

Performance Modeling of Unstructured Mesh Particle Transport Computations

Mark M. Mathis and Darren J. Kerbyson
Performance and Architecture Laboratory (PAL),
Los Alamos National Laboratory, CCS-3
PO Box 1663, Los Alamos, NM 87545
{mmathis,djk}@lanl.gov

Abstract

The performance of unstructured mesh applications presents a number of complexities and subtleties that do not arise for dense structured meshes. From a programming point of view, handling an unstructured mesh has an increased complexity to manage the necessary data structures and interactions between mesh-cells. From a performance point of view, there are added difficulties in understanding both the processing time on a single processor and the scaling characteristics. In this work we present a performance model for the calculation of deterministic S_N transport on unstructured meshes. It builds upon earlier work that successfully modeled the same calculation on structured meshes. The model captures the key processing characteristics and is parametric using both the system performance data (latency, bandwidth, processing rate etc.) and application data (mesh size etc.) as input. The model is validated on two clusters (an HP AlphaServer and an Itanium-2 system) showing high accuracy. Importantly it is also shown that a single formulation of the model can be used to predict the performance of two quite different implementations of the same calculation.

1 Introduction

Performance modeling is an important tool that can be used by a performance analyst to provide insight into the achievable performance of a system and/or an application. It is only through knowledge of the workload the system is to be used for that a meaningful performance comparison can be made. It has been recognized that performance modeling can be used throughout the life-cycle of a system, or of an application, from first design through to maintenance [4] including procurement and system installation.

Recent work at Los Alamos National Laboratory (LANL) has demonstrated the use of performance modeling in many situations, for instance: 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 applications prior to implementation [7]; and to verify the performance of the 20-Tflop ASCI Q system during its installation [6], which ultimately lead to system optimizations resulting with a factor of 2 performance improvement [13]. Models have also been used to compare large-scale system performance including a comparison of several of the highest peak-rated terascale systems such as the Earth Simulator and ASCI Q [5].

In this work we present the development and use of a model that accurately captures the performance characteristics of deterministic S_N transport on unstructured meshes. This calculation solves the Boltzmann equation. It is an important application that uses a high percentage of the total cycles across the ASCI (Accelerated Strategic Computing Initiative) machines.

Unstructured meshes have several benefits over the use of structured meshes in terms of the calculation undertaken, but also have significant extra overhead in terms of performance. Several important performance factors that can reduce the overall calculation efficiency of this type of computation on large-scale parallel systems are analyzed in this paper.

Efforts devoted to the performance analysis of S_N transport date back many years. Research has included the development of performance models as a function of problem mesh and machine size [8]. More detailed performance models have been developed that also include detailed communication, and SMP cluster characteristics [1, 2]. Other work considers 3D partitioning, but still using an underlying structured mesh [9].

The key contribution of this paper is the development of a general analytical performance model of S_N transport computations on unstructured meshes. The model is shown to be applicable across different implementations including an experimental code under development at LANL called Tycho [11], and a production code called UMT2K [15] from Lawrence Livermore National Laboratory (LLNL). Tycho

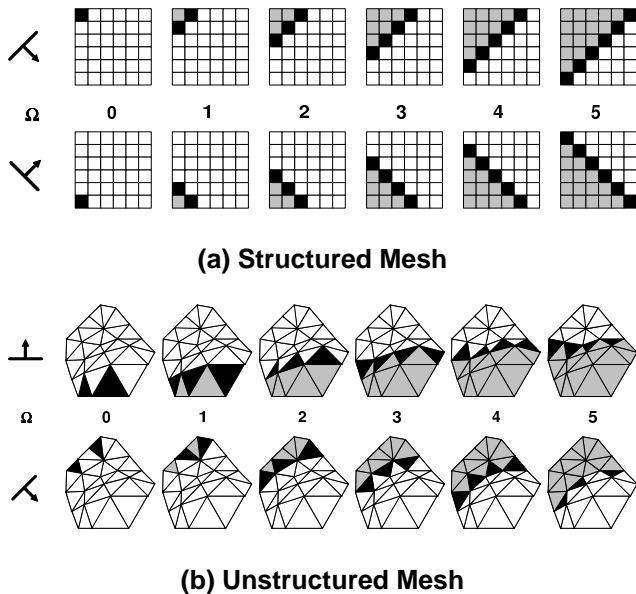


Figure 1. Example sweep processing in two dimensions.

is written in C++, and UMT2K is written in a combination of Fortran and C. UMT2K was part of the benchmark suite used in the recent procurement of ASCI purple. Other codes are also being developed for S_N transport computations e.g. [14].

