

# Scalability Analysis of Multidimensional Wavefront Algorithms on Large-Scale SMP Clusters

Adolfy Hoisie, Olaf Lubeck, and  
Harvey Wasserman  
<hoisie, oml, hjw> @lanl.gov

Scientific Computing Group  
Los Alamos National Laboratory  
Los Alamos, NM 87545

**Abstract.** *We develop a model for the parallel performance of algorithms that consist of concurrent, two-dimensional wavefronts implemented in a message passing environment. The model combines the separate contributions of computation and communication wavefronts. We validate the model on three supercomputer systems, with up to 500 processors, using data from an ASCI deterministic particle transport application, although the model is general to any wavefront algorithm implemented on a 2-D processor domain. We also use the model to make estimates of performance and scalability of wavefront algorithms on 100-TFLOPS computer systems expected to be in existence within the next decade. Our model shows that on a 1-billion-cell problem, single-node computation speed (not inter-processor communication performance, as is widely believed) is the bottleneck. Finally, we present preliminary considerations that reveal the additional complexity associated with modeling wavefront algorithms on reduced-connectivity network topologies, such as clusters of SMPs.*

## 1. Introduction

Wavefront techniques are used to enable parallelism in algorithms that have recurrences by breaking the computation into segments and pipelining the segments through multiple processors [1]. First described as “hyperplane” methods [2], wavefront methods now find application in several important areas including particle physics [3], parallel iterative solvers [4], and parallel solution of triangular systems of linear equations [5-7].

Wavefront computations present interesting implementation and performance modeling challenges on distributed memory machines because they exhibit a subtle balance between processor utilization and communication cost. Optimal task granularity is a function of machine parameters such as raw computational speed, and inter-processor communication latency and bandwidth. Although it is simple to model the computation-only portion of a single wavefront, it is considerably more complicated to model multiple, simultaneous wavefronts, due to potential overlap of computation and communication and/or

overlap of different communication or computation operations individually. Moreover, specific message-passing synchronization methods impose constraints that can further limit the available parallelism in the algorithm. A realistic scalability analysis must take into consideration these constraints.

Much of the previous parallel performance modeling of software-pipelined applications has involved algorithms with one-dimensional recurrences and/or one-dimensional processor decompositions [5-7]. A key contribution of this paper is the development of an analytic performance model of wavefront algorithms that have recurrences in multiple dimensions and that have been partitioned and pipelined on multidimensional processor grids.

Our vehicle for these studies is a “compact application” called SWEEP3D, a time-independent, Cartesian-grid, single-group, “discrete ordinates” deterministic particle transport code taken from the DOE Accelerated Strategic Computing Initiative (ASCI) workload. SWEEP3D represents the core of a widely utilized method of solving the Boltzmann transport equation. Estimates are that deterministic particle transport accounts for 50-80% of the execution time of many realistic simulations on current DOE systems. This percentage may expand on future 100-TFLOPS systems. Thus, an equally important contribution of this work is the use of our model to explore SWEEP3D scalability and to predict performance of SWEEP3D on future-generation systems.

Efforts devoted to improving performance of discrete ordinates particle transport codes date back many years and have extended recently to massively parallel systems [8-12]. Research has included models of performance as a function of problem and machine size, as well as other characteristics of both the simulation and the computer system under study. For example, Koch, Baker, and Alcouffe [3] developed a parallel efficiency formula that considered computation only, while Baker and Alcouffe [9] developed a model specific to CRAY T3D put/get communication. These previous models had limiting assumptions about the computation and/or target machines.

In this work, we model parallel discrete ordinates transport and account for both computation and communi-

cation. We validate the model on several architectures within the realistic limits of all parameters appearing in the model. Although our validation is carried out on systems employing full network connectivity between nodes, we conclude by presenting preliminary considerations that reveal the additional complexity associated with modeling wavefront algorithms on reduced-connectivity network topologies.

## 2. Description of Discrete Ordinates Transport

Although much more complete treatments of discrete ordinates neutron transport have appeared elsewhere [12-15], we include a brief explanation here to make clear the origin of the wavefront process in SWEEP3D. The basis for neutron transport simulation is the time-independent, multigroup, inhomogeneous Boltzmann transport equation, which is formulated as

$$\begin{aligned} \nabla \cdot \Omega \Psi(\mathbf{r}, E, \Omega) + \int \int \sigma(\mathbf{r}, E) \Psi(\mathbf{r}, E, \Omega) = \\ \int \int dE' d'\Omega' (\mathbf{r}, E' \rightarrow E, \Omega, \Omega') \Psi(\mathbf{r}, E', \Omega') + \\ (1/4\pi) \int \int dE' d\Omega' \chi(\mathbf{r}, E' \rightarrow E) \nu \sigma(\mathbf{r}, E') \Psi(\mathbf{r}, E', \Omega') + \\ Q(\mathbf{r}, E, \Omega). \end{aligned}$$

The unknown quantity is  $\Psi$ , the flux of particles at the spatial point  $\mathbf{r}$ , with energy  $E$ , traveling in direction  $\Omega$ . Numerical solution involves discretization of the multi-dimensional phase space defined by  $\mathbf{r}$ ,  $\Omega$ , and  $E$ . In the discrete ordinates approximation (also referred to as the  $S_N$  method), the angular-direction  $\Omega$  is discretized into a set of quadrature points. The discretization is completed by differencing the spatial domain of the problem on to a grid of cells.

The solution method involves an iterative procedure called a “source iteration,” the most time-consuming portion of which is a transport sweep through the entire grid-angle space in the direction of particle travel [13]. In Cartesian geometries, each octant of angles has a different sweep direction through the mesh, and all angles in a given octant sweep the same way.

For a given discrete angle, boundary conditions and the spatial differencing allow the sweep to be initiated at the object’s exterior. Thereafter, for any given cell, the fluxes on the three incoming cell planes for particles traveling in a given discrete angle are known and are used to solve for the cell center and the three outgoing cell faces. Thus, each interior cell requires in advance the solution of its three upstream neighboring cells – a three-dimensional recursion.

Figure 1 shows that on a 2-D grid, cells within a diagonal are independent of each other. Diagonal concurrency can also be the basis for an implementation using a decomposition of the mesh into subdomains and message passing to communicate the boundaries between processors, as described in [12] and shown in Figure 2.

The transport sweep is performed subdomain by subdomain in a given angular direction. Each processor’s exterior surfaces are computed by, and received in a message from, “upstream” processors owning the subdomains sharing these surfaces. Parallelization in SWEEP3D uses a 2-D processor decomposition of the spatial domain because recursion in one spatial direction cannot be eliminated.

