LA-UR-01-2501

Approved for public release;
distribution is unlimited.

Title: PREDICTIVE PERFORMANCE AND SCALABILITY MODELING OF A LARGE-SCALE APPLICATION

Author(s):
Darren J. Kerbyson, CCS-3
Henry J. Alme, CCS-3
Michael L. Gittings, X-2
Adolfy Hoisie, CCS-3
Fabrizio Petrini, CCS-3
Harvey J. Wasserman, CCS-3

Submitted to:
SuperComputing 2001
Denver, CO
November 2001

Los Alamos
NATIONAL LABORATORY

Predictive Performance and Scalability Modeling of a Large-Scale Application

D.J. Kerbyson[*], H.J. Alme[*], A. Hoisie[*], F. Petrini[*], H.J. Wasserman[*], M. Gittings[**]

[*]Parallel Architectures and Performance Team
CCS-3 Modeling, Algorithms and Informatics Group,
Los Alamos National Laboratory, Los Alamos, NM 87545

[**]SAIC and Los Alamos National Laboratory

Abstract

In this work we present an analytical model that encompasses the performance and scaling characteristics of an important ASCI application. SAGE (SAIC's Adaptive Grid Eulerian hydrocode) is a multidimensional hydrodynamics code with adaptive mesh refinement. The model is validated against measurements on several systems including ASCI Blue Mountain, and a yet to be announced Compaq System showing high accuracy. It is parametric - basic machine performance numbers (latency, MFLOPS rate, bandwidth) and application characteristics (problem size, decomposition method, etc.) serve as input. The model adds insight into the performance of current systems, revealing bottlenecks, and is able to show where tuning efforts would be most effective. It also allows prediction of performance on future systems which is important for both application and system architecture design as well as for the procurement of supercomputer architectures.

Keywords: Performance analysis, full application codes, parallel system architecture, Teraflop scale computing.

1. Introduction

SAGE (SAIC's Adaptive Grid Eulerian hydrocode) is a multidimensional (1D, 2D, and 3D), multimaterial, Eulerian hydrodynamics code with adaptive mesh refinement. The code uses second order accurate numerical techniques, and 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. SAGE represents a large class of production ASCI applications at Los Alamos that routinely run on 2000-4000 processors for months at a time. Examples of SAGE use to set up, execute, and analyze complex extremely large hydrodynamic simulations may be found in [6].

SAGE is a large-scale parallel code written in Fortran 90, using MPI for inter-processor communications. Early versions of SAGE were developed for vector architectures. Optimized versions of SAGE have recently been ported to all teraflop-scale ASCI architectures, as well as the CRAY T3E and Linux-based cluster systems.

This document describes a performance and scalability analysis of SAGE. One essential result is the development of a performance model that encapsulates the code's crucial performance and scaling characteristics. The performance model has been formulated from an analysis of the code, inspection of key data structures, and analysis of traces gathered at run-time. The model has been validated against a number of ASCI machines with high accuracy. The model is applied in this work to predict the

performance of SAGE on extreme-scale future architectures, such as clusters of SMPs. Also included is the application of the model for predicting the performance of the code when algorithmic changes are implemented, such as using a different parallel data decomposition.

There are few existing performance studies that extend to full codes (for instance [7]), many tend to consider smaller applications especially in distributed environments (e.g. [5]). This paper represents an example of performance engineering applied to a full-blown code. SAGE has been analyzed, a performance model proposed and validated on all architectures of interest. The validated model is utilized for point-design studies involving changes in the architectures on which the code is running and in the algorithms utilized in the code. A predictive performance model of another important ASCI application is described in previous work [4].

2. Description of the Essential Characteristics of the SAGE Code

In this section we describe the characteristics of SAGE that affect its performance and scaling behavior. In particular, the spatial data decomposition, the scaling of the subgrid, and the common operations within a code cycle are analyzed.

2.1 Parallel Spatial Decomposition in SAGE

The code uses a spatial discretization of the physical domain utilizing Cartesian grids. The spatial domain is partitioned across processors in "subgrids" such that the first processor is assigned the first E cells in the grid (indexed in dimension order – X,Y,Z), the second processor is assigned the next E cells and so on. The assignment is actually done in blocks of 2x2x2 as illustrated in Figure 1. Figure 1a) shows the index ordering of a 2x2x2 block, and Figure 1b) illustrates the approximate assignment of cells over 4 processors (PEs). Note that processors contain cells which are either:

a) internal –all its neighbor cells are contained on the same PE,
b) boundary – has one or more surfaces ("faces") which are on the spatial domain's physical boundary, or
c) inter-processor boundary – neighbor cells in physical space belong to subgrids contained on different PEs (in any dimension).