The analytical model developed here is shown to have reasonable accuracy through a validation process using a 64 node HP AlphaServer system and a 32-node Itanium-2 cluster. We concentrate on the development and the validation of the model in this work. However, the accuracy of the model is such that it may be used to explore many performance scenarios, for instance to examine the achievable performance that could be obtained on hypothetical future architectures and also to indicate the size of mesh that could be processed in a given time.

The paper is organized as follows. In Section 2, the S_N transport calculation is detailed and comparisons between its operation on structured and un-structured meshes are made. In Section 3 the key characteristics of the processing are described which are used in the development of the performance model in Section 4. The models are validated in Section 5 on a number of different input unstructured meshes that represent different physical geometries for both Tycho and UMT2K.

2 Overview of S_N Transport Algorithms

The algorithms employed in deterministic S_N transport (discrete ordinate) computations fall in a class generically named wavefront techniques. In a nutshell they utilize an

iterative approach using a method of “sweeping” [1]. Each spatial cell in a mesh is processed in a specified order for each direction in the discrete ordinates set. The wavefronts (or sweeps) are software pipelined in each of the processing directions. Wavefront algorithms exhibit several interesting performance characteristics, related to the pipelined nature of the wavefront dynamics. These include a pipeline delay across processors for a sweep, and a repetition rate of both computation and communication in the direction of the sweep.

In the case of a structured mesh a high processing efficiency can be achieved as all active processors perform the same amount of work, and communicate the same sized boundary data [1]. However, the efficiency when using unstructured meshes is lower due to a possible imbalance of work across processors as the wavefronts progress. The processing flow of the calculation is described below for both structured or unstructured meshes.

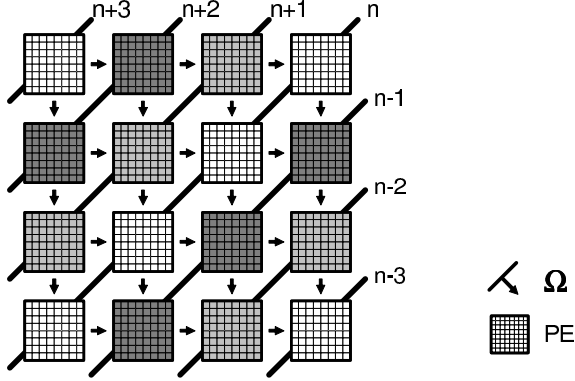
2.1 The method of sweeping

Structured meshes

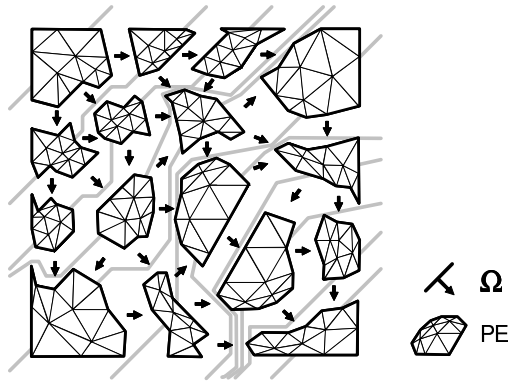
In three-dimensions, each sweep direction can be considered to originate in one of the 8 corners (“octants”) of the spatial domain. For a typical discrete ordinate order of 6, i.e. S_6 , the total number of sweep directions is 48 or 6 per octant. For sweep directions within each octant, the ordering of cell processing is identical. Figure 1 (a) shows the first six steps of two separate sweeps at different angles, Ω , originating from different quadrants for a two-dimensional spatial domain. The edge of the sweep corresponds to a wavefront and is shown as black. Cells on the sweep edge require the grey cells to have been processed in previous steps. The same operation can take place in three dimensions resulting in a wavefront surface. The wavefront propagates across the spatial domain at a constant calculation velocity since the time needed to process a cell is constant. This processing algorithm as developed in [8], uses direct indexing of the spatial mesh as the cell processing order is deterministic for each sweep direction.