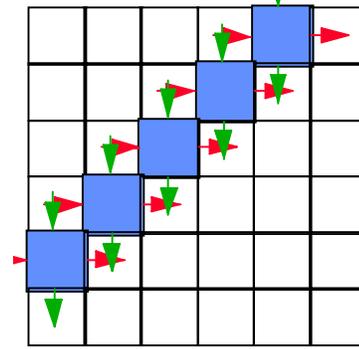


Figure 1. Diagonal Wavefront Transport Sweep

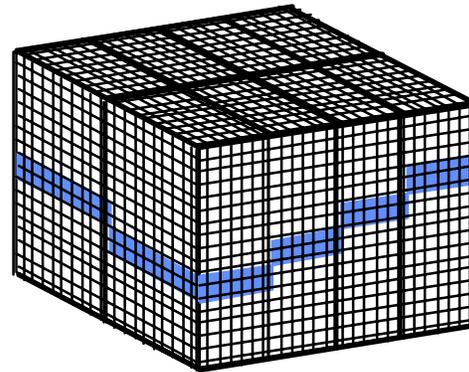


Figure 2. 2-D Domain decomposition on eight processors with 2 k-planes per block. The transport sweep has started at top of the processor in the foreground. Concurrently-computed cells are shaded.

With this domain decomposition method, parallel efficiency is limited if each processor computes its entire octant-angle-ijk local domain before communicating with its neighbors. To improve efficiency we compute only a smaller “block” of angles and k-planes before communicating. Subsequent discussions refer often to these k- and angle-block sizes, since varying these block sizes changes the balance between parallel utilization and communication time. Figure 2 shows an example with k-block size of two planes per block.

### 3. A Performance Model for Parallel Wavefronts

We use a pipelined wavefront as the basic abstraction and predict the execution time of the transport sweep as a function of primary computation and communication parameters. We use a two-parameter (latency/bandwidth) linear model for communication performance, using multiple parameters where appropriate, to capture the effects of multiple MPI buffering strategies. Computation time is parameterized by problem size, number of floating-point calculations per grid point, and a characteristic, single-CPU floating-point speed.

#### 3.1 Pipelined Wavefront Abstraction

An abstraction of the SWEEP3D algorithm partitioned for message passing on a 2-D processor domain (ij plane) is described in Figure 3. The inner-loop body of this algorithm describes a wavefront calculation with recurrences in two dimensions. Each processor must wait for boundary information from neighboring processors to the north and west before computing on its subdomain. For convenience, we assume that the implementation uses MPI with synchronous, blocking sends/receives. There is little loss of generality in this assumption since the subdomain computation must wait for message receipt. Multiple waves initiated by the octant, angle-block and k- block loops are pipelined one after another as shown in Figure 4, in which two inner loop bodies (or “sweeps”) are executing on a  $P_x$  by  $P_y$  processor grid. Each diagonal line of processors is executing the same k-block loop iteration in parallel on a different subdomain; two such diagonals are highlighted in the figure.

The number of steps required to execute a computation of  $N_{sweep}$  wavefronts, each with a pipeline length of  $N_s$  stages and a repetition delay of  $d$  is given by equation (1).

$$Steps = N_s + d(N_{sweep} - 1), \quad (1)$$

The first wavefront exits the pipeline after  $N_s$  stages and subsequent waves exit at the rate of  $1/d$ .

The pipeline consists of both computation and communication stages. The number of stages of each kind and the repetition delay per wavefront need to be determined as a function of the number of processors and shape of the processor grid. The cost of each individual computation/communication stage is dependent on problem size, processor speed and communication parameters.

#### 3.2 Computation Stages

Figure 4 shows that the number of computation stages is simply the number of diagonals in the grid. A different number of processors is employed at each stage but all stages take the same amount of time since processors on a diagonal execute concurrently. The cost of one computational stage is thus the time to complete one COMPUTE\_MESH function (see Figure 3) on a processor’s subdomain. The discussion can be summarized with Equation (2), which gives the number of steps in the computation pipeline, and Equation 3, which gives the cost of each step.

$$N_s^{comp} = P_x + P_y - 1 \quad (2)$$

$$T_{cpu} = \left( \frac{N_x}{P_x} + \frac{N_y}{P_y} + \frac{N_z}{K_b} + \frac{N_a}{A_b} \right) \frac{N_{flops}}{R_{flops}}, \quad (3)$$

Here  $N_x$ ,  $N_y$ , and  $N_z$  are the number of grid points in each direction;  $K_b$  is the size of the k-plane block;  $A_b$  is the size of the angular block;  $N_{flops}$  is the number of floating-point operations per gridpoint; and  $R_{flops}$  is a characteristic floating-point processing rate. The next sweep can begin as soon as the first processor completes its computation so the repetition delay,  $d^{comp}$ , is 1 computational step (i.e., the time for completing one diagonal in the sweep).

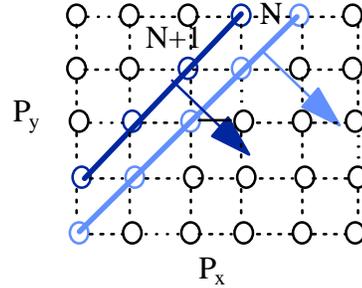


Figure 4. Multidimensional Pipelined Wavefronts

```

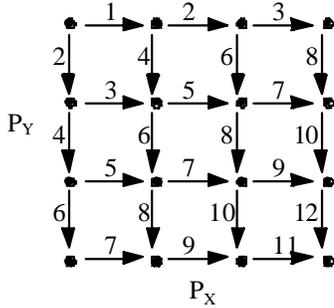
FOR EACH OCTANT DO
  FOR EACH ANGLE-BLOCK IN OCTANT DO
    FOR EACH K-BLOCK DO
      IF (NEIGHBOR_ON_EAST) RECEIVE FROM EAST (BOUNDARY DATA)
      IF (NEIGHBOR_ON_NORTH) RECEIVE FROM NORTH (BOUNDARY DATA)
      COMPUTE_MESH (EVERY I,J DIAGONAL; EVERY K IN K-BLOCK;
                    EVERY ANGLE IN ANGLE-BLOCK)
      IF (NEIGHBOR_ON_WEST) SEND TO WEST(BOUNDARY DATA)
      IF (NEIGHBOR_ON_SOUTH) SEND TO SOUTH(BOUNDARY DATA)
    END FOR
  END FOR
END FOR

```

**Figure 3. Pseudo Code for the wavefront Algorithm**

### 3.3 Communication Stages

The number and cost of communication stages are dependent on specific characteristics of the communication system. The effect of blocking synchronous communications is that messages initiated by the same processor occur sequentially in time and messages must be received in the same order that they are sent. As implemented, the order of receives is first from the west, then from the north, and the order of sends is first to the east and then to the south. These rules lead to the ordering (and concurrency) of the communications for a 4 x 4 processor grid as shown in Figure 5 for a sweep that starts in the upper-left quadrant.