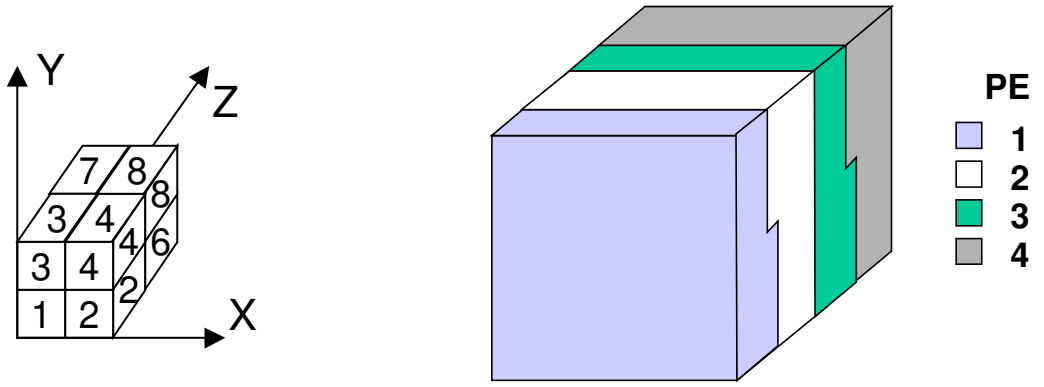

Figure 1: Cell assignment to processors in SAGE. a) ordering of a 2x2x2 cube, b) example assignment on four PEs.

A library designed for the communication pattern of the code is used to handle the necessary communications within SAGE. This includes the common MPI operations of allreduce and broadcast for instance, as well as two main application specific communication kernels: gather (get data) and scatter

(put data) operations. These operations are used when the processor requires an update of its subgrid with local cell information and processor boundary data. The library uses a notion of *tokens* to record where all the necessary data can be found for each individual processor. A token is defined for each data neighbor direction (6 in total) for individual cells and for entire faces of the subgrids. Each token contains information on:

- subgrid boundaries,
- data held locally within a processor,
- data held off processor (requiring communication), and
- data required off processor (also requiring communication).

2.2. Scaling of the Subgrid

The subgrid size is a function of the input number of cells per PE which is specified in the input deck. SAGE assigns this number of cells to each PE. We are concerned with "weak scaling" in this analysis, in which the problem size scales, with each PE doing approximately the same amount of work.

The decomposition of the entire spatial grid onto PEs is done in "slabs" (2-D partitioning), as suggested in Figure 1b). Each slab is uniquely assigned to one processor.

Taking the number of cells in each subgrid to be E, then the total grid volume V in cells is:

$$V = E*P \tag{1}$$

The volume of each subgrid:

$$E=l*L^2 \tag{2}$$

where P is the number of PEs, l is the short side of the slab (in the Z direction) and L the side of the slab in X and Y directions (assuming a square cross-section). The surface L^2 of the slab in the X-Y plane is:

$$L=(EP)^{1/3} \qquad l=E/L^2 \;=E/(EP)^{2/3}=E^{1/3}*P^{-2/3} \qquad L^2 \sim (E.P)^{2/3} \tag{3}$$

From equation 3 it can be seen that the surface increases as $P^{2/3}$. This subgrid surface is directly proportional to the maximum data size that is communicated between PEs on a gather/scatter operation.

The maximum size of a surface that a PE will have is constrained by E. In fact, since the assignment of cells to PEs is done in 2x2x2 blocks, the maximum surface is $E/2$, at which point the slab degenerates to a "foil" with a thickness of 2 cells. It is possible for the surface of the full spatial domain to be greater than E - thus resulting in a surface being assigned to more than one PE. This will lead to physically neighboring data cells assigned to logically distant PEs. Hence communications will take place between more distant processors. The total communication requirements will remain as $(E.P)^{2/3}$, but will be dealt with by more than one PE.

