Predictive Analysis of a Wavefront Application Using LogGP*

David Sundaram-Stukel

Epic Systems Corporation
5301 Tokay Blvd
Madison, WI 53711

dsundara@epicsys.com

Mary K. Vernon

University of Wisconsin-Madison
Computer Sciences Dept.
1210 W. Dayton Street
Madison, WI 53706-1685

vernon@cs.wisc.edu

ABSTRACT

This paper develops a highly accurate LogGP model of a complex wavefront application that uses MPI communication on the IBM SP/2. Key features of the model include: (1) elucidation of the principal wavefront synchronization structure, and (2) explicit high-fidelity models of the MPI-send and MPI-receive primitives. The MPI-send/receive models are used to derive *L*, *o*, and *G* from simple two-node micro-benchmarks. Other model parameters are obtained by measuring small application problem sizes on four SP nodes. Results show that the LogGP model predicts, in seconds and with a high degree of accuracy, measured application execution time for large problems running on 128 nodes. Detailed performance projections are provided for very large future processor configurations that are expected to be available to the application developers. These results indicate that scaling beyond one or two thousand nodes yields greatly diminished improvements in execution time, and that synchronization delays are a principal factor limiting the scalability of the application.

Keywords

Parallel algorithms, parallel application performance, LogP model, particle transport applications.

1. INTRODUCTION

This paper investigates the use of the parallel machine model called LogGP to analyze the performance of a large, complex application on a state-of-the-art commercial parallel platform. The application, known as Sweep3D, is of interest because it is a three-dimensional particle transport problem that has been identified as an ASCI benchmark for evaluating high performance parallel architectures. The application is also of interest because it has a fairly complex synchronization structure. This synchronization structure must be captured in the analytic model in order for the model to accurately predict application execution times and thus provide accurate performance projections for larger systems, new architectures, or modifications to the application.

One question addressed in this research is which of variants of the LogP model [4] is best suited for analyzing the performance of Sweep3D on the IBM SP system. Since this version of Sweep3D uses the MPI communication primitives, the LogGP model [2] which includes an additional parameter, G, to accurately model communication cost for large pipelined messages, turned out to provide the requisite accuracy. Possibly due to the blocking nature of the MPI primitives, the contention at message processing resources is negligible and thus recent extensions to LogP for capturing the impact of contention [7,12] are not needed.

In previous work [4,6,7], the LogP models have been applied to important but fairly simple kernel algorithms, such as FFT, LU decomposition, sorting algorithms, or sparse matrix multiply. Two experimental studies have applied the model to complex full applications such as the Splash benchmarks [9, 11]. However, in these studies, the effects of synchronization on application performance and scalability were measured empirically rather than estimated by the model. Many other previous analytic models for analyzing application performance are restricted to simpler synchronization structures than Sweep3D (e.g., [8]). One exception is the deterministic task graph analysis model [1], which has been shown to accurately predict the performance of applications with complex synchronization structures. The LogGP model represents synchronization structures more abstractly than a task graph. A key question addressed in this research is whether the more abstract representation is sufficient for analyzing a full, complex application such as Sweep3D.

We construct a LogGP model that not only captures the synchronization structure but also *elucidates* the basic synchronization structure of Sweep3D. Similar to the approach in [2], we use communication micro-benchmarks to derive the input parameters, L, o, and G. However, as we show in section 3, deriving these parameters is somewhat more complex for MPI communication on the SP/2 than for the Meiko CS-2; thus explicit models of the MPI-send and MPI-receive primitives are developed. Although the LogGP input parameters are derived from four-processor runs of Sweep3d, the LogGP model projects performance quite accurately up to 128 processors, for several fixed total problem sizes and several cases of fixed problem size per processor. The model also quickly and easily projects performance for the very large future processor configurations that are expected to be available to the application developers.

*This research is supported in part by DARPA/ITO under contract N66001-97-C-8533.

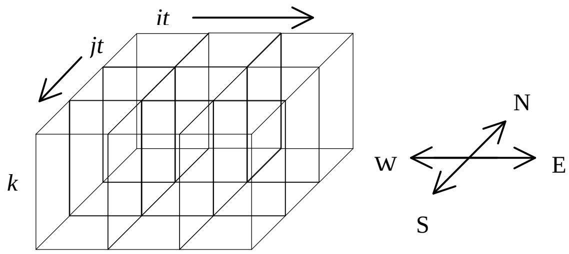


Figure 1: Partitioning the 3D Grid in the i and j Dimensions

We show several interesting results that can be derived from the analysis.

Section 2 provides a brief overview of the Sweep3D application. Section 3 derives the models of MPI-send and MPI-receive and the parameter values that characterize communication cost. Section 4 presents the LogGP equations for Sweep3D, as well as the modifications that are needed when the application utilizes the multiprocessor SMP nodes of the SP/2. In the latter case, there are two types of communication costs: intra-cluster and inter-cluster. Section 5 provides model validation results as well as performance projections for future systems. Section 6 provides the conclusions of this work.