**Figure 5. Communication Pipeline.**

In Figure 5 edges labeled with the same number are executed simultaneously and the graph shows that it takes 12 steps to complete one communication sweep on a 4 x 4 processor grid. We assume that a logical processor mesh can be imbedded into the machine topology such that each mesh node maps to a unique processor and each mesh edge maps to a unique router link. One can generalize the number of stages to a grid of  $P_x$  by  $P_y$  processors by observing that communication for each row of processors is initiated by a message from a north neighbor in the first column of processors. South-going messages in the first column of processors occur on every other step since each processor in the column a) has no west neighbor, and b) must send east before sending south. Thus the last processor in the first column receives a message on step  $2(P_y - 1)$ . This initiates a string of west-going messages along

the last row that are also sent on every other step, and the number of stages in the communication pipeline is given by

$$N_s^{comm} = 2(P_y - 1) + 2(P_x - 1) \quad (4)$$

Analogous to the computational pipeline, different stages of the communication pipeline have different numbers of point-to-point communications. However, since these occur simultaneously, the cost of any single communication stage is the time of a one-way, nearest neighbor communication. This time is given by:

$$T_{msg} = t_0 + \frac{N_{msg}}{B} \quad (5)$$

where  $t_0$  and  $B$  are the message latency and bandwidth, respectively.

The repetition delay for the communication pipeline,  $d^{comm}$ , is 4 because a message sent from the top-left processor (processor 0) to its east neighbor (processor 1) on the second sweep cannot be initiated until processor 1 completes its communication with its south neighbor from the first sweep (Figure 5).

### 3.4 Combining Computation and Communication Stages

In the previous two sections, we derived formulas that are general for any pipelined wavefront computation. We can summarize the discussion in two equations that give the separate contributions of computation and communication:

$$T^{comp} = [(P_x + P_y - 1) + (N_{sweep} - 1)] * T_{cpu} \quad (6)$$

$$T^{comm} = [2(P_x + P_y - 2) + 4(N_{sweep} - 1)] * T_{msg} \quad (7)$$

The major remaining question is whether the separate contributions,  $T^{comp}$  and  $T^{comm}$ , can be summed to derive the total time. They would not be additive if there were any additional overlap of communication with computation not already accounted for in each term. In a previous publication [ 16] we used a task graph for an execution consisting of two wavefronts on a 3 x 3 processor grid (Figure 6) to demonstrate that the critical path for the cal-

ulation is exactly the number given by eqns. (2) and (4).

In summary, total time for the sweep algorithm is the sum of eqns. (6) and (7), where  $T_{cpu}$  is given by eqn. (3) and  $T_{msg}$  is given by eqn. (5). Validation of the model against experiment involves the measurement and/or modeling of  $T_{msg}$ , the time needed for the completion of a send/receive pair of an appropriate size, and  $T_{cpu}$ , the time for the subgrid computation on each processor.

#### 4. Validation of the Model

In this section, we present results that validate the model with performance data from SWEEP3D on three different machines, with up to 500 processors, over the entire range of the various model parameters. Note that all figures use  $(P_x + P_y)$  as the abscissa, since this follows naturally from eqns. (6) and (7). These equations suggest the following validation regimes:

$N_{sweep} = 1$ : Validates the number of pipeline stages in  $T^{comp}$  and  $T^{comm}$ , as functions of  $(P_x + P_y)$ , in the available range of processor configurations.

$N_{sweep} \sim (P_x + P_y)$ : Validation of a case where the contributions of the  $(P_x + P_y)$  and  $N_{sweep}$  terms are comparable.

$N_{sweep} \gg (P_x + P_y)$ : Validates the pipeline repetition rate.

For each of these cases, we analyze problem sizes chosen in such a way as to make:

$T^{comp} \gg T^{comm}$ , (validate equation. (6) only)

$T^{comp} = 0$ ; (validate equation. (7) only)

$T^{comp} \sim T^{comm}$ , (validate the sum of equations. (6) and (7))

##### 4.1 $N_{sweep} = 1$

For a single sweep, the coefficients of  $T_{msg}$  and  $T_{cpu}$  in equations 6 and 7 represent the number of communication and computation stages in the pipeline, respectively. Any overlap in communication or computation during the single sweep of the mesh is encapsulated in the respective coefficients. In hypothetical problems with  $T_{msg} \sim T_{cpu}$ , and in the limit of large processor configurations (large  $P_x + P_y$ ), equations 6 and 7 show that the communication component of the elapsed time would be twice as large as the contribution of the computation time. In reality, for problem sizes and partitionings reasonably designed (small subgrid surface-to-volume ratio),  $T_{cpu}$  is considerably larger than  $T_{msg}$ . Computation is the dominant component of the elapsed time.

This is apparent in Figure 7, which presents the model-experiment comparison for a weak scalability analysis of a  $16 \times 16 \times 1000$  subgrid size sweeping only one octant. This size was chosen to reflect an estimate of the subgrid size for a 1-billion cell-problem running on a machine with about 4,000 processors; the former is a canonical goal of ASCI and the latter is simply an estimate of the

machine size that might satisfy a 3-TFLOPS-peak performance requirement. In a “weak scalability” analysis, the problem size scales with the processor configuration so that the computational load per processor stays constant. This experiment shows that the contribution of communication is small (in fact, the model shows that it is about 150 times smaller than computation), and the model is in very good agreement with the experiment.

In the absence of communication our model reduces to the linear “parallel computational efficiency” models used by Baker [9] and Koch [3] for  $S_N$  performance, in which parallel computational efficiency is defined as the fraction of time a processor is doing useful work.

To validate the case with  $N_{sweep} = 1$  and “comparable” contributions of communication and computation we had to use a subgrid size that is probably unrealistic for actual production simulation purposes ( $5 \times 5 \times 1$ ). Even with this size computation outweighs communication by about a factor of 6. Figure 8 depicts a weak scalability analysis on the SGI Origin 2000 for this size. The model-experiment agreement is again very good.

Validation of cases where  $T^{comp} = 0$  involved the using a code that simply implements a receive-west, receive-north, send-south, send-east communication pattern enclosed in loops that initiate multiple waves; i.e., there is no computation. Figure 9 shows a very good agreement of the model with the measured data from this code.