Consider again the volume of the entire grid:

$$V = E.P = l*L^2 \tag{4}$$

This is partitioned across PEs such that there will be $L/2P$ foils of width 2 on each PE, or:

$$(E.P)^{1/3}/2P \;=\; (E/8P^2)^{1/3} \tag{5}$$

When this value is less than one, a processor contains less than a single foil, i.e. when

$$P > SQRT(E/8) \tag{6}$$

The maximum distance between the processors that hold a foil is:

$$\left\lceil \left(E/8P^2\right)^{-1/3} \right\rceil \tag{7}$$

The minimum distance between the processors that hold that foil, the "PE distance" (*PED*):

$$PED = MAX\left(\left\lceil \left(E/8P^2\right)^{-1/3} \right\rceil - 1, 1 \right) \tag{8}$$

Thus when a processor is not assigned a full surface of the entire spatial domain, boundary exchange will involve all the processors that own the boundary, located at a logical distance *PED* apart along the Z axis.

The subgrid surface (L^2), the actual inter-processor boundary area owned by a processor (due to the slab degenerating to a foil and the subsequent splitting of the boundary amongst the processors within PED), the "PE surface", and the PED are shown in Figure 2. The PE surface achieves a maximum after 32 PEs. The value of *E* utilized in the plot of the PE surface was 13,500 cells per processor, a typical value for SAGE runs. It should also be noted that the subgrid surface approximately equals the PE surface multiplied by the PE neighbor distance. It is important to note that the PE distance is related to the communication requirements of the code, more precisely it is proportional to the number of messages generated in order to satisfy each necessary inter-processor boundary exchange. This is a consequence of the slab decomposition and could lead to communication inefficiencies depending on the specific machine topology.

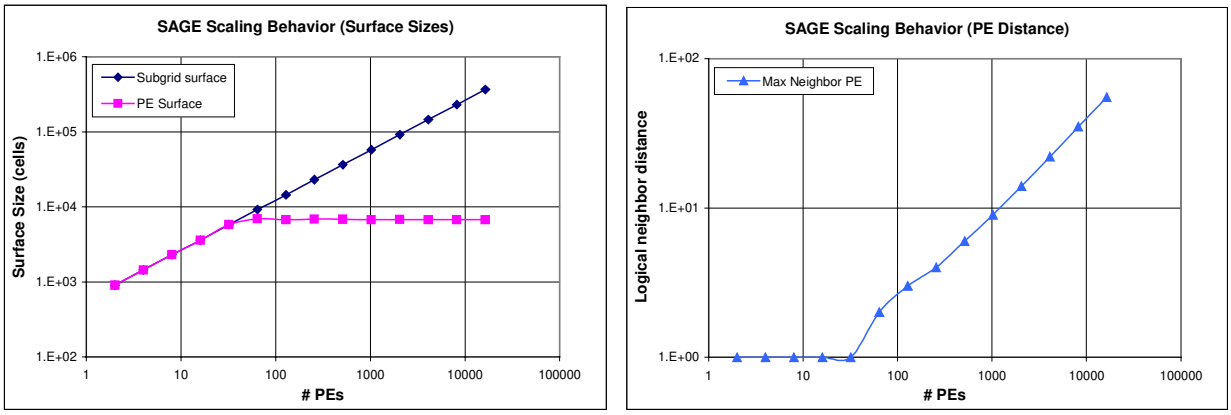

Figure 2: Sage scaling behavior – a) subgrid surface and PE surface sizes, b) PED, the logical neighbor distance.

A further observation related to the communication pattern is that when processors are labeled in a vector-like manner from 0 to P-1 and with p processors per SMP box, out-of-box communication involves no more than a number of pairs of processors equal to the *min (PED,p).* Of course, if PED is larger than p, more than two SMP boxes will be involved in the boundary exchange. As an example, on the ASCI Blue Mountain at Los Alamos, composed of Origin 2000 boxes, given that p=128 and that, from figure 2b) the PED cannot be larger than 100, no more than 2 Origin 2K boxes will communicate for one boundary exchange.

2.3. An iteration *cycle* of SAGE

The processing stages within a cycle typically involve three operations which are repeated a number of times dependent on the time interval utilized for integration of the equations:

- one (or more) gather operations to obtain a copy of the local and remote neighbor data
- computation in each of the local cells
- one (or more) scatter operations to update data on remote processors.

These three operations of SAGE directly relate to the surface-to-volume ratio of the code [2]. The first and the third stage define the surface, related to the amount and pattern of communication, while the second stage represents the volume, related to the amount of computation.

