE Model.

Parameter	1-GHz Alpha EV68	1.25-GHz Alpha EV68
$T_{comp}(E)(s)$	0.38	0.26
$T_{mem}(P)(\mu s)$	$\begin{cases} 1.8 & P = 2 \\ 4.8 & P > 2 \end{cases}$	$\begin{cases} 1.7 & P = 2 \\ 3.3 & P > 2 \end{cases}$

The model is based on a static analysis of the code and includes a summation of the time to process a single cell (multiplied by the number of cells per processor) and the communication of boundary data between processors. The amount of overlap between messaging and computation is negligible. The model is parameterized in terms of dynamic aspects of the code - i.e. those features which are not known through a static analysis. These include: the

Table 3. Measured Communication Performance Parameters for the SAGE Model.

Parameter	33-MHz PCI bus	66-MHz PCI bus
$L_c(S)(\mu s)$	$\begin{cases} 9.0 & S < 64 bytes \\ 9.70 & 64 \leq S \leq 512 \\ 17.4 & S > 512 \end{cases}$	$\begin{cases} 6.11 & S < 64 bytes \\ 6.44 & 64 \leq S \leq 512 \\ 13.8 & S > 512 \end{cases}$
$1/B_c(S)(ns)$	$\begin{cases} 0.0 & S < 64 bytes \\ 17.8 & 64 \leq S \leq 512 \\ 12.8 & S > 512 \end{cases}$	$\begin{cases} 0.0 & S < 64 bytes \\ 12.2 & 64 \leq S \leq 512 \\ 8.30 & S > 512 \end{cases}$

number of cells, the number of newly adapted cells, and also any load-balancing across processors. These "histories" need to be known on a cycle by cycle basis if the model prediction is to be accurate.

The installation process required that the model predict performance for the first phase of ASCI Q with 2 different PCI bus speeds (initially 33-MHz and later 66-MHz). The PCI bus speed determines the available bandwidth between the Quadrics NIC and processor memory and has a significant impact on performance. In addition, when two NICs are present within the node (in a 2-rail system), the asymptotic bandwidth increases by approximately 180% if simultaneous messages can take advantage of the two rails [14]. Individual messages are not striped across rails.

Later in the installation process, the Alpha processors were upgraded from a clock speed of 1-GHz to 1.25-GHz. The faster processors also had an increased L2 cache capacity of 16MB in comparison to the 8MB cache of the slower processors. Thus during the installation there were actually three configurations of processors and PCI bus speeds: initially a 1-GHz processor with a 33-MHz bus, a 1-GHz processor with a 66-MHz bus, and finally a 1.25-GHz processor with a 66-MHz bus.

5.1 Expected Performance

The performance model was used to provide the expected performance of SAGE on the ES45 system with a 33-MHz and 66-MHz PCI bus and a 1-GHz and 1.25-GHz Alpha EV68 processors. These predictions are shown in Figure 3.

We note the following observations: 1) since the runs of SAGE were performed for weak scaling, the time should ideally be constant across all processor configurations; 2) the model predicts that the two-fold improvement in PCI bus speed results in only a 20% performance improvement in the code overall; 3) the 25% improvement in processor clock speed results in a 22% performance improvement overall; 4) the SAGE cycle time is predicted to plateau above 512