##### 4.2 $N_{sweep} \sim (P_x + P_y)$

As described in Section 3, sweeps of the domain generated by successive octants, angle blocks, and k-plane

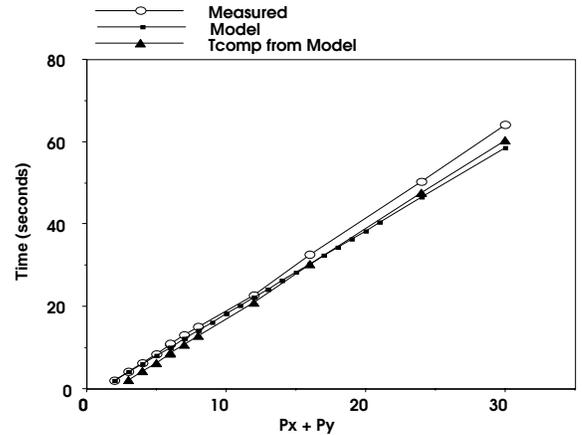


Figure 7.  $T^{comp}$  dominant.  $N_{sweep} = 1$ . IBM RS/6000.

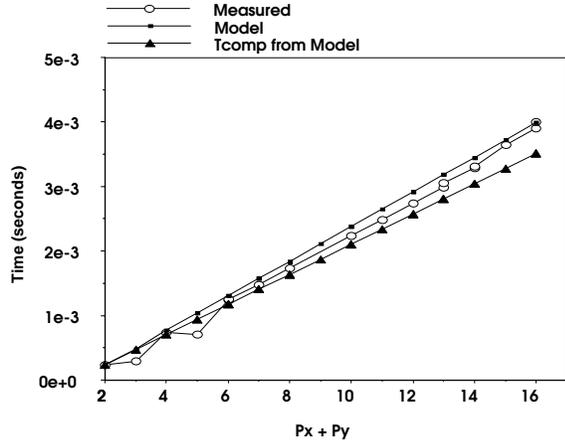


Figure 8.  $T^{comp} \sim T^{comm}$ .  $N_{sweep} = 1$ . SGI Origin.

blocks are pipelined, with the depth of the pipeline,  $N_{sweep}$ , given by the product of the number of octants, angle blocks, and k-plane blocks. We can select k- and angle-block sizes so that  $N_{sweep} = 10$ , which, in turn, balances the contribution of  $N_{sweep}$  and  $(P_x + P_y)$  for processor configurations used in this work. In Figure 10 the comparison using a data size for which  $T^{comp}$  is dominant is presented, showing an excellent agreement with the measured elapsed time.

The case with no computation is in fact a succession of 10 sweeps of the domain, with the communication overlap described by equation 6. Figure 11 shows a very good agreement with experimental data for this case.

An excellent model-experiment agreement is similarly shown in Figure 12, for a subgrid size  $5 \times 5 \times 1$ , which leads to balanced contributions of the communication and computation terms to the total elapsed time of SWEEP3D.

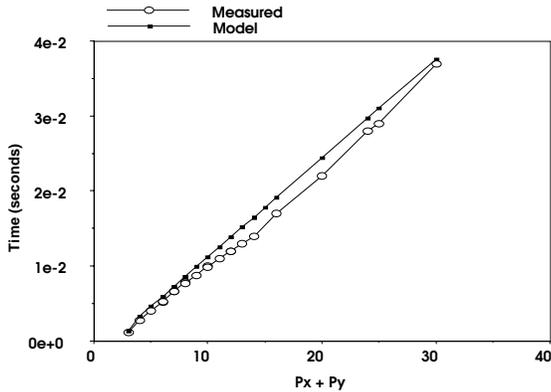


Figure 9.  $T^{comp} = 0$ .  $N_{sweep} = 1$ . SGI Origin.

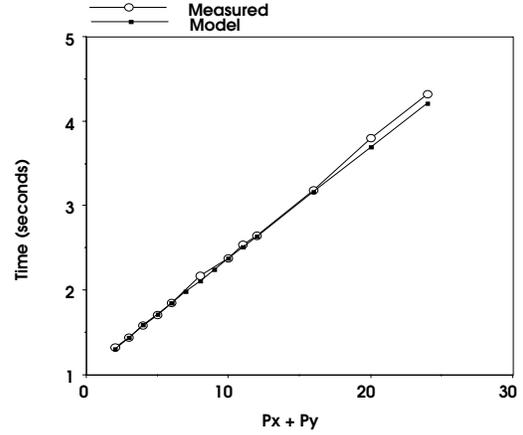


Figure 10.  $T^{comp}$  dominant.  $N_{sweep} = 10$ . SGI Origin

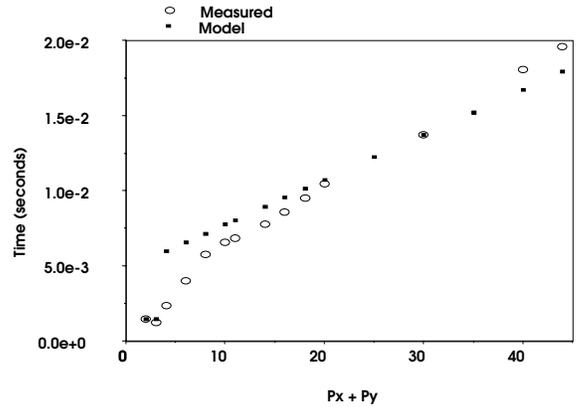


Figure 11.  $T^{comp} = 0$ .  $N_{sweep} = 10$ . CRAY T3E.

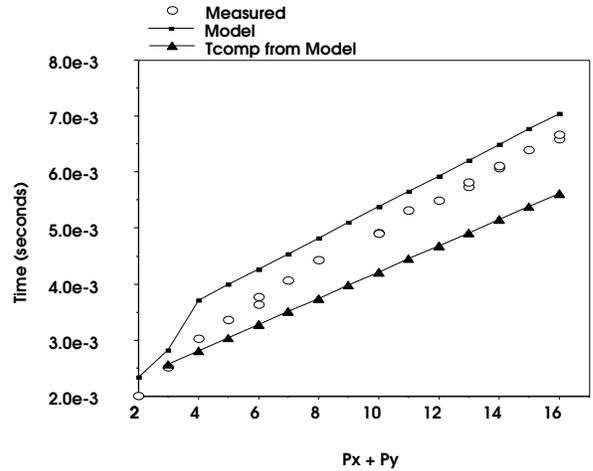


Figure 12  $T^{comp}$  dominant.  $N_{sweep} = 10$ . SGI Origin

### 4.3 $N_{sweep} \gg Px + Py$

We present model-data comparisons using weak scalabil-

ity experiments for cases in which  $N_{sweep}$  is large compared with  $(Px+Py)$  in Figure 13 ( $6 \times 6 \times 360$  subgrid;  $T^{comp} \sim T^{comm}$ ) and in Figure 14 ( $16 \times 16 \times 1000$  subgrid;  $T^{comp}$  dominant). The model is in good agreement with the measured execution times in both cases.