2. Sweep3D

Sweep3D is described in [10]. A detailed task graph showing the complex synchronization among the tasks in the version of the code that is analyzed in this paper, is given in [5]. Here we give a simple overview of this version of Sweep3D, including only the aspects that are most relevant to the LogGP model. The structure of the algorithm will be further apparent from the LogGP model presented in section 4.

As its name implies, the Sweep3D transport calculations are implemented as a series of pipelined sweeps through a three dimensional grid. Let the dimensions be denoted by (i,j,k). The 3D grid is mapped onto a two-dimensional array of processors, of size $m \times n$, such that each processor performs the calculations for a partition in the i and j dimensions of size $it \times jt \times k$, as shown in Figure 1. Note that, due to the problem mapping in Figure 1, the processors in the processor grid of Figure 2 will be numbered $p_{i,j}$ where *i* varies from 1 to *n* and indicates the *horizontal* position of the processor.

A single iteration consists of a series of pipelined sweeps through the 3D grid starting from each of the 8 corners (or octants) of the grid. The mapping of the sweeps to the two dimensional processor grid is illustrated in Figure 2. If *mo* denotes the number of angles being considered in the problem, then each processor performs $it \times jt \times k \times mo$ calculations during the sweeps from each octant. To create a finer granularity pipeline, thus increasing parallelism in the computation, the block of data computed by a given processor is further partitioned by an angle blocking factor (*mmi*) and a k-plane blocking factor (*mk*). These parameters specify the number of angles and number of planes in the k-dimension, respectively, that are computed before boundary data is forwarded to the next processor in the pipeline. Each processor in the interior of the processor grid receives this boundary data from each of two neighbor processors, computes over a block based on these values, and then sends the results of its calculations to two neighbor destination processors, determined by the direction of the sweep.

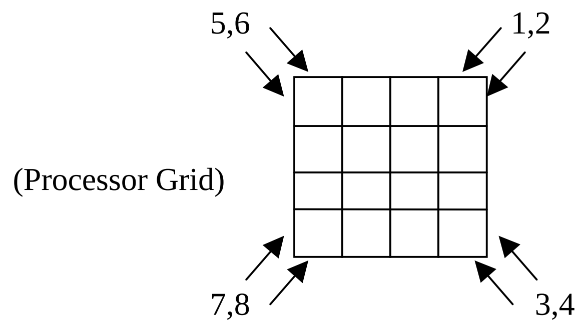


Figure 2: The Sweeps for each Octant

In the optimized version of Sweep3D that we analyze, once all blocks at a given processor are calculated for the sweeps from a given pair of octants, the processor is free to start calculating blocks for sweeps from the next pair of octants. For example, the lower left corner processor can start to compute the first block of the sweep for octant 7 after it has computed its last block of the sweep originating from octant 6. This will be shown in greater detail in the LogGP model of Sweep3D in section 4.

The pipelined sweep for octant 8 completes one iteration of the algorithm for one energy group. In the code we analyze, twelve iterations are executed for one time step. The target problems of interest to the ASCI program involve on the order of 30 energy groups and 10,000 time steps, for grid sizes on the order of 10^9 ($1000 \times 1000 \times 1000$) or twenty million (e.g., $280 \times 280 \times 255$). We can scale the model projections to these problem sizes, as shown in section 5.

3. Communication Parameters: *L, o, G*

Before we present the LogGP model of Sweep3D for the SP/2, we derive models of the MPI-send and MPI-receive communication primitives that are used in the application. The MPI-send/receive models are needed in the LogGP model of Sweep3D, and are also needed to derive two of the communication parameters values, namely the network *Latency* (*L*), and the processing *overhead* (*o*) to send or receive a message. The communication structure of Sweep3D is such that we can ignore the *gap* (*g*) parameter, as the time between consecutive message transmissions is greater than the minimum allowed value of inter-message transmission time.

Below we give the roundtrip communication times for MPI communication on the IBM SP, which are measured using simple communication micro-benchmarks. The value of *G* (*Gap per byte*) is derived directly from these measurements. We then discuss how we modeled the SP/2 MPI-send and MPI-receive primitives using the *L, o,* and *G* parameters, followed by a description of how the values of *L* and *o* are derived. A significant result is that we derive the same values of *L* and *G* (but different values of *o*) from the Fortran and the C micro-benchmark measurements. This greatly increases our confidence in the validity of the MPI communication models.