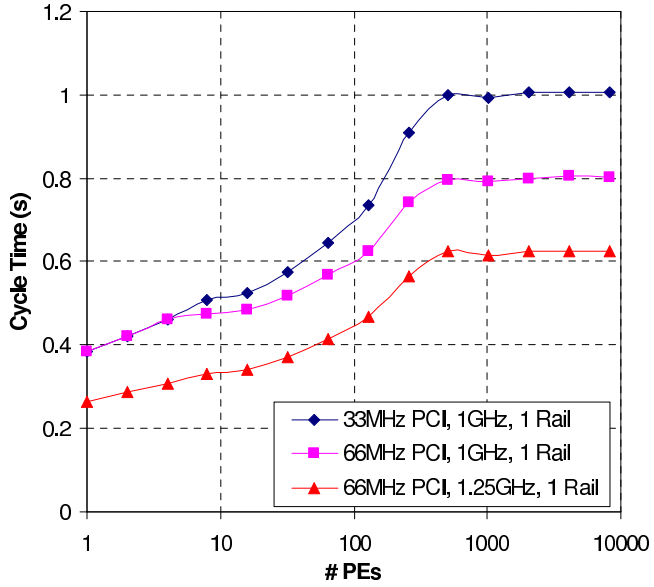

Figure 3. Performance predictions of SAGE on an ES45 system with QsNet.

processors – this is the point at which all gather/scatter communications are out-of-node.

5.2 Measured Performance

Table 3 summarizes the test conditions on each test date and the refinements made from one test to another. There are three distinct phases of the installation representing: initial hardware of 1-GHz processors with a 33-MHz PCI bus (tested between Sept and Oct '01), 1-GHz processors with a 66-MHz bus (tested between Jan and April '02), and the final hardware of 1.25-GHz processors with a 66-MHz bus (tested between Sept '02 and May '03).

The performance of SAGE was measured at several points after the installation of the initial hardware had taken place: as soon as the machine was up and running (Sept. 9th), after a O/S upgrade and faulty hardware was replaced (Sept. 24th), and after an O/S patch (Oct. 24th). The upgrades that were most significant in terms of performance included bug fixes to the Quadrics RMS resource scheduling software and O/S patches that affected the priority of a process that determined the allocation of the two rails. Interestingly, this affected both 1- and 2-rail performance. These three sets of measurements, which are based on the 1-GHz processors with a 33-MHz PCI bus, are compared with the model in Figure 4

Table 4. Summary of Test Conditions.

Date	# nodes in system	Performance issues
9-Sept-01	128	Some faulty nodes and communication links resulted in poor communication performance, especially on 2 rails
24-Sept-01	128	Faulty hardware replaced, still poor 2-rail performance
24-Oct-01	128	2-Rail OS patch improved Quadrics Performance
04-Jan-02	512	PCI bus upgraded to 66 MHz; SAGE performance at pre-Oct-24 (not all nodes successfully ran at 66 MHz)
02-Feb-02	512	All nodes at 66 MHz but some configured out causing lower performance in collective communications
20-April-02	512	All nodes configured in, collectives improved
21-Sept-02	1024	Processors upgraded to 1.25-GHz. 1st Phase of ASCI Q. Performance lower than expected on processor counts above 512
25-Nov-02	1024	2nd phase of ASCI Q, performance consistent with 1st phase
27-Jan-03	1024	Impact of operating system events reduced, performance significantly improved
1-May-03	2048	First test of full-system

The corresponding model prediction and measurements based on the 1-GHz processors after the PCI bus was upgraded to 66-MHz are shown in Figure 5. Initially (Jan 4th), not all nodes ran at 66-MHz. By Feb 2nd this had been resolved; however, not all nodes were available for testing. The Quadrics QsNet requires contiguous nodes in order to use its hardware-based collective operations. When nodes are configured out then a software component in the collectives is required which reduces overall performance. By April 20th all nodes were configured in and SAGE achieved the performance predicted by the model for all configurations except for the largest count of 512 nodes.

The performance of the system was again measured after upgrading the processors to 1.25-GHz, and the system increasing in size, first to separate two segments of 1024 nodes each and then to a combined 2048 node system. The first measurements made on each of the two segments (Sept. 21st, and Nov. 25th) resulted in performance highly consistent with each other. However, a major performance issue was identified concerning the impact of the operating system within each cluster of 32 nodes. This resulted in very poor performance on applications with synchronization requirements. These effects were minimized by reducing the number of operating system daemons, reducing the frequency of some monitoring activities, and configuring out 2 nodes per 32-node cluster [15]. The performance measured after this (Jan. 27th) showed much improved performance, very close to the model on the highest processor

counts. In addition, the first test on the full ASCI Q system (May 1st) resulted in performance consistent with that in Jan. '03. This data has not been included in Figure 6 as only a few measurements were taken at this point. The minimization of the impact of the operating system is ongoing.