#### 4.4 Strong Scalability

In a “strong scalability” analysis, the overall problem size remains constant as the processor configuration increases. Therefore,  $T_{msg}$  and  $T_{cpu}$  vary as the subgrid size decreases. In Figure 15 comparison between measured and modeled time for a strong scalability analysis out to nearly 500 processors on a  $50 \times 50 \times 50$  global problem is shown. The agreement is excellent.

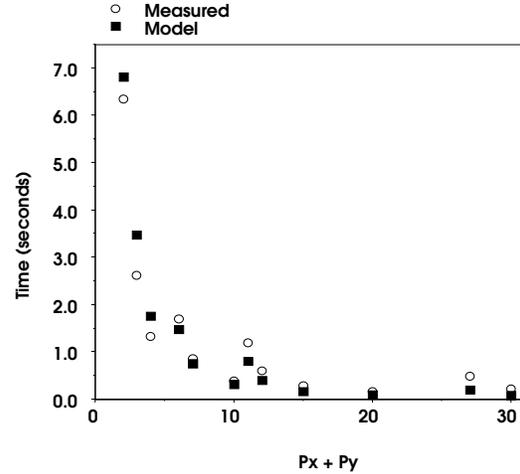


Figure 15. Strong Scalability. CRAY T3E.

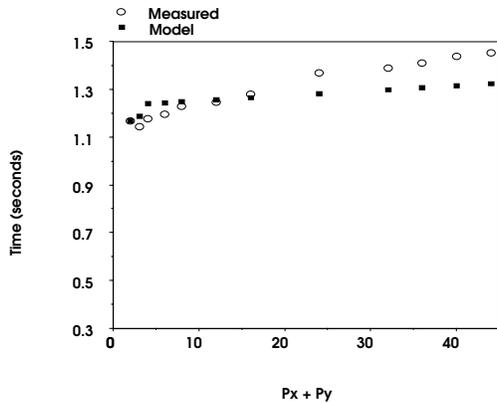


Figure 13.  $T^{comp} \sim T^{comm}$ .  $6 \times 6 \times 360$ .  $N_{sweep}$  large. CRAY T3E.  $K_b = 10$ .

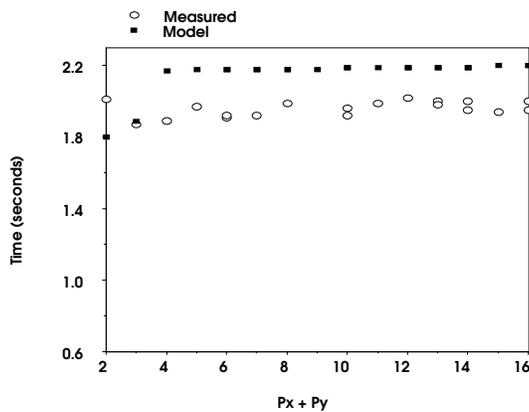


Figure 14.  $T^{comp}$  dominant.  $16 \times 16 \times 1000$ .  $N_{sweep}$  large. IBM RS/6000 SP.

#### 4.5 Blocking tradeoffs

It is of interest to investigate whether the model captures the variation of the elapsed time with the size of the angle- and k-blocks. In particular, it is important that the model correctly predicts the optimal angle- and k- blocking values for different problem sizes.

Intuitively, larger block sizes increase the computation/communication ratio due to fewer communication steps and larger message sizes. For wavefront algorithms a tradeoff occurs because smaller blocks lead to better parallel efficiency as the wavefronts have a more rapid succession over the processor array. For specific subgrid size and machine characteristics, unique optimal values for the blocking parameters result from this tradeoff.

Figure 16 shows modeled and experimental data for a  $16 \times 16 \times 1000$  subgrid with 10 k-planes per block. Compare this with Figure 17 which shows the same data on this subgrid size but with one k-plane per block. A similar comparison using a  $6 \times 6 \times 360$  subgrid is shown in Figures 18 and Figure 13 (above). For a  $6 \times 6 \times 360$  subgrid size, 10 planes per block leads to lower elapsed time, whereas for the  $16 \times 16 \times 1000$  subgrid, 1 plane per block is optimal.

The explanation is that (on the T3E), for the smaller subgrid ( $6 \times 6 \times 360$ ), larger k-blocks are required in order to increase the computation time and decrease communication. In contrast, the larger grid ( $16 \times 16 \times 1000$ ) already affords a better computation/communication ratio, so that the smaller k-block yields higher parallel efficiency. In this case, the more wavefronts generated, the better the runtime.

The model resolves the tradeoff, predicting accurate values for the blocking parameters for any grid size and machine characteristics.

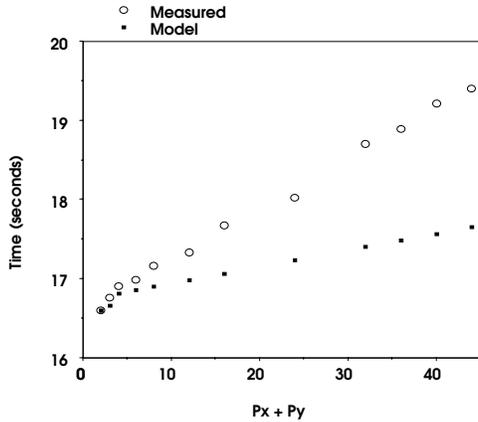


Figure 16. 16 x 16 x 1000. CRAY T3E.  $K_b = 10$ .

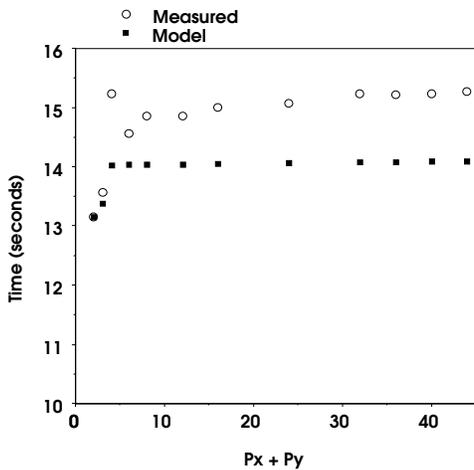


Figure 17. 16 x 16 x 1000. Cray T3E.  $K_b = 1$ .