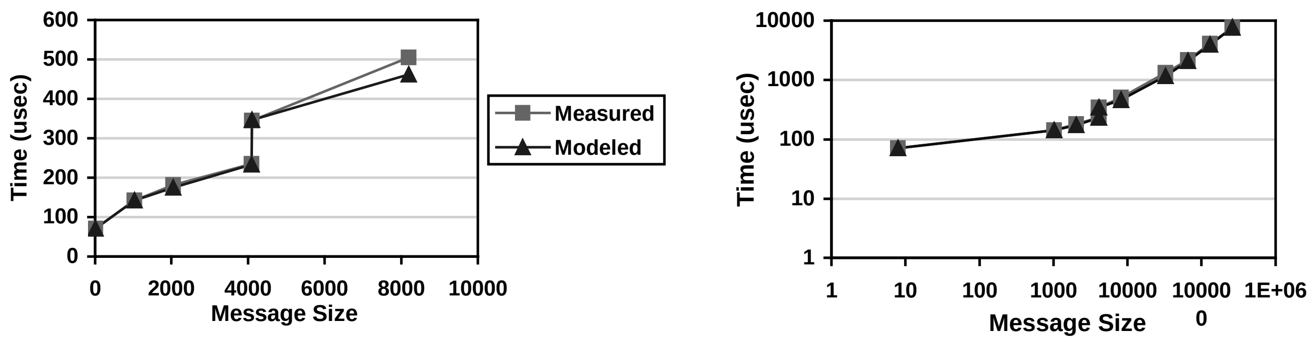


Figure 3: MPI Round Trip Communication Times.

3.1 Measured Communication Times

The roundtrip communication time as a function of message size for a simple Fortran communication micro-benchmark is given in Figures 3 (a) and (b). For each data point, a message of the given size is sent from processor A to processor B, received by a process on processor B, and immediately sent back to A. The roundtrip time is measured on A by subtracting the time just before it calls MPI-send from the time that its MPI-receive operation completes. Each figure also includes the results of our model of the roundtrip communication, which is used to derive the L and o parameters, as discussed below.

As can be seen in the figures, the measured communication time increases significantly with message size. Hence, the *G* parameter is required to accurately model communication cost. Two further points are worth noting from the Figure:

- The communication cost changes abruptly at message size equal to 4KB, due to a handshake mechanism that is implemented for messages larger than 4KB. The handshake is modeled below.
- The slope of the curve (*G*) changes at message size equal to 1KB.

The message processing overhead (*o*) is also different for messages larger than 1 KB than for messages smaller than 1KB, due to the maximum IP packet size. Thus, we will derive separate values of G_s / G_l and of o_s / o_l for "small" (<1KB) and "large" (>1KB) messages.

3.2 Models of MPI-send and MPI-receive

The models developed here reflect a fairly detailed understanding of how the MPI-send and MPI-receive primitives are implemented on the SP/2, which we were able to obtain from the author of the MPI software. It might be necessary to modify the models for future versions of the MPI library, or if Sweep3D is run on a different message-passing architecture or is modified to use non-blocking MPI primitives. The models below illustrate a general approach for capturing the impact of such system modifications.

Since the SP/2 system uses polling to receive messages, we can assume the overhead to send a message is approximately the same as the overhead to receive a message, *o*.

For messages smaller than 4KB, no handshake is required, and the total end-to-end cost of sending and receiving a message is modeled simply as:

$$\mathbf{Total\_Comm = o + (message\_size \times G) + L + o} \qquad (1)$$

where the values of *G* and *o* depend on whether the message size is larger or smaller than 1KB.

For messages larger than 4KB, the end-to-end communication requires a "handshake" in which just the header is initially sent to the destination processor and the destination processor must reply with a short acknowledgment when the corresponding receive has been posted. If the receive has been posted when the header message is sent, the end-to-end cost is modeled as follows:

$$\mathbf{Total\_Comm = o_s + L + o_s + o_s + L + o_l + (message\_size \times G_l) + L + o_l} \qquad (2)$$

Note that the processing overhead for receiving the ack is modeled as being subsumed in the processing overhead for sending the data. If the corresponding receive has not yet been posted, a additional synchronization delay will be incurred. This delay is modeled in the next section.

In addition to the total cost for communication given above, the LogGP model for Sweep3D requires separate costs for sending and receiving messages. For message size less than 4KB:

$$\mathbf{Send = o} \qquad (3a)$$

$$\mathbf{Receive = o} \qquad (3b)$$

where the value of o depends on the message size. For message size greater than or equal to 4KB:

$$\mathbf{Send = o_s + L + o_s + o_s + L + o_l} \qquad (4a)$$

$$\mathbf{Receive = o_s + L + o_l + (message\_size \times G_l) + L + o_l} \qquad (4b)$$

The receive cost includes the time to inform the sending processor that the receive is posted, and then the delay for the message to arrive.