Unstructured meshes

An example two-dimensional unstructured mesh consisting of triangles is depicted in Figure 1 (b). Two sweep directions are again used to illustrate the processing over a total of six steps. As before, the cells being processed in the current step are shown as black and require the previously calculated grey cells. The processing order of the cells is dependent on the direction of the sweep, and is not the same for sweeps that originate in the same octant (as was the case for structured meshes). The incoming data to a cell are determined by the mesh geometry, and it is apparent that



(a) Structured Mesh



(b) Unstructured Mesh

Figure 2. The pipeline effect of processing sweeps in parallel.

the propagation speed of the wavefronts also varies with direction. The same situation occurs with three-dimensional geometry, only with the mesh being composed of tetrahedrons, pyramids, hexahedrals and prisms.

2.2 Sweeping in Parallel

Parallel wavefront computations exhibit a balance between processor efficiency and communication cost [1]. Faster wavefronts, generated by a data decomposition leading to small subgrid sizes per processor, introduce higher communication costs but result in high processor utilization. The opposite holds true for slower moving sweeps due to larger subgrid sizes. In order to optimize wavefront dynamics, S_N transport applications typically utilize blocking of the spatial subgrid and/or blocking of the wavefront angle set.

Important performance considerations in parallel wavefront applications, which need to be captured in a model, are as follows:

pipeline effects a processor is inactive until the edge of a sweep enters cells in that particular processor. However, multiple sweeps are active at any given time in the processor array. Overlap exists between computation and communication within each sweep, and across the active sweeps.

communication costs for boundary data transfer.

load balancing of the number of cells processed on each processing element (PE) in a step. This applies to wavefronts on unstructured meshes only as an equal number of cells are processed in each step on each PE for a structured mesh.

In order to analyze these effects, the processing that takes place on both structured and unstructured meshes is illustrated below.

Structured meshes

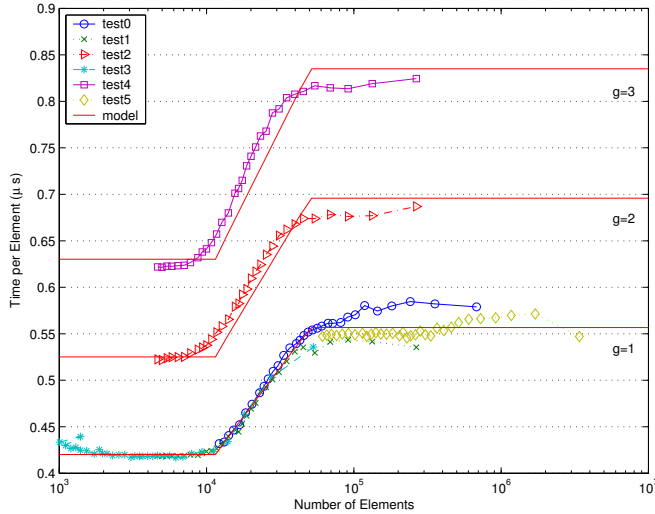
In codes such as Sweep3D [1] which perform an S_N transport computation on a structured 3-D mesh, the mesh is mapped onto a 2-D processor array such that each processor has a column of data which can be blocked during computation in its third dimension. The processing is effectively synchronized after the first sweep has moved across the processor array resulting in all processors being active. The processing involved in each sweep is dependent on the block size (a known constant). Thus one diagonal of processors will be processing one sweep (or one block in the third mesh dimension) while the previous diagonal is processing the next sweep and so on (Figure 2 (a)). The direction of sweep travel is again indicated by Ω and inter-processor communications shown by arrows. It has been shown that the cost of performing this calculation on a structured mesh conforms to a pipeline model [1]:

$$T_{Total} = (P_x + P_y - 1)(T_{CPU} + 2T_{msg}) + (N_{sweep} - 1)(T_{CPU} + 4T_{msg}) \quad (1)$$