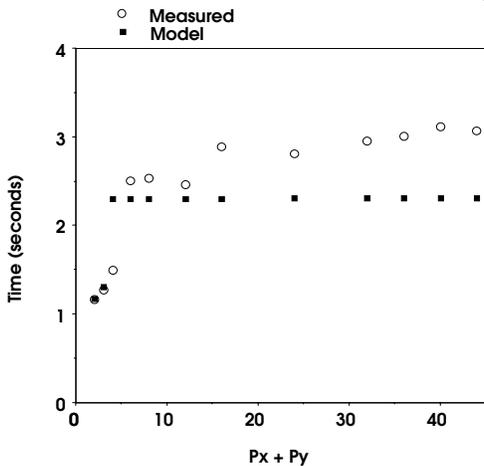


Figure 18. 6 x 6 x 360. CRAY T3E.  $K_b = 1$ .

## 5. Applications of the Model. Scalability Predictions.

Performance models of applications are important to computer designers trying to achieve proper balance between performance of different system components. ASCI is targeting a 100-TFLOPS system in the year 2004, with

a workload defined by specific engineering needs. In this section we apply our model to predict the machine parameters under which the runtime goal might be met. We assume a 100-TFLOPS-peak system composed of about 20,000 processors (5 GFLOPS peak per processor, an extrapolation of Moore's law).

Three sources of difficulty with such a prognosis are (1) making reasonable estimates of machine performance parameters for future systems; (2) managing the SWEEP3D parameter space (i.e., block sizes); and (3) estimating what problem sizes will be important. We handle the first by studying a range of values covering both conservative and optimistic changes in technology. We handle the second by reporting results that correspond to the shortest execution time (i.e., we use block sizes that minimize runtime). We handle the third as follows.

For particle transport, one ASCI target problem involves  $O(10^9)$  mesh points, 30 energy groups,  $O(10^4)$  time steps, and a runtime goal of about 30 hours. With 5,000 unknowns per grid point, this requires about 40 TBytes total memory. On 20,000 processors the resulting subgrid size is approximately  $6 \times 6 \times 1000$ . In a different ASCI scenario, particle transport problem size is determined by external factors. Based on [17], such computations will involve smaller grid sizes (20 million cells) but the full resources of the machine are still used. The 20 million-cell problem would utilize a  $2 \times 2 \times 250$  subgrid.

### 5.1. The 1 Billion-Cell Problem

Plots showing dependence of runtime with sustained processor speed and latency MPI communications latency are shown in Figures 19 and 20 for several k-plane block sizes and using optimal values for the angle-block size. Table 1 collects some of the modeled runtime data for a few important points: Sustained processor speeds of 10% and 50% of peak, and MPI latencies of 0.1, 1, and 10 microseconds. Our model shows that the dependence on bandwidth is small, and as such no sensitivity plot based on ranges for bandwidth is presented. All results assume 400 Mbytes/s MPI bandwidth [18].

One immediate observation is that runtime under the most optimistic technological estimates in Table 1 is still larger than the 30-hour goal by a factor of two. The goal could be met if, in addition to these values of processor speed and MPI latency, we used what we believe to be an unrealistically high bandwidth of 4 GBytes/s.

Assuming a more realistic sustained processor speed of 10% of peak (based on data from today's systems), Table 1 shows that we miss the goal by about a factor of six even when using  $0.1 \mu\text{s}$  MPI latency. With the same assumption for processor speed, but with a more conservative value for latency ( $1 \mu\text{s}$ ), the model predicts that we are a factor of 6.6 off. In fact, our results show that the best way to decrease runtime is to increase sustained per-

processor performance. Changing the sustained processor rate by a factor of five decreases the runtime by a factor of three, while decreasing the MPI latency by a factor of 100 reduces runtime by less than a factor of two. This is a result of the relatively low communication/computation ratio that our model predicts. For example, using values of 1  $\mu$ s and 400 MB/sec for the communication latency and bandwidth, and a sustained processor speed of 0.5 GFLOPS, the communication time will only be 20% of the total runtime.

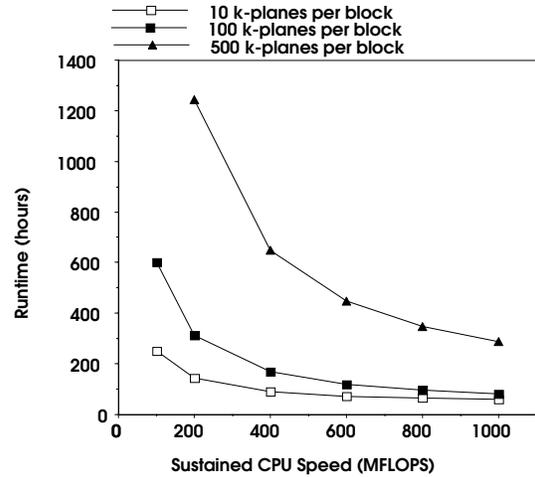


Figure 19. Model-projected sensitivity of the billion-cell transport sweep time to sustained per-processor CPU speed on a hypothetical 100-TFLOPS system for several k-plane block sizes. MPI latency = 15 ms, bandwidth = 400 Mbytes/s.

Table 1. Estimates of SWEEP3D Performance on a Future-Generation System as a Function of MPI Latency and Sustained Per-Processor Computing Rate				
MPI Latency	10% of Peak		50% of Peak	
	Runtime (hours)	Amount of Communication	Runtime (hours)	Amount of Communication
0.1 $\mu$ s	180	16%	56	52%
1.0 $\mu$ s	198	20%	74	54%
10 $\mu$ s	291	20%	102	58%

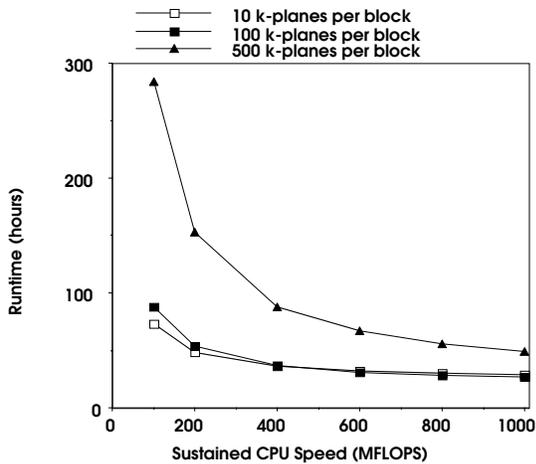


Figure 20. Model-projected sensitivity of the billion-cell transport sweep time to MPI latency on a hypothetical 100-TFLOPS system for several k-plane block sizes. Sustained per-processor CPU speed = 500 MFLOPS, bandwidth = 400 Mbytes/s.