3.3 Communication Parameter Values

Using the above equations for Total_Comm and the measured round-trip communication times, we can derive the values of L, o_s, o_l, G_s, and G_l, which are given in Table 1. The values of G_s and G_l are computed directly from the slope of the curve (in Figure 3) for the respective range of message sizes. To derive *L* and *o*, we solve three equations for Total_Comm (for message sizes less than 1KB, between 1-4KB, and greater than 4KB, respectively) in three unknowns (*L*, o_s, and o_l). Applying this method to the roundtrip time measurements obtained with C micro-benchmarks yields the same values of L and G as for the measurements

Message Size:	≤ 1024	> 1024
L	23 μsec	23 μsec
o (Fortran)	23 μsec	47 μsec
o (C)	16 μsec	36 μsec
G	0.07 μsec	0.03 μsec

Table 1. SP/2 MPI Communication Parameters

obtained with Fortran benchmarks, although the value of o is different, as shown in Table 1. This greatly increases our confidence in the validity of the above models of the MPI communication primitives. Using the parameter values derived in this way, the measured and modeled communication costs differ by less than 4% for messages between 64-256KB, as shown in Figure 3. Note that although the measured and modeled values seem to diverge at message size equal to 8KB in figure 2(a), figure 2(b) shows that the values for message sizes above 8KB are in good agreement.

4. The LogGP Model of Sweep3D

In this section we develop the LogGP model of Sweep3D, using the models of the MPI communication costs developed in section 3. We first present the model that assumes each processor in the $m\times n$ processor grid is mapped to a different SMP node in the SP/2. In this case, network latency is the same for all communication. We then give the modified equations for the case that 2×2 regions of the processor grid are mapped to a single (four-processor) SMP node in the SP/2. The round-trip times and parameter values computed in section 3 were for communication between processors in different SMP nodes. The same equations can be used to compute intra-node communication parameters.

4.1 The Basic Model

The LogGP model takes advantage of the symmetry in the sweeps that are performed during the execution, and thus calculates the estimated execution time for sweeps from one octant pair and then uses this execution time to obtain the total execution time for all sweeps, as explained below.

During a sweep, as described in section 2, a processor waits for input from up to two neighbor processors and computes the values for a portion of its grid of size mmi × mk × it × jt. The processor then sends the boundary values to up to two neighbor processors, and waits to receive new input again. Using costs associated with each of these activities, we develop the LogGP model summarized in Table 2, which directly expresses the precedence and send/receive synchronization constraints in the implemented algorithm.

The time to compute one block of data is modeled in equation (5) of Table 2. In this equation, W_g is the measured time to compute one grid point, and *mmi*, *mk*, *it* and *jt* are the input parameters, defined in section 2, that specify the number of angles and grid points per block per processor.

Consider the octant pair (5,6) for which the sweeps begin at the processor in the upper-left corner of the processor grid, as shown in Figure 2. Recall that the upper-left processor is numbered $p_{1,1}$. To account for the pipelining of the wavefronts in the sweeps, we use the recursive formula in equation (6) of Table 2 to compute the time that processor $p_{i,j}$ begins its calculations for these sweeps, where i denotes the horizontal position of the processor in the grid. The first term in equation (6) corresponds to the case where the message from the West is the last to arrive at processor $p_{i,j}$. In this case, the message from the North has already been sent but cannot be received until the message from the West is processed due to the blocking nature of MPI communications. The second term in equation (6) models the case where the message from the North is the last to arrive. Note that $\text{StartP}_{1,1} = 0$, and that the appropriate one of the two terms in equation (6) is deleted for each of the other processors at the east or north edges of the processor grid.

The Sweep3D application makes sweeps across the processors in the same direction for each octant pair. The critical path time for the two right-downward sweeps is computed in equation (7) of Table 2. This is the time until the lower-left corner processor $p_{1,m}$ has finished communicating the results from its last block of the sweep for octant 6. At this point, the sweeps for octants 7 and 8 (to the upper right) can start at processor $p_{1,m}$ and proceed toward $p_{n,1}$. Note that the subscripts on the Send and Receive terms in equation (7) are included only to indicate the direction of the communication event, to make it easier to understand why the term is included in the equation. The send and receive costs are as derived in section 3.2.

The critical path for the sweeps for octants 7 and 8 is the time until all processors in the grid complete their calculations for the sweeps, since the sweeps from octants 1 and 2 (in the next iteration) won't begin until processor $p_{n,1}$ is finished. Due to the symmetry in the Sweep3D algorithm, mentioned above, the time for the sweeps to the Northeast is the same as the total time for the sweeps for octants 5 and 6, which start at processor $p_{0,0}$ and move Southeast to processor $p_{n,m}$. Thus, we compute the critical path time for octants 7 and 8 as shown in equation (8) of Table 2.