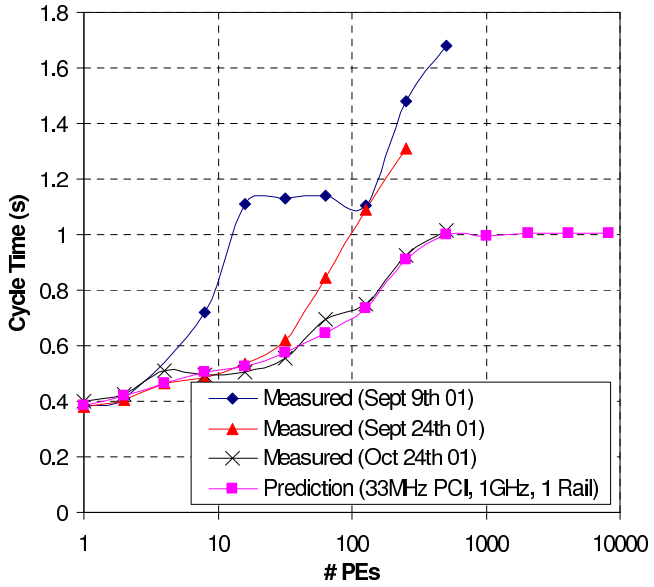

Figure 4. Measured performance of SAGE (33-MHz PCI bus, 1-GHz processors) compared with model predictions using a single rail.

The differences between the model predictions and the final set of measurements obtained for each of the three installation phases are shown in Figure 7. Note that in the difference is small in all but the case of the largest processor counts. We expect the performance on largest configurations to improve after further operating system optimizations. The average difference across all the data shown in Figure 7, between the measurements and model predictions, is 3.7%.

Figures 4, 5 and 6 show that only after all the upgrades and system debugging had taken place that the measurements closely matched the expected performance. When the measured data matched the modeled data there was some confidence in the machine performing well. Without the model, it would have been difficult to know conclusively when to stop debugging, or more importantly when not to. When differences did occur between the model and measurements, microkernel benchmarks were run on the computational nodes and the communication network to help identify the source of the problem. This was especially important in the minimization of operating system effects

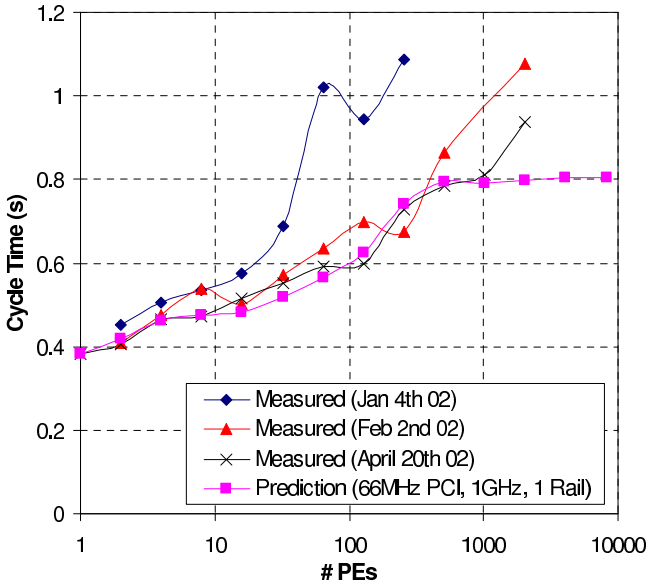

Figure 5. Measured performance of SAGE (66-MHz PCI bus, 1-GHz processors) compared with model predictions using a single rail.