The gather and scatter operations are performed using the token library described in section 2.1. A depiction of the three stages in a cycle of SAGE is shown schematically in Figure 3. In this example, it is assumed that the number of PEs (*P*) is 256, and the number of cells per PE (*E*) is 13,500. A single gather operation (in all dimensions) is depicted, followed by processing, and then a single scatter operation (in all dimensions). The communication is shown only for the processor *n* but in reality all processors perform communication of the same size in the same direction at the same time. In this example it can be seen that the main communication is in the Z dimension dealing with the subgrid surface. The preponderance of communication in the Z-direction is also a consequence of the slab decomposition and is intuitive from Figure 1b). The message sizes in both directions (HI and LO in SAGE terminology) of the three axes is shown in the box on the right side of Figure 3.

In addition to the gather/scatter style operation, a number of other communications take place including several MPI type allreduce communications per cycle. A number of broadcast operations also exist but only during the initialization phase of the code.

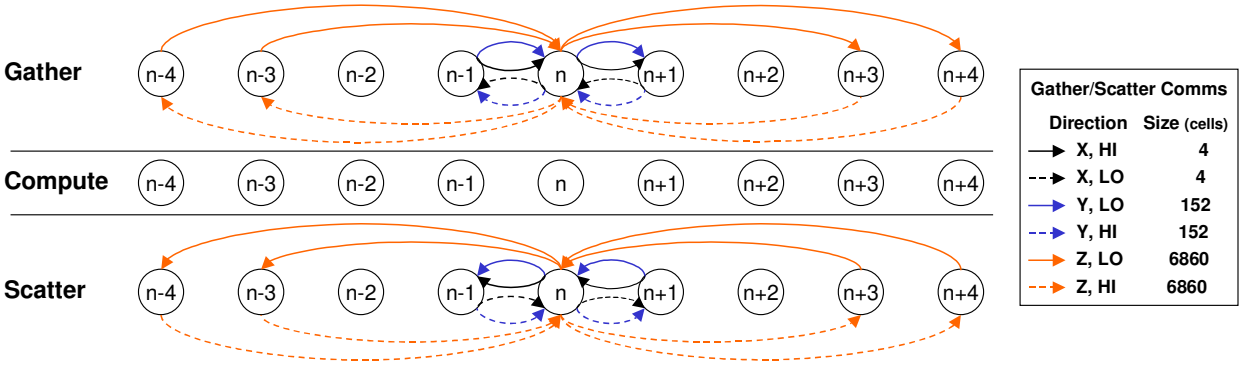

Figure 3: A schematic representation of the communication and computation in a cycle of SAGE consisting of a data gather, processing, and a data scatter.

The frequency of the gather/scatter operations was analyzed using MPI trace data. From this, the number of scatter/gather operations was taken to be 160 real and 17 integer operations per cycle. The PE surface communications represent 20% of the total number of messages but over 95% of the total communication time. In addition, 120 allreduce operations also take place per cycle 4 bytes in size.

3. A Performance Model for SAGE

The communication and computation stages of SAGE are centered around the gather/compute/scatter operations as described in Section 2.3. The runtime for one cycle of the code, given that the three stages are not overlapped, is:

$$T_{cycle}(P,E) = T_{comp}(E) + T_{GScomm}(P,E) + T_{allreduce}(P) + T_{memcon}(P,E) \quad (9)$$

where:

P is the number of PEs
E is the number of cells per PE
T_{comp} is the computation time
$T_{GScomm}(P,E)$ is the gather/scatter communication time
$T_{allreduce}(P)$ is the allreduce communication time
$T_{memcon}(P,E)$ is memory contention that may occur between PEs within an SMP

We describe here briefly only the critical parts of this model:

- computation time - which can be measured from an execution on a single PE for a given E
- gather/scatter communication – this is the time to exchange boundary information among all processors that own that boundary. This is related to PED, the communication distance described in section 2.2, and on the sizes of the messages. The model used for communication takes into account the specifics of the gather/scatter operations within SAGE. A LogGP approach [1] is used.

Full details of the model will be included in the final paper.