Equation (8) represents the time until processor $p_{n,m}$ has finished its last calculation for the second octant pair. The processor

$$\mathbf{W_{i,j} = W_g \times mmi \times mk \times it \times jt} \tag{5}$$

$$\mathbf{StartP_{i,j} = max\ (StartP_{i-1,j} + W_{i-1,j} + Total\_Comm + Receive,\ StartP_{i,j-1} + W_{i,j-1} + Send + Total\_Comm)} \tag{6}$$

$$\mathbf{T_{5,6} = startP_{1,m} + 2[(W_{1,m} + Send_E + Receive_N + (m\text{-}1)L) \times \#k\text{-}blocks \times \#angle\text{-}groups]} \tag{7}$$

$$\mathbf{T_{7,8} = startP_{n\text{-}1,m} + 2[(W_{n\text{-}1,m} + Send_E + Receive_W + Receive_N + (m-1)L + (n\text{-}2)L) \times \#k\text{-}blocks \times \#angle\text{-}groups] + Receive_W + W_{n,m}} \tag{8}$$

$$\mathbf{T = 2\ (T_{5,6} + T_{7,8})} \tag{9}$$

Table 2 LogGP Model of Sweep3D

directly to its East, $p_{n-1,m}$, must start computing, calculate and communicate all needed results from the blocks for both octants, and then wait for processor $p_{n,m}$ to receive the results from the last block of these calculations and compute the results based on this block.

Due to the symmetry between the sweeps for octants 1 through 4 and the sweeps for octants 5 through 8, the total execution time of one iteration is computed as in equation (9) of Table 2.

The equation for $T_{5,6}$ contains one term [$(m-1)L$], and the equation for $T_{7,8}$ contains two terms [$(m-1)L$ and $(n-2)L$], that account for synchronization costs. These synchronization terms are motivated by the observation that measured communication times within Sweep3D are greater than the measured MPI communication cost discussed in section 3. The $(m-1)L$ term in $T_{5,6}$ and $T_{7,8}$ captures the delay caused by a send which is blocked until the destination processor posts the corresponding receive. This delay accumulates in the j direction; thus the total delay at $p_{1,m}$ depends on the number of processors to its North (m-1). Furthermore, this synchronization cost is zero for the problems with message sizes smaller than 4KB, since in this case, the processor sends the message whether or not the corresponding receive has been posted. The second synchronization delay in $T_{7,8}$, $(n-2)L$, represents difference between when a receive is posted, and when a message is actually received from the sending processor. Since a processor receives from the North *after* the West on a southeast sweep, it is more likely to wait for the message from the West. Since this delay is cumulative over all processors in the i dimension, at processor $p_{(n-1),m}$ we model this delay as $(n-2)L$. Notice that this receive synchronization term is 0 for processors on the west edge of the processor grid since there are no processors to its West from which to receive a message. This is why it was not included in the $T_{5,6}$ expression above.

4.2 The Model for the Clustered SMP Nodes

A few modifications to the above model are needed if each 2×2 region of processor grid is mapped to a single four-processor SMP cluster in the IBM SP/2, rather than mapping each processor in the grid to a separate SMP node. These changes are outlined here, in anticipation of the next generation of MPI software for the SP that will support full use of the cluster processors.

Let L_{local} denote the network latency for an intracluster message, L_{remote} denote the latency for an intercluster message, and $L_{avg} = (L_{local} + L_{remote})/2$. In the following discussion, o and G are assumed to be the same for intra-cluster and inter-cluster messages, but the equations can easily be modified if this is not the case. Let L and R be subscripts that denote a model variable (e.g,. TotalComm, Send, or Receive) that is computed using L_{local} or L_{remote}, respectively. Using this notation, the modified equations that compute the execution time of Sweep3D are given in Table 3 and described below.

Recall that processor numbering starts from 1 in both the i and j dimensions. Also recall that, for processor $p_{i,j}$, i denotes its horizontal position in the processor grid. If both i and j are even, then all incoming messages are intra-cluster and all outgoing messages are inter-cluster. The vice versa is true if both i and j are odd. This means that $StartP_{i,j}$ is computed with $TotalComm_L$, $Receive_L$, and $Send_L$ (for the incoming messages) in the former case, and with $TotalComm_R$, $Receive_R$, and $Send_R$ in the latter case. For i odd and j even, the variables in the first term of $StartP_{i,j}$ are for inter-cluster communication and the communication variables in the second term are for intra-cluster communication. The vice versa is true for i even and j odd.

The Send and Receive variables in the equations for $T_{5,6}$ and $T_{7,8}$ are all intra-cluster variables, assuming that the number of processors in each of the i and j dimensions is even when mapping 2×2 processor regions to the SMP clusters. The synchronization terms in $T_{5,6}$ and $T_{7,8}$ are computed using L_{avg}. These are the only changes required in the model.

The modified model has been validated against detailed simulation [3]. However, since we cannot yet validate them with system measurements (because efficient MPI software for intra-cluster communication doesn't yet exist), only results for the case that each processor is mapped to a separate SMP node are given in this paper. Nevertheless, the changes to the model for full cluster use are simple and illustrate the model's versatility. Furthermore, these equations can be used to project system performance for the next generation MPI software.