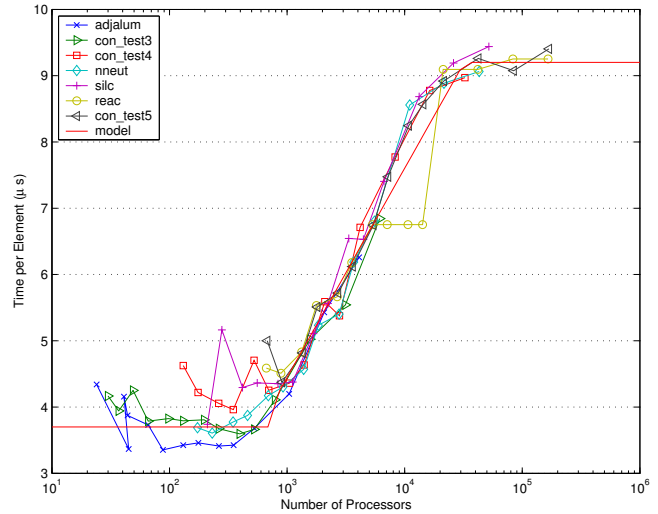
where P_x and P_y are dimensions of the processor grid, N_{sweep} is the number of sweeps, T_{CPU} and T_{MSG} are the times to process a cell block, and to communicate a message respectively. The first part of this equation corresponds to the length of the pipeline and the second part is the number of repetitions once the pipeline is filled.

Unstructured meshes

The processing on an unstructured mesh follows the same dependency rules as above, but the mesh partitioning is typically done in all 3 dimensions. An example 2-D unstructured mesh partitioned in both dimensions is shown in Figure 2 (b). The communication between processors is shown



(a) UMT2K



(b) Tycho

Figure 3. Processing time per cell for different meshes and energy groups.

by arrows, and a simplified propagation of the sweep in the indicated direction is shown by the grey lines. A sweep for each direction (angle) is required. The unit of processing work can be considered as a single cell-angle pair.

The sweep processing on the unstructured mesh can also be blocked - up to a set maximum number of cell-angle pairs can be processed per step. This blocking is analogous to the blocking in the third dimension of in the case of the structured mesh. However, it can result in processor inefficiency down the pipeline - there is no guarantee that the same number of cell-angle pairs will be processed by all processors in a step resulting in possible processor idleness.

3 Processing Characteristics of S_N transport on unstructured meshes

The significant processing characteristics for S_N transport on unstructured meshes are: mesh partitioning, pipeline processing, processor utilization, and strong scaling. An understanding of these factors is required in order to formulate a performance model. There are also differences between the Tycho and UMT2K implementations considered in this work.

3.1 Mesh partitioning

The partitioning is not done directly by the applications, rather the Metis partitioner [3] is used. This mesh partitioner aims to produce equally sized partitions (equal number of cells) while minimizing boundaries. In general such an optimal partitioning of the mesh keeps the work across

processors constant and minimizes the communication cost. However, due to the pipeline processing and load-balancing characteristics in Tycho and UMT2K as well as S_N transport in general, this partitioning may not be optimal. For a 3-D partitioning produced by Metis, the number of cells per partition can be taken to be $E_p = N/P$ where N is the total number of cells in the mesh, and P is the number of PEs (which is equal to the number of partitions). Each 3-D partition would ideally have six nearest neighbors (equal to that of a 3-D partitioned structured mesh) each with a boundary size of $E_p^{2/3}$ cells.

3.2 Pipeline processing

The first cell-angle pairs processed are those that lie on the boundary of the overall spatial mesh - those cells that have no inflows in the direction of the sweep. This corresponds to nearly all boundary elements. The sweeps thus generally start from the surface of the mesh and work their way to the center before propagating out to the opposite side of the mesh.

The dynamics of the pipeline is determined by the pipeline length and by the amount of computation done on each mesh partition. The pipeline length is determined by the number of stages in the propagation of the sweep from one side of the mesh to another. In 2-D example shown in Figure 2 (b) the number of grey lines represent the number of stages. In general, given an ideal 3-D mesh partitioning, the pipeline length is given by:

$$P_L = (P_x - 1) + (P_y - 1) + (P_z - 1) \quad (2)$$

Table 1. Hardware parameters for both validation systems.

	Itanium-2 1GHz		AlphaServer ES40 833MHz
$T_e(E_p, G)(\mu s)$	$\begin{cases} 0.139 * (3 + G) & E_p \geq 50K \\ (0.015Ln(E_p) - 0.094) * (3 + G) & 12K < E_p < 50K \\ 0.105 * (3 + G) & E_p \leq 12K \end{cases}$		$\begin{cases} 9.2 & E_p \geq 16K \\ 1.8Ln(E_p) - 8.4 & 800 < E_p < 16K \\ 3.7 & E_p \leq 800 \end{cases}$
$L_c(S)(\mu s)$	$\begin{cases} 6.48 & S < 64bytes \\ 8.21 & 64 \leq S \leq 256 \\ 17.1 & S > 512 \end{cases}$		$\begin{cases} 9.28 & S < 64bytes \\ 9.00 & 64 \leq S \leq 256 \\ 21.4 & S > 512 \end{cases}$
$1/B_c(S, D)(ns)$	$\begin{cases} 0.0 & S < 64bytes \\ 25.5 & 64 \leq S \leq 512 \\ 13.7 & S > 512 \end{cases}$		$\begin{cases} 0.0 & S < 64bytes \\ 22.7 & 64 \leq S \leq 512 \\ 11.2 & S > 512 \end{cases}$