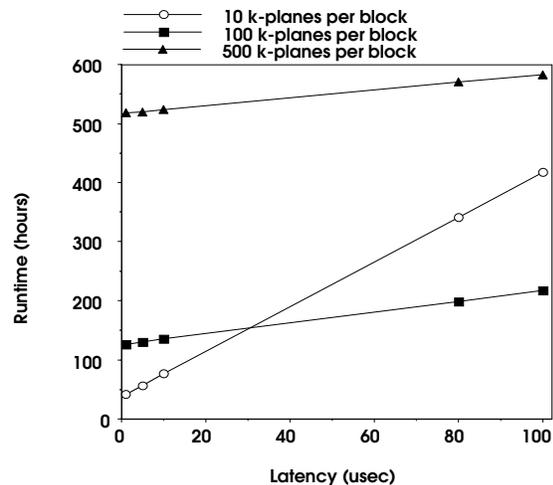
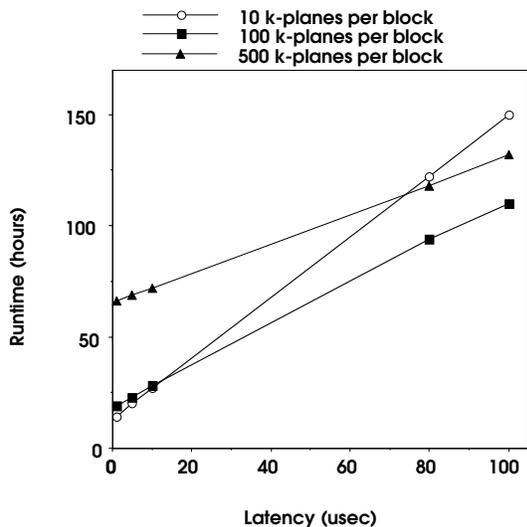


Figure 21. Model-projected sensitivity of a 20 million-cell transport sweep time to sustained per-processor CPU speed on a hypothetical 100-TFLOPS system for several k-plane block sizes. MPI latency = 15  $\mu$ s, BW=400 MB/s.

## 5.2. The 20 million-cell problem

Communication is important for this problem size – the model predicts that communication time ranges from one-half the total time to two-thirds of the total time depending on specific values for the latency and processor speed. The contribution of bandwidth to the communication cost is, again, negligible. Figures 21 and 22 show the runtime variation with MPI latency and sustained processor speed, respectively. For this problem size latency and processor speed are equally important in decreasing the runtime, as expected given the fact that the communication time is now a significant component of the total runtime.



**Figure 22. Model-projected sensitivity of the 20 million-cell transport sweep time to MPI latency on a hypothetical 100-TFLOPS system for several k-plane block sizes. Sustained per-processor CPU speed = 500 MFLOPS, bandwidth = 400 MB/s.**

## 6. Wavefront Algorithms on Clusters of SMPs

Equation (4) gives the number of steps required for a communication wavefront to pass through the 2-D processor grid. Implicit in this equation is the diagonal concurrency in communication operations shown in Figure 5. All discussion to this point assumes sufficient network links to preserve this concurrency. We now discuss the case of a cluster of SMPs, in which a portion of the network contains fewer links than required, resulting in messages that "collide" when sharing a common link. We seek to quantify the impact of this topological modification on the overall runtime of SWEEP3D. Although we have not yet obtained a closed-

form analytic model to predict this behavior, we present initial results characterizing the additional complexity of this situation. As an example, consider a two-by-two cluster of 16-processor SMPs, assuming complete connectivity within each SMP, but only a single link between SMPs. Figure 23 follows the progress of three communication wavefronts pipelined through such a system. Note that the second wave ends on communication step 33, whereas equation (4) predicts 32 steps. This is due to a conflict on communication step 14 of the second wave, indicated by the boxed label. Communication step 13 in the second wave collides with a communication step 13 in the first wave. Subsequent waves will also be delayed even more as the number of collisions increases. We have developed a computer code that calculates the number of communication steps for any SMP configuration and for any number of links between SMPs. Instead of a closed-form solution given by equation (4), the model for clusters of SMPs necessarily involves a recursive step in which the number of communication steps is obtained from the computer program. As an example, assume a system configuration of 144 X 144 processors, composed of a 9 X 9 cluster of SMPs. Each SMP has 128 processors, which we view as a logical 16 X 16 array. (This system has about the same number of processors as the 100-TFLOPS system studied in Section 5.) Equation (4) predicts that, given full interprocessor bandwidth, the number of communication steps in order to complete 160 sweeps is 1,208. (This is the number of sweeps per timestep per inner iteration in the 1-billion-cell problem of Section 5.1.) By contrast, assuming that each SMP is connected by only one link in each direction (north-south and east-west), the number of communication steps needed is 3,315. The communication component of the total runtime has increased threefold, even neglecting the effect of higher inter-SMP MPI latency.

The number of computation steps is identical to that given by equation (2), although computational waves will no longer follow a diagonal path; rather, they will be "wavy", with the distortion caused by the delay of additional communication steps. The revised formula for the total time is given by the following equation

$$T = N_s'^{Comm} * T_{msg} + N_s^{Comp} * T_{cpu}$$

where  $N_s'^{Comm}$ , the new number of communication steps, is that calculated by the computer code. In the hypothetical limit of  $T_{msg} = 0$ , this equation reduces to the computational time given by (6). Simply stated, the additional runtime of SWEEP3D on a cluster of SMPs is solely due to the increased communication time.

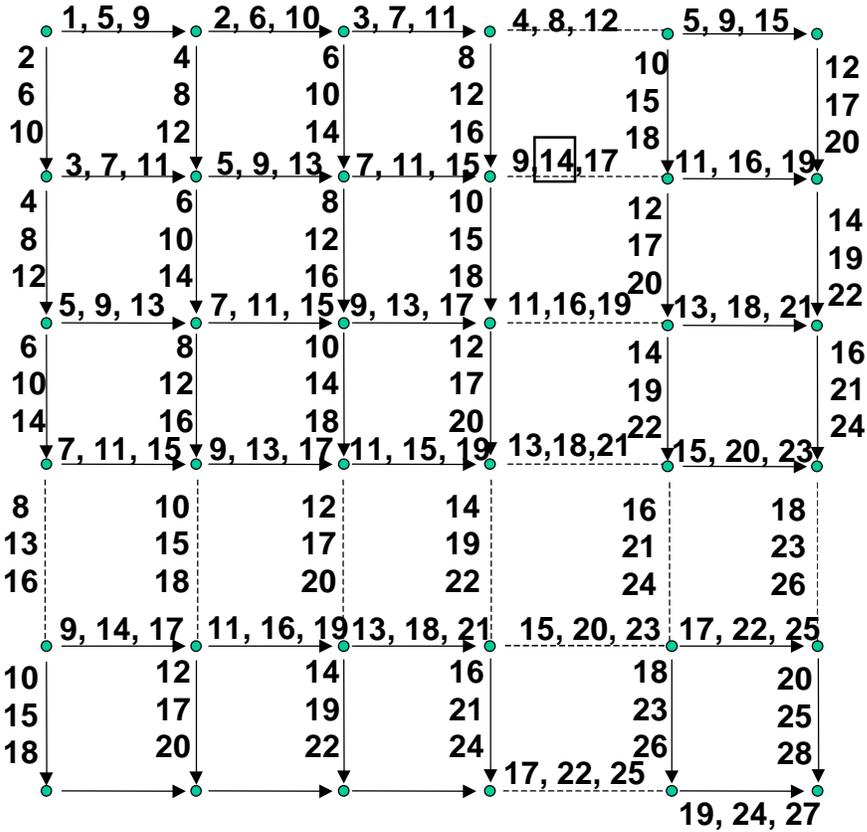


Figure 23. Communication steps on a cluster of SMPs consisting of one 16-processor SMP (4x4) in the upper left and portions of three other SMPs shown. Labels on each edge correspond to the communication steps of three consecutive sweeps. Dashed lines represent shared links between SMPs.