Input parameters to the model consist of: the serial run-time of the code, the latency and bandwidth of the MPI communication, characteristics of the memory hierarchy within an SMP box, and characteristics of the network topology (as described below). Two machines have been used to validate this model: an unannounced product from Compaq, and the ASCI Blue Mountain (a cluster of SGI Origin 2000 nodes). [Note to reviewers: the final version of the paper will contain the name of this product.] As an example, the communication parameters for the model for these two architectures are: latency (6μs and 15μs respectively), and bandwidth (293MB/s and 100MB/s respectively). The full details will be included in the final paper.

The impact of PED on communication performance depends on the specific network topology. On the unannounced Compaq product, the maximum contention from an SMP box occurs when all 4 PEs within the box perform out-of-box sends and each receive from out-of-box PEs. This system's topology is a fat-tree using the Quadrics QSNet (Figure 4). This network is able to handle any logical PED without penalty – hence for this particular network there will be no extra overhead due to the physical distance between processors within the PED.

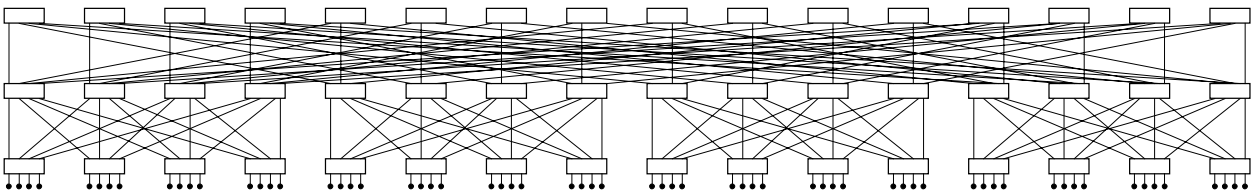

Figure 4: Network topology for a cluster of Compaq SMPs using Quadrics' QSNet Fat-tree network.

Other topologies are not contention free under this communication pattern, for example the Cray T3E, ASCI Red, and ASCI Blue Mountain. Communication involving processors within the PED will be bottlenecked by the lack of physical links between processors that limit the concurrency of messages. For example, with the ASCI Blue Mountain, the minimum number of HiPPi channels that are used to interconnect SMP boxes of 128 PEs is 2, as shown in Figure 5.

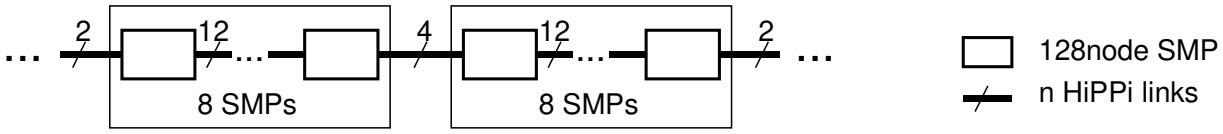

Figure 5: Inter-SMP network on ASCI Blue Mountain

4. Application of the Model

In this section we validate the model proposed in the previous section and apply it to predicting performance of SAGE on future architectures. We also investigate its performance given algorithmic changes that could be implemented in the code.

4.1. Validation and Performance Prediction on Future Architectures

The model presented in Section 3 has been validated against measurements from the unannounced Compaq product (Figure 6) and the ASCI Blue Mountain machine (Figure 7). Predictions show high accuracy – mostly within 10% of the actual measurements.

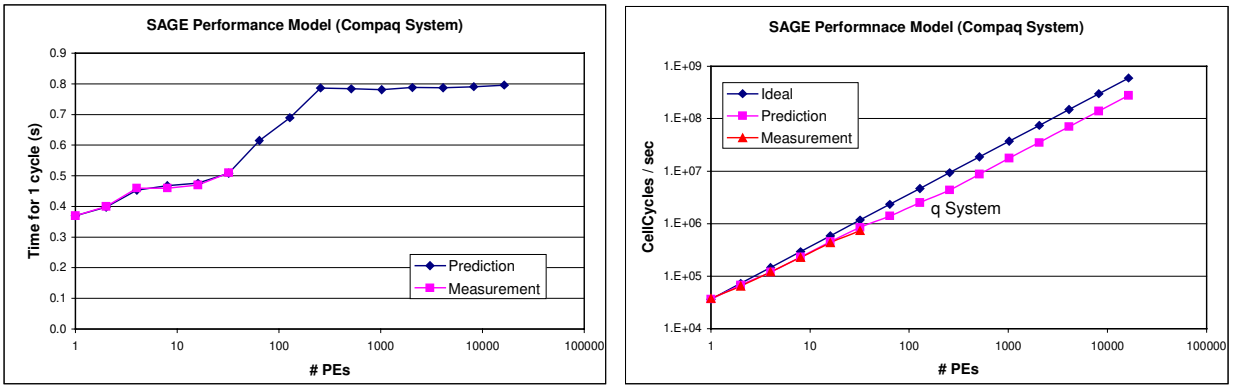

Figure 6: Comparison of performance model predictions against measurements of SAGE on the Compaq system.