where P_x , P_y and P_z are the number of PEs in each of the three dimensions respectively and $P = P_x \cdot P_y \cdot P_z$. The total work done, or the total number of cell-angle pairs processed, in each mesh partition in an iteration of the S_N transport computation is equal to:

$$W_p = E_p * N_\Omega \quad (3)$$

where N_Ω is number of sweep directions. For a specific mesh, the pipeline length, P'_L can be obtained by inspection of the mesh after the partitioning has been performed. P'_L is equal to the maximum number of PEs traversed in any sweep direction and will in general be greater than P_L . The total amount of work done per partition remains as above. Both applications allow sweeps in multiple directions to be processed in parallel (using a multithreaded sweep kernel).

3.3 Processor utilization

Each step of both Tycho and UMT2K consists of three stages: process the cell-angle pairs which have available their in-flow boundary data, send boundary data that is produced, and receive boundary data from neighbor sub-grids. The processing situation is complicated by the processing dependence between upstream and downstream cells in the sweep directions. This dependence may lead to downstream PEs waiting for the upstream PEs to send the necessary boundary information. In general there will be a degree of inefficiency in this operation and processors will be starved of work waiting for the results from other PEs. It is interesting to note that for the case of structured meshes, the work on each PE is equal throughout and thus the processors are fully utilized once the pipeline is filled. To quantify this inefficiency, the Parallel Computational Efficiency, PCE [11], is used:

$$PCE = \frac{W_p}{\sum_{S=1}^{N_{steps}} \max_P(\|work(P, S)\|)} \quad (4)$$

where $work(P, S)$ is the number of cell-angle pairs processed on processor P in step S , and W_p is the total number

of cell-angle pairs processed on each processor in an iteration. PCE represents the fraction of the maximum number of cells that are processed in all steps in an iteration. When $PCE = 1$ the efficiency is 100% - this can only occur on a small processor run (typically < 9 PEs). The lower the value of PCE , the greater the inefficiency.

A value for PCE can be obtained for a specific mesh after its partitioning and before the S_N transport calculation. The number of steps required to perform the total number of cell-angle pairs per PE (excluding the pipeline effect) is given by:

$$\frac{W_p}{(MCPS * PCE)} \quad (5)$$

where $MCPS$ is the maximum number of cells that can be processed in a step. In the general case, i.e. without pre-inspection of the mesh, a value for PCE has to be assumed - possibly based on experience from prior meshes. This assumption can be inaccurate reflecting the tradeoff between generality and accuracy that is present in performance modeling work.

3.4 Strong scaling

Typical S_N transport calculations on unstructured meshes are executed in a strong scaling mode i.e. parallelism is used to solve the same problem but in reduced time. The input mesh geometry does not change and thus partitions become smaller on larger processor counts. Strong scaling causes a change in the actual use of the memory hierarchy when increasing the number of processors used and hence its performance has to be carefully considered. For instance when a mesh partition becomes small enough to fit in cache the performance will be better than if main memory has to be accessed.

Figure 3 shows the computation time per cell for a number of different meshes and partition sizes for (a) UMT2k on a 1GHz Itanium-2 processor and (b) Tycho on an 833MHz Alpha EV68 processor. The Itanium-2 used has a 3MB L3

Table 2. Test cases for the validation of the UMT2K performance model

Case	Mesh	#Cells	Description	Error (%)
0	MMesh	680,400	Medium mesh, 4950 cells/layer, 3 layers, 1 energy group	12.53
1	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 1 energy group	8.33
2	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 2 energy groups	8.98
3	SMesh	53,136	Small mesh, 398 cells/layer, 3 layers, 1 energy group	11.41
4	SMesh	265,680	Small mesh, 398 cells/layer, 15 layers, 3 energy groups	8.87
5	MMesh	3,402,000	Large mesh, 4950 cells/layer, 15 layers, 1 energy group	9.06