4.3 Measuring the Work (W)

The value of the work per grid point, W_g, is obtained by measuring this value on a 2x2 grid of processors. In fact, to obtain the accuracy of the results in this paper, we measured W_g for each per-processor grid size, to account for differences (up to 20%) that arise from cache miss and other effects. Since the Sweep3D program contains extra calculations ("fixups") for five of the twelve iterations, we measure W_g values for both of these iteration types. Although this is more detailed than the creators of LogP/LogGP may have intended, the increased accuracy is substantial and needed for the large scale projections in section 5. Furthermore, our recursive model of Sweep3D only represents the

i even, j even: $\mathbf{StartP_{i,j} = \max(StartP_{i-1,j} + W_{i-1,j} + Total\_Comm_L + Receive_L,\ StartP_{i,j-1} + W_{i,j-1} + Send_L + Total\_Comm_L)}$

i odd, j odd: $\mathbf{StartP_{i,j} = \max(StartP_{i-1,j} + W_{i-1,j} + Total\_Comm_R + Receive_R,\ StartP_{i,j-1} + W_{i,j-1} + Send_R + Total\_Comm_R)}$

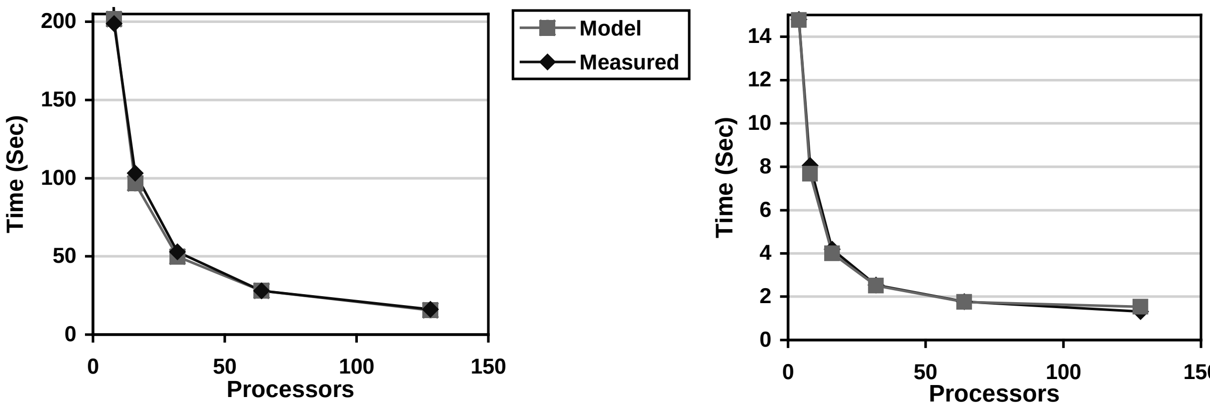
i odd, j even: $\mathbf{StartP_{i,j} = \max(StartP_{i-1,j} + W_{i-1,j} + Total\_Comm_R + Receive_R,\ StartP_{i,j-1} + W_{i,j-1} + Send_L + Total\_Comm_L)}$

i even, j odd: $\mathbf{StartP_{i,j} = \max(StartP_{i-1,j} + W_{i-1,j} + Total\_Comm_L + Receive_L,\ StartP_{i,j-1} + W_{i,j-1} + Send_R + Total\_Comm_R)}$

$$\mathbf{T_{5,6} = startP_{1,m} + 2[(W_{1,m} + Send_E + Receive_N + (m-1)L_{avg}) \times \#k\text{-}blocks \times \#angle\text{-}groups]}$$

$$\mathbf{T_{7,8} = startP_{n-1,m} + 2[(W_{n-1,m} + Send_E + Receive_W + Receive_N + (m-1)L_{avg} + (n-2)L_{avg}) \times \#k\text{-}blocks \times \#angle\text{-}groups] + Receive_W + W_{n,m}}$$