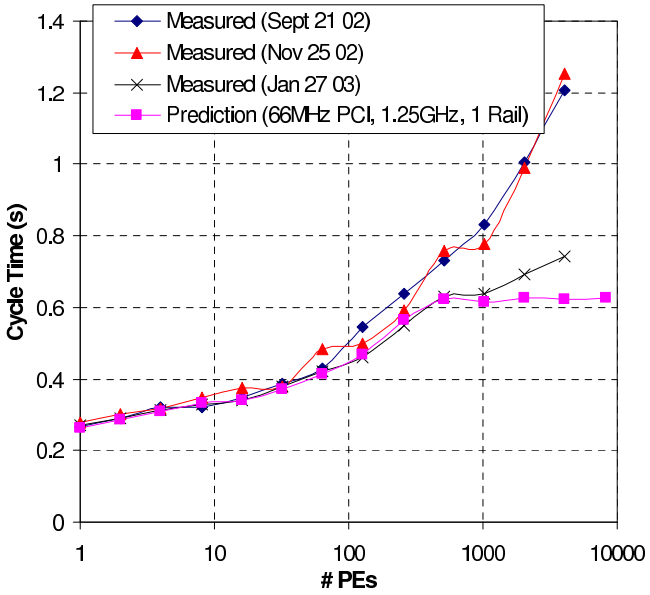

Figure 6. Measured performance of SAGE (66-MHz PCI bus, 1.25-GHz processors) compared with model predictions using a single rail.

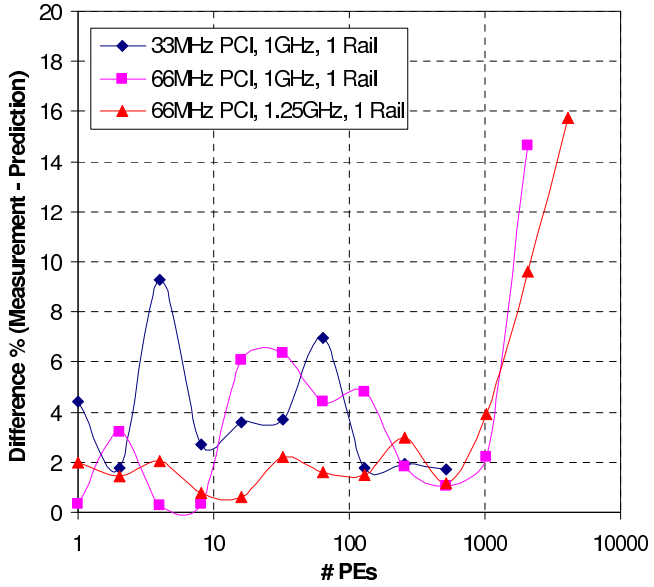

Figure 7. Difference between measurements and predictions for SAGE.

that resulted in a significant performance improvement on very large processor counts [15].

6. Summary

Our team's research over the last few years has focused on the development of analytical performance models for the ASCI workload. It has been said that modeling and predicting the performance of large-scale applications on HPC systems, is one of the great, unsolved challenges for computer science [13]. Clearly, ASCI has a critical need for information on how best to map a given application to a given architecture, and performance modeling is the only means by which such information can be obtained quantitatively. Our approach has been successfully demonstrated for the 100,000-line+ adaptive mesh code reported here, for structured [4], and unstructured mesh transport codes [9], and for a Monte-Carlo particle transport code [10].

The work reported in this paper represents a small but important step in applying our performance models in a very practical way. We expect that ASCI platforms and software will be performance engineered, and that models will provide the means for this. The models can play a role throughout a system's lifecycle: starting at design when no hardware is available for measurement, in procurement for the comparison of systems, through to implementation / installation, and to examine the effects of updating a system over time. At each point

the performance model provides an expectation of the achievable performance with a high level of fidelity. The SAGE performance model has been used for procurement purposes but company-sensitive information precludes disclosure of this in the literature. We can report, as here, how the model becomes the tool for assessing machine performance. Implementation milestone tests related to ASCI Q contractual obligations were based partially on comparison of observed data with predictions from the SAGE model, in a manner similar to the process described in this paper.