Table 3. Test cases for the validation of the Tycho performance models

Case	Mesh	#Cells	Description	Error (%)
6	Nneut	43,012	Neutron well-logging tool and surrounding media	13.48
7	Silc	51,963	Computer Chip and packaging for radiation shielding	12.07
8	Reac	165,530	Reactor pressure vessel and surrounding cavity structures	7.44
9	Con_test5	168,356	Cube divided into approximately equal-sized elements	8.07

cache, and the Alpha EV68 has an 8MB L2 cache. There are three regions evident: when the partition does not fit into cache (right hand plateau of the curves), when the mesh fits into cache (left hand plateau), and when partial cache reuse occurs (middle region). Note that there is also a further parameter in the case of UMT2K which is the number of energy groups that are processed per cell. This can vary and increases the processing time per cell by a multiplicative factor as shown in Figure 3. The number of energy groups in Tycho is fixed at one.

There is some variation in performance between the meshes in this analysis due to the different memory access patterns and hence the actual cache reuse. It can be seen that a good approximation to the time taken to process a cell can be encapsulated in a piece-wise linear model. In general however, we are interested in large meshes - those that unfortunately will not exhibit cache re-use, and also those that cannot be executed on small processor counts due to limitations in the size of the actual memory per processor.

4 Analytical Performance Model

In the performance model of S_N transport computation we assume that the three stages that constitute a processing step are distinct and do not overlap - those of computation, blocking sends and blocking receives for the boundary communications. Further the model also takes a maximum of the amount of work performed in each step over all processors, as well as the maximum of the communications performed to/from any one node. This assumption will tend to give an over prediction of the iteration time. The model for

the time for one iteration is formulated as:

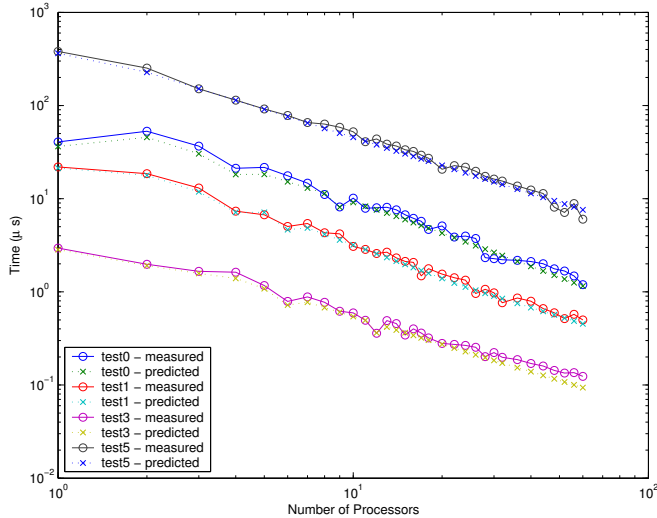
$$T_i = \left(\sum_{S=1}^{N_{steps}} \max_P (Work(P, S)) \right) \cdot T_e\left(\frac{N}{P}, G\right) + \sum_{S=1}^{N_{steps}} \max_P \left(\sum_{C=1}^{\|N_c(S, P)\|} T_c(N_c(S, P, C), N_s(S, P, C)) \right) \quad (6)$$

where the first term represents computation and the second term communication. There are N_{steps} steps per iteration with $Work(P, S)$ cell-angle pairs being processed on processor P in step S . The time to process x cell-angle pairs for G energy groups is given by $T_e(x, G)$. $N_c(S, P, C)$ is the destination PE for communication C in step S on processor P and $N_s(S, P, C)$ is the size of the same communication. $\|N_c(S, P)\|$ is the total number of communications originating from processor P in step S . The time to communicate a message of size y bytes to processor x is given by $T_c(x, y)$.

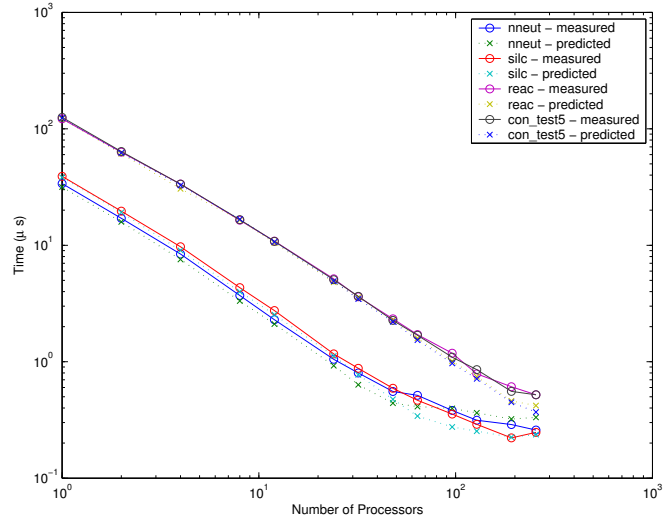
The differences between the two S_N transport implementations become apparent in the way in which the parameters to this model are specified. The number of steps in Tycho is given by:

$$N_{steps} = \left(\frac{E_P * N_\Omega}{MCPS * PCE} \right) + (P_x - 1) + (P_y - 1) + (P_z - 1) \quad (7)$$

where the first part of the equation represents the number of steps required to process a sub-grid given a value of PCE , and the second part is the length of the pipeline in 3-D. $MCPS$ is the MaxCellPerStep input blocking parameter



(a) UMT2K



(b) Tycho

Figure 4. Model Validations for (a) UMT2K and (b) Tycho.

to Tycho. The number of steps in UMT2K is given by:

$$N_{steps} = N_{\Omega} * noutmx \quad (8)$$

where N_{Ω} is the size of the discrete ordinates set, and $noutmx$ is the number of outer iterations. This is independent of the pipeline length since UMT2K does not implement a strict sweep operation - it uses old sub-grid boundary data from the previous time-step which also has the effect of making $PCE = 1$. It can be considered as a simpler S_N transport implementation than Tycho. It does not require the same complexity in the handling of the boundary data flow which is reflected in the difference in the time taken to process a single element shown in Figure 3.

The communications per step in both Tycho and UMT2K are approximated to be similar to that of a partitioned structured 3-D mesh and remain constant throughout. This will in general under predict the actual communication time. A structured grid has 6 local neighbors resulting from a 3-D partitioning of a 3-D spatial grid. The size of the boundary on each boundary surface is $E_p^{2/3}$.

The parameters of $T_e()$, and $T_{comm}()$ are specific to a particular system and are measured. A two-parameter, piece-wise linear model for the communication is assumed which uses the Latency (L_c) and Bandwidth (B_c) of the network communication.

$$T_{comm}(S) = L_c(S, D) + S \cdot \frac{1}{B_c(S, D)} \quad (9)$$

where L_c is the communication latency, B_c is the communication bandwidth, and S is the message size.

The communication time is subject to contention in the communication network. Our experience on using UMT2K and Tycho as well as other codes on clusters of SMPs, is that the main contention occurs on the number of out-of-node communications that occur simultaneously. For example with the fat-tree network of the Quadrics network [12], the number of communications that collide in higher levels of the fat-tree is low due to dynamic routing. The contention is taken into account by a multiplicative constant on the communication time, T_{comm} , which represents the number of simultaneous out-of-node communications.

5 Performance Model Validation

The performance model described in Section 4 is validated below on a 32 node Itanium-2 cluster in the case of UMT2K and a 64 node AlphaServer ES40 Cluster in the case of Tycho. The Itanium-2 cluster consists of 2 processors per node running at 1GHz each with a 256K L2 cache, a 3MB L2 cache, and 2GB main memory. The AlphaServer cluster consists of 4 processors per node running at 833MHz each with an 8MB L2 cache and 2GB main memory. The nodes in both the clusters are interconnected using the Quadrics QSnet-I high speed network with Elan3 switching technology. The performance characteristics of these systems are listed in Table 1.

Several unstructured meshes are used in the validation process. For UMT2K, the input meshes consist of a 2-D mesh of triangles which are projected into the third dimension to form multiple layers of cells. For Tycho, the input meshes consist of a 3-D mesh of tetrahedrals. Six mesh con-

figurations are considered for UMT2K which consist of two different meshes which are projected by different amounts in the third dimension, and also differ in the number of energy groups processed per cell. Four different meshes are considered for Tycho. The tests are chosen to represent a range of partition sizes and are listed in Tables 2 and 3.

Measurements and model predictions for four of the test cases on UMT2K are shown in Figure 4 (a), and for the four test cases on Tycho are shown in Figure 4 (b). A summary of the difference between the model predictions and the measurements is listed in Tables 2 and 3. It can be seen that the model predicts each case with high accuracy with a typical error of 11% across all the test cases.