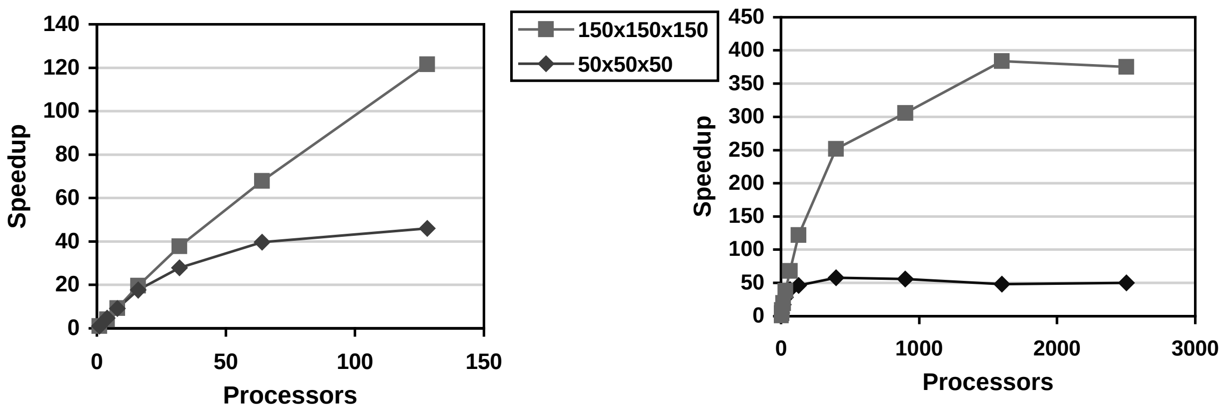
Table 3: Modified LogGP Equations for Intra-Cluster Communication on the SP/2



(a) Problem size: 150×150×150

(b) Problem size: 50×50×50

Figure 4: Validation of the LogGP Model for Fixed Total Problem Size (Fortran Code, mk=10, mmi=3)



(a) Up to 128 processors

(b) Up to 2500 processors

Figure 5: Sweep3D Speedups for Fixed Total Problem Sizes in Figure 4 (Fortran Code, mk=10, mmi=3)

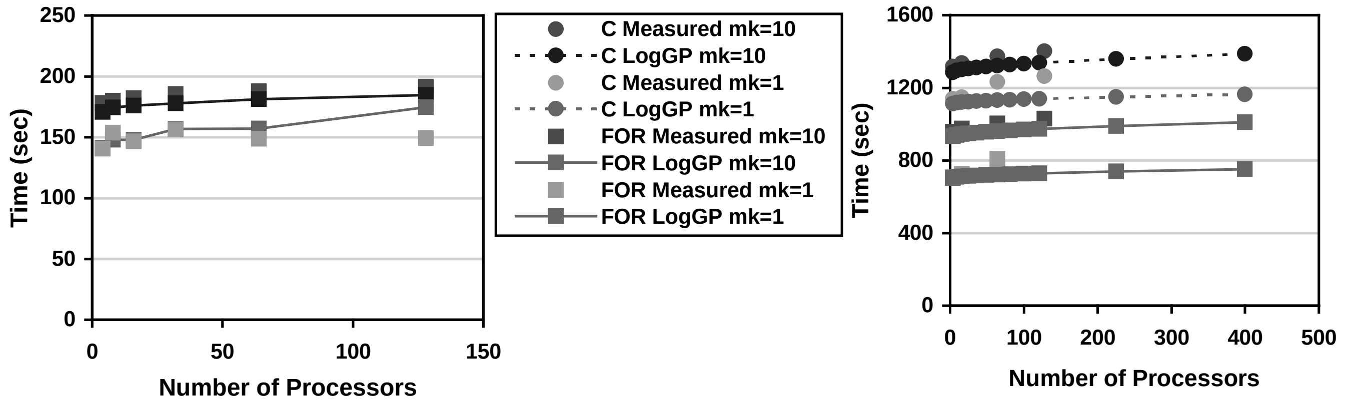
sweeps of the Sweep3D code. In addition, we measure the computation time before and after this main body of the code (*i.e.*, between the iterations for a time step). These computation times, denoted W_{before} and W_{after}, are measured during a single processor run of a specific problem size. All model parameters are thus measured using simple code instrumentation and relatively short one, two, and four-processor runs. In the next section we investigate how accurately the model predicts measured execution time for the Sweep3D application.

5. Experimental Results

In this section we present the results obtained from the LogGP model. We validate the LogGP projections of Sweep3D running time against measured running time for up to 128 processors and then use the LogGP model to predict and evaluate the scalability of Sweep3D to thousands of processors, for two different problem sizes of interest to the application developers. Unless otherwise stated the reported execution times are for one energy group and one time step with twelve iterations in the time step.