## 7. Conclusions

We introduced a scalability model for parallel, multi-dimensional, wavefront calculations with machine performance characterized using three parameters. The model accounts for overlap in communication and computation. Agreement with experimental data is very good under a variety of model sizes, data partitionings, blocking strategies, and on three different parallel architectures. Using our model, we analyzed performance of a deterministic transport code on a hypothetical future parallel system of interest to ASCI. A proposed 100-TFLOPS system with conservative estimates for communication bandwidth and latency improvements would not be capable of running a

billion-point ASCI  $S_N$  problem within time-limit goals. Our analysis showed that contrary to conventional wisdom, inter-processor communication performance was not the bottleneck for such a problem, although communication does become important for smaller problem sizes. For the largest problem, single-node efficiency was the dominant factor.

In the case of an SMP cluster with reduced connectivity between SMPs, preliminary analysis in this work showed a three-fold increase in the number of communication steps required for the billion-cell problem on a 100-TFLOPS system. A future publication will further refine the cluster of SMP model and validate it on the ASCI Bluemountain cluster.

## 8. Acknowledgements.

We thank Randy Baker and Ken Koch of LANL's X-Division for many helpful discussions and for providing several versions of SWEEP3D. We thank Vance Faber and Madhav Marathe of LANL Group CIC-3 for interesting discussions regarding mapping problem meshes to processor topologies. We acknowledge the use of computational resources at the Advanced Computing Laboratory, Los Alamos National Laboratory, and support from the U.S. Department of Energy under Contract No. W-7405-ENG-36. We also thank SGI/CRAY for a generous grant of computer time on the CRAY T3E system. We also acknowledge the use of the IBM SP2 at Lawrence Livermore National Laboratory.

## 9. References.

1. G. F. Pfister, In Search of Clusters – The Coming Battle in Lowly Parallel Computing, Prentice Hall PTR, Upper Saddle River, NJ, 1995, pages 219-223.
2. L. Lamport, The Parallel Execution of DO Loops," Communications of the ACM, 17(2):83:93, ?, 197.
3. K. R. Koch, R. S. Baker and R. E. Alcouffe, "Solution of the First-Order Form of the 3-D Discrete Ordinates Equation on a Massively Parallel Processor," Trans. of the Amer. Nuc. Soc., 65, 198, 1992.
4. W. D. Joubert, T. Oppe, R. Janardhan, and W. Dearholt, "Fully Parallel Global M/ILU Preconditioning for 3-D Structured Problems," to be submitted to SIAM J. Sci. Comp.
5. J. Qin and T. Chan, "Performance Analysis in Parallel Triangular Solve," IEEE Second International Conference on Algorithms & Architectures for Parallel Processing, pp 405-412, June 1996.
6. M. T. Heath and C. H. Romine, "Parallel Solution of Triangular Systems on Distributed Memory Multiprocessors," SIAM J. Sci. Statist. Comput. Vol. 9, No. 3, May 1988.
7. R. F. Van der Wijngaart, S. R. Sarukkai, and P. Mehra, "Analysis and Optimization of Software Pipeline Performance on MIMD Parallel Computers," Technical Report NAS-97-003, NASA Ames Research Center, Moffett Field, CA, February, 1997.
8. R. E. Alcouffe, "Diffusion Acceleration Methods for the Diamond-Difference Discrete-Ordinates Equations," Nucl. Sci. Eng. {64}, 344 (1977).
9. R. S. Baker and R. E. Alcouffe, "Parallel 3-D  $S_N$  Performance for DANTSYS/MPI on the CRAY T3D, Proc. of the Joint Intl'l Conf. On Mathematical Methods and Supercomputing for Nuclear Applications, Vol 1. page 377, 1997.
10. M. R. Dorr and E. M. Salo, "Performance of a Neutron Transport Code with Full Phase Space Decomposition and the CRAY Research T3D," ???
11. R. S. Baker, C. Asano, and D. N. Shirley, "Implementation of the First-Order Form of the 3-D Discrete Ordinates Equations on a T3D, Technical Report LA-UR-95-1925, Los Alamos National Laboratory, Los Alamos, NM, 1995; 1995 American Nuclear Society Meeting, San Francisco, CA, 10/29-11/2/95.
12. M. R. Dorr and C. H. Still, "Concurrent Source Iteration in the Solution of Three-Dimensional Multigroup Discrete Ordinates Neutron Transport Equations," Technical Report UCRL-JC-116694, Rev 1, Lawrence Livermore National Laboratory, Livermore, CA, May, 1995.
13. E. E. Lewis and W. F. Miller, Computational Methods of Neutron Transport, American Nuclear Society, Inc., LaGrange Park, IL, 1993.
14. R. E. Alcouffe, R. Baker, F. W. Brinkley, Marr, D., R. D. O'Dell and W. Walters, "DANTSYS: A Diffusion Accelerated Neutral Particle Transport Code," Technical Report LA-12969-M, Los Alamos National Laboratory, Los Alamos, NM, 1995.
15. R. S. Baker, K. R. Koch, "An Sn Algorithm for the Massively Parallel CM200 Computer", Nucl. Sci. and Eng., Vol 128, No. 3, p. 312, 1998.
16. A. Hoisie, O. Lubeck, and H. Wasserman, "Performance Analysis of Multidimensional Wavefront Algorithms with Application to Deterministic Particle Transport," Intl. J. High Perf. Computing, *to appear*.
17. R. S. Baker (LANL), Private Communication, June 1998.
18. C. Holt, M. Heinrich, J. P. Singh, E. Rothberg, and J. L. Hennessy, "Effects of Latency, Occupancy, and Bandwidth in DSM Multiprocessors," Stanford Univ. Comp. Sci. Report CSL-TR-95-660, 1/95.