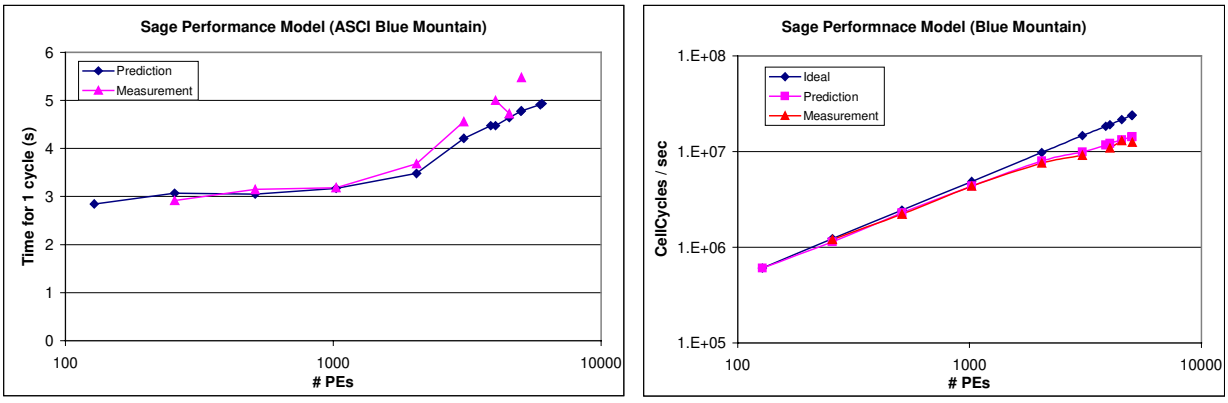

Figure 7: Comparison of performance model predictions against measurements of SAGE on ASCI Blue Mountain

A comparison of the cell-cycles per second on SAGE is shown in Figure 8. This metric is used by SAGE as an indication of performance. It represents the number of cells that can be processed in each wall-clock time unit. In Figure 8 measurements are used for the CRAY T3E, ASCI Red, and ASCI Blue Mountain, whereas we predict the performance of the Compaq system using our model.

The model predicts the performance of the Compaq system to be a factor of 4.5 times greater than that on a CRAY T3E on a comparable number of processors. A system with a peak performance of 30Tflops composed of the Compaq SMP boxes with Quadrics QSNet would be approximately 20 times greater than the performance of SAGE achieved to date on the ASCI Blue Mountain with 6000 O2K processors. By comparison, the ratio of peak speeds is approximately 10.

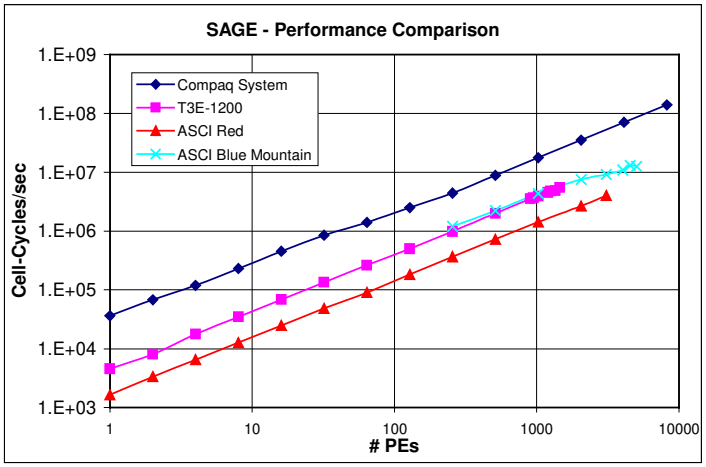

Figure 8: Comparison of the performance of SAGE across several systems.