When installing a new system, refinements to both the software system, and hardware components, are often necessary before the machine operates at the expected level of performance. The performance model for SAGE has been shown to be of great use in this process. The model has effectively provided the performance and scalability baseline for the system performance on a realistic workload. Initial system testing showed that its performance was almost 50% less than expected. After several system refinements and upgrades over a number of months, the achieved performance matched closely the expectation provided by the model. Thus, performance models can be used to validate system performance.

Acknowledgments

Los Alamos National Laboratory is operated by the University of California for the National Nuclear Security Administration for the US Department of Energy.

References

[1] L. Carrington, A. Snavely, N. Wolter, X. Gao, A Performance Prediction Framework for Scientific Applications, in *Proc. Workshop on Performance Modeling and Analysis*, ICCS, Melbourne, Australia, June 2003.

[2] S. Coll, E. Frachtenberg, F. Petrini, A. Hoisie and L. Gurvits, Using Multirail Networks in High-Performance Clusters, in *Proc. of Cluster2001*, Newport Beach, CA, 2001.

[3] S. Girona, J. Labarta and R.M. Badia, Validation of Dimemas communication model for MPI collective operations, in *Proc. EuroPVM/MPI2000*, Balatonfured, Hungary, September 2000.

[4] A. Hoisie, O. Lubeck and H.J. Wasserman, Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications, *Int. J. of High Performance Computing Applications*, **14** (2000) 330–346.

[5] D.J. Kerbyson, H.J. Alme, A. Hoisie, F. Petrini, H.J. Wasserman and M.L. Gittings, Predictive Performance and Scalability Modeling of a Large-scale Application, in *Proc. SC2001*, Denver, CO, 2001.

[6] D.J. Kerbyson, H.J. Wasserman and A. Hoisie, Exploring Advanced Architectures using Performance Prediction, in *Innovative Architectures for Future Generation High-Performance Processors and Systems* (A. Veidenbaum and K. Joe, Eds), IEEE Computer Society, 2002.

[7] D.J. Kerbyson, A. Hoisie and H.J. Wasserman, A Comparison between the Earth Simulator and AlphaServer Systems using Predictive Application Performance Models, in *Proc. of Int. Parallel and Distributed Processing Symposium (IPDPS)*, Nice, France, April 2003.

[8] D.J. Kerbyson, A. Hoisie and H.J. Wasserman, Use of Predictive Performance Modeling during Large-Scale System Installation, Parallel Processing Letters, World Scientific Publishing Company, 2003.

[9] D.J. Kerbyson, A. Hoisie and S.D. Pautz, Performance Modeling of Deterministic Transport Computations, in *Performance Analysis and Grid Computing*, Kluwer, 2003.

[10] M. Mathis, D.J. Kerbyson, A. Hoisie and H.J. Wasserman, Performance Modeling of MCNP on Large-Scale Systems, in *Proc. Los Alamos Computer Science Institute Symposium*, Santa Fe, NM, Oct. 2002.

[11] G.R. Nudd, D.J. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper and D.V. Wilcox, PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems, *Int. J. of High Performance Computing Applications*, **14** (2000) 228–251.

[12] E. Papaefstathiou, Design of Performance Technology Infrastructure to Support the Construction of Responsive Software, in *Proc. 2ns ACM Int. Workshop on Software and Performance (WASP)*, Ottawa, Canada, (2000) 96–104.

[13] See http://perc.nersc.gov/main.htm.

[14] F. Petrini, W.C. Feng, A. Hoisie, S. Coll and E. Frachtenberg, The Quadrics Network, *IEEE Micro*, **22(1)** (2002) 46–57.

[15] F. Petrini, D.J. Kerbyson and S. Pakin, The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q, in *Proc. of SC2003*, Phoenix, AZ, 2003.

[16] R. Weaver, Major 3-D Parallel Simulations, BITS - Computing and communication news, Los Alamos National Laboratory, June/July, (1999) 9–11.