6 Summary

In this work we have presented a predictive analytical performance and scalability model for S_N transport computations on unstructured meshes. These calculations are relevant to the ASCI workload and are representative of a high proportion of cycles used across all ASCI systems. Thus modeling and understanding their performance is important not only on current systems, but also looking ahead to possible larger scale systems in the future.

The performance model has been shown to be accurate with a typical error of 11% across a range of configurations in terms of processor count, mesh geometry, and systems utilized. Further, the model is shown to be applicable to two quite different implementations of these types of computations. The implementations differ in their input meshes, and in the actual type of S_N transport calculation performed. The differences are handled by changing the parameter inputs to the analytical performance model.

We believe performance modeling is key to building performance engineered applications and architectures. This work is one of few performance models that exist for entire applications. It follows on from our work on structured particle transport modeling [1], adaptive mesh refinement modeling [7], and Monte-Carlo simulation [10].

7 Acknowledgements

We would like to thank Shawn Pautz for many informative discussions on the implementation details of Tycho. This work was supported in part by LDRD 2001609DR “Performance Analysis and Modeling of Extreme-Scale Parallel Architectures”. Mark Mathis is currently a PhD student at Texas A&M University under the guidance of Nancy Amato and is supported in part by a Department of Energy High Performance Computer Science Fellowship. Los Alamos National Laboratory is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36.

References

- [1] A. Hoisie, O. Lubeck, and H. Wasserman. Performance and Scalability Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications. *Int. J. of High Performance Computing Applications*, 14(4):330–346, 2000.
- [2] A. Hoisie, O. Lubeck, H. Wasserman, F. Petrini, and H. Alme. A general predictive performance model for wavefront algorithms on clusters of smps. In *Proceedings of ICPP 2000*, pages 20–25, Toronto, Canada, August 2000.
- [3] G. Karypis and V. Kumar. METIS 4.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical report, Department of Computer Science, University of Minnesota, 1998.
- [4] D. Kerbyson, A. Hoisie, and H. Wasserman. Modeling the Performance of Large-Scale Systems. *IEEE Proceedings (Software)*, 150(4):214–221, 2003.
- [5] D. Kerbyson, A. Hoisie, and H. Wasserman. A Performance Comparison between the Earth Simulator and other Tera-scale Systems on a Characteristic ASCI Workload. *To appear in Concurrence and Computation, Practice and Experience*, 2004.
- [6] D. Kerbyson, A. Hoisie, and H. Wasserman. Use of Predictive Performance Modeling During Large-Scale System Installation. *To appear in Parallel Processing Letters*, 2004.
- [7] D. J. Kerbyson, H. Alme, A. Hoisie, F. Petrini, H. Wasserman, and M. Gittings. Predictive Performance and Scalability Modeling of a Large-scale Application. In *Supercomputing*, Denver, Nov. 2001.
- [8] K. Koch, R. Baker, and R. Alcouffe. A Parallel Algorithm for 3D S_N Transport Sweeps. Technical Report LA-CP-92-406, Los Alamos National Laboratory, 1992.
- [9] M. M. Mathis, N. M. Amato, and M. L. Adams. A general performance model for parallel sweeps on orthogonal grids for particle transport calculations. In *Proc. ACM Int. Conf. Supercomputing (ICS)*, pages 255–263, 2000.
- [10] M. M. Mathis, D. J. Kerbyson, and A. Hoisie. A performance model of non-deterministic particle transport on large-scale systems. In *Proc. Int. Conf. on Computational Science (ICCS)*, Melbourne, Australia, June 2003.
- [11] S. D. Pautz. An Algorithm for Parallel S_N Sweeps on Unstructured Meshes. *J. Nuclear Science and Engineering*, 140:111–136, 2002.
- [12] F. Petrini, W. C. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network: High-Performance Clustering Technology. *IEEE Micro*, 22(1):46–57, 2002.
- [13] F. Petrini, D. Kerbyson, and S. Pakin. The case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. In *Supercomputing*, Phoenix, Nov. 2003.
- [14] S. Plimpton, B. Hendrickson, S. Burns, and W. McLendon. Parallel Algorithms for Radiation Transport on Unstructured Grids. In *Supercomputing*, Dallas, November 2000.
- [15] *The UMT2K (UMT 1.2) README File*. On the Web, <http://www.llnl.gov/asci/purple/benchmarks/limited/umt/umt1.2.readme.html>, November 2003.