4.2. Performance Prediction Given Algorithmic Transformations: a New Parallel Data Decomposition

The surface-to-volume ratio is dependent on the grid decomposition. There is a large difference between the use of the slab decomposition (Figure 1) and a "cube" decomposition (Figure 9). Where the slab decomposition results in communications scaling as the 2/3 power of the number of PEs, as shown by equation (3), with a cube decomposition the communication size will remain constant, though the number of PE pairs communicating will be larger. It can be easily shown [see for example 2] that the surface-to-volume ratio (i.e. the communication-to-computation ratio) gets better (i.e. smaller) as the aspect ratio of the subgrids changes towards being perfect cubes, as suggested in Figure 9. Of course, perfect cubic decomposition can only be achieved when the number of processors is a cubic power, as is the decomposition on 8 processors shown in Case 3 of Figure 9.

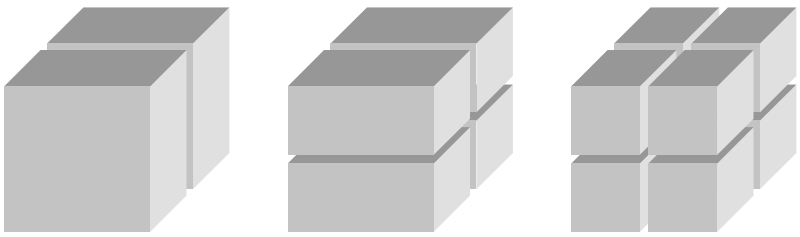

Figure 9: Possible 3-D decomposition configurations for 2, 4 and 8 processors

A comparison between the cube decomposition and the slab decomposition is shown in Figure 10. The total surface of an individual PE is plotted which is proportional to the communication that takes place in each gather (and scatter) operation. The PE distance (PED) is also shown in Figure 10b). The curves for the slab decomposition have already been presented in Figure 2. The PED for the slab decomposition in the X and Y dimensions are always equal to 1. For the cube decomposition PED is always equal to 1 in the X dimension, but varies in the Y and Z dimensions.

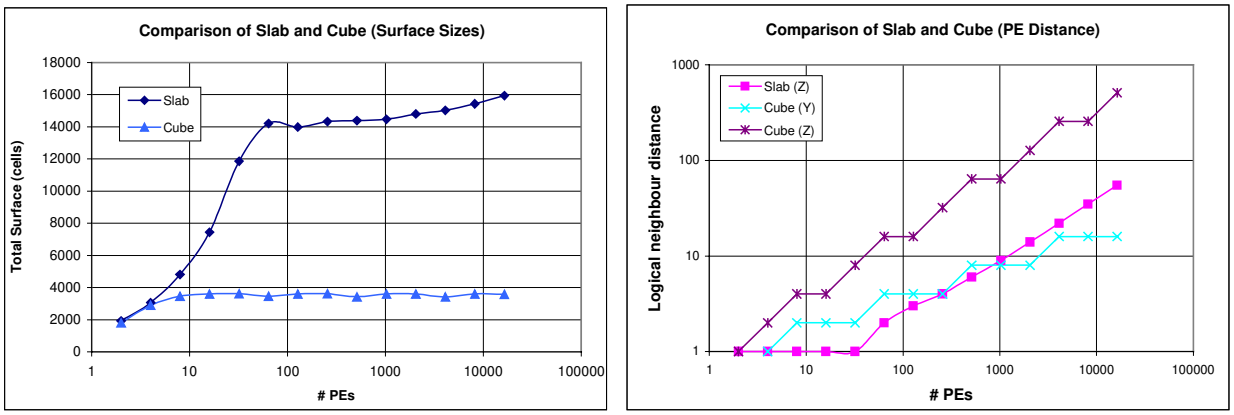

Figure 10: Comparison of Slab and Cube decomposition on PE surface and PED.

The communication size using the cube decomposition is considerably smaller than that for the slab, but the PED is considerably larger. A comparison between the expected performance of SAGE using cube decomposition and the current slab decomposition on the Compaq system, and the ASCI Blue Mountain, is shown in Figure 11. The use of cube reduces communication requirements and hence results in an expected performance improvement of 35% (on the Compaq system), and between 15% and 45% (on the SGI system) compared with the use of slabs.

SAGE could benefit from a cube decomposition if the communication network within the machine is able to handle the large logical PEDs without performance penalty. This is true in the fat-tree topology of the Quadrics network used on the cluster of Compaq SMPs as described in Section 3.

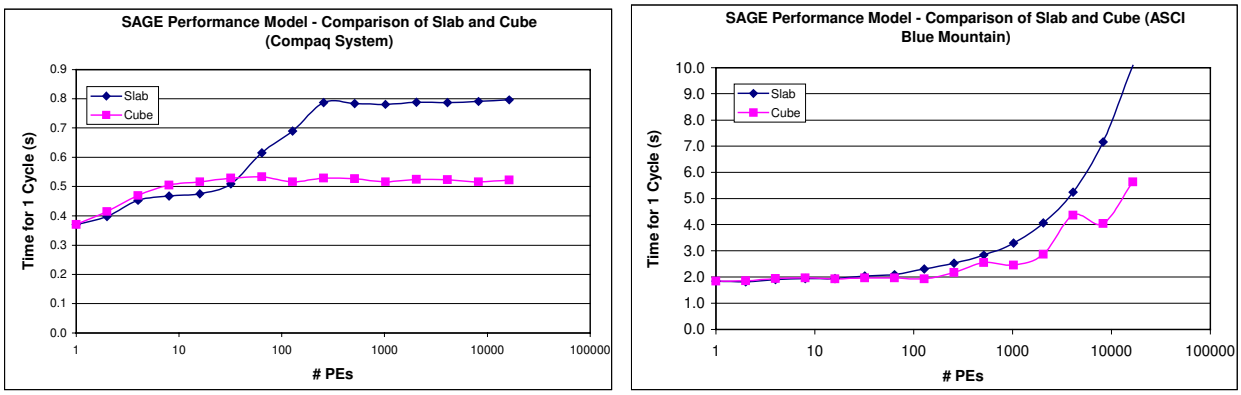

Figure 11: Comparison of slab and cube decompositions on the Compaq system, and the ASCI Blue Mountain.

5. Summary and Conclusions

In this paper we have presented a predictive performance and scalability model for an important application from the ASCI workload. The model takes into account the main computation and communication characteristics of the entire code. The model proposed was validated on two large-scale ASCI architectures, showing very good accuracy. The model was then utilized to predict performance of SAGE on future architectures and when using a better parallel data decomposition.

We believe that performance modeling is the key to building performance engineered applications and architectures. To this end, the work presented in this paper represents one of a very few existing performance models of entire applications. Like our previous performance model of a particle transport application [4], the model incorporates information from various levels of the benchmark hierarchy [3] and is parametric - basic machine performance numbers (latency, MFLOPS rate, bandwidth) and application characteristics (problem size, decomposition method, etc.) serve as input. Such a model adds insight into the performance of current systems, revealing bottlenecks and showing where tuning efforts would be most effective. It also allows prediction of performance on future systems. The latter is important for both application and system architecture design as well as for the procurement of supercomputer architectures

A performance model is meant to be *updated, refined*, and *further validated* as new factors come into play. The work performed in this report is concerned with the analysis of SAGE in absence of grid adaptation. It is foreseen that with additional analysis, the results can be extended to include much of the adaptation process. Also, for the full paper we anticipate that validation of the model on other computers will be presented, including a large-scale IBM RS/6000 SP, the ASCI White architecture.

6. Bibliography

[1] D.E. Culler, J.P. Singh, A Gupta, Parallel Computer Architecture, Morgan Kaufmann, ISBN 1-55860-343-3, 1999.

[2] S. Goedecker, A. Hoisie, Performance Optimization of Numerically Intensive Codes, SIAM Press, ISBN 0-89871-484-2, March 2001.

[3] R. Hockney, M. Berry (Eds). Public International Benchmarks for Parallel Computers. Scientific Programming, Vol. 3, pp. 101-104, 1994.

[4] A. Hoisie. O. Lubeck, H. Wasserman. Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications, Int. J. of High Performance Computing Applications, Winter 200, Vol. 14, No. 4, pp. 330-346

[5] G.R. Nudd, D.J. Kerbyson et.al. PACE: A Toolset for the Performance Prediction of Parallel and Distributed Systems, in the Journal of High Performance Applications, Vol. 14, No. 3, Fall 2000, pp. 228-251.

[6] R. Weaver, Major 3-D Parallel Simulations, http://www.lanl.gov/orgs/cic/cic6/bits/99june_julybits/jjbits4.html, Los Alamos National Laboratory, July 1999.

[7] P. H. Worley. Performance Tuning and Evaluation of a Parallel Community Climate Model, SC99, Portland, Oregon, November 1999.