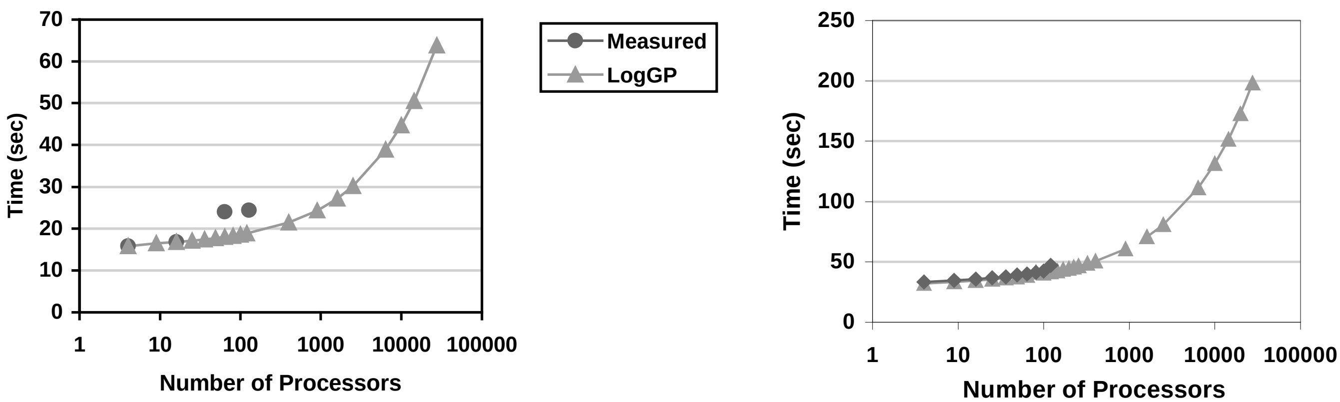
In Figure 4 we compare the execution time predicted by the LogGP model to the measured execution time for the Fortran version of Sweep3d on up to 128 SP/2 processors, for fixed total problems sizes (150×150×150 and 50×50×50), and k-blocking factor, *mk*, equal to 10. As the number of processors increases, the message size and the computation time per processor decrease, while the overhead for synchronization increases. For these problem sizes and processor configurations, the message sizes vary from over 16KB to under 1KB; there is remarkably high agreement between the model estimates and the measured system performance across the entire range. Figure 5 shows that the larger problem size achieves reasonably good speedup (*i.e.*, low communication and synchronization overhead) on 128 processors while the smaller problem size does not. Note that the model is highly accurate for both cases.

In Figure 6, we show the predicted and measured application execution time as a function of the number of processors on the SP/2, for two different cases of fixed problem size *per processor*. In Figure 6(a) each processor has a partition of the three-dimensional grid that is of size 20×20×1000. In Figure 6(b), each processor has a partition of size 45x45x1000. In these experiments, the total problem size increases as the number of processors increases. The agreement between the model estimates



(a) 20×20×1000 **(b) 45×45×1000**

Figure 6: Validation of the LogGP Model, Fixed Problem Size Per Processor (mmi=3)



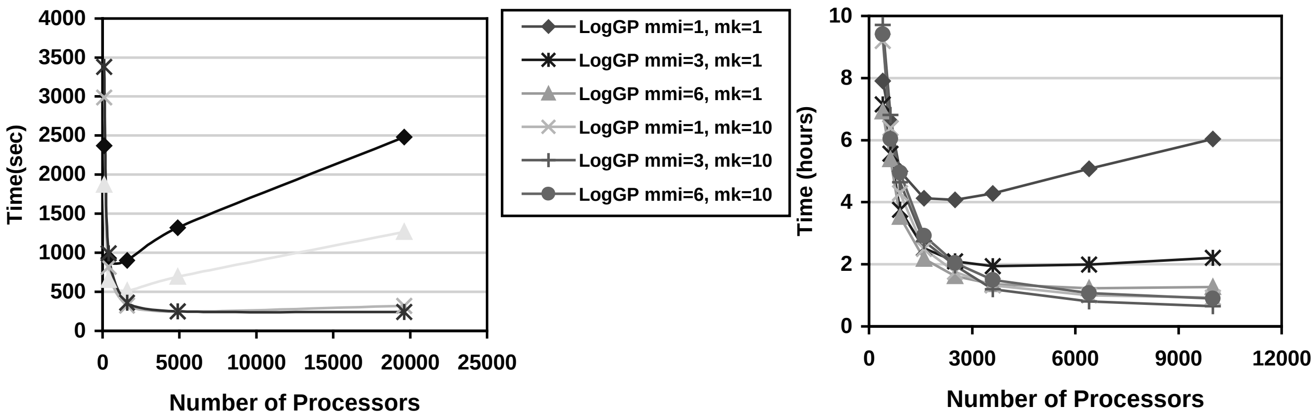
(a) 6×6×1000 (Fortran) **(b) 14×14×255 (C)**

Figure 7: Projected Sweep3D Execution Time, Fixed Problem Size Per Processor (mmi=3, mk=10)

and the measured execution time is again generally excellent for the level of abstraction in the model. However these results show that the model is less quantitatively accurate when mk=1. We have verified for many configurations that the LogGP model is qualitatively accurate in determining whether the execution time with mk=1 is higher or lower than the execution time with mk=10. We have also verified that the model is quantitatively accurate for values of mk larger than 10. The results for 45x45x1000 also illustrate that the C version of the code (which was created from the Fortran version using f2c) is somewhat slower than the Fortran code. Although the absolute performance for C differs, the performance *trends* that we report in this paper for the Fortran code are also observed in the C code and model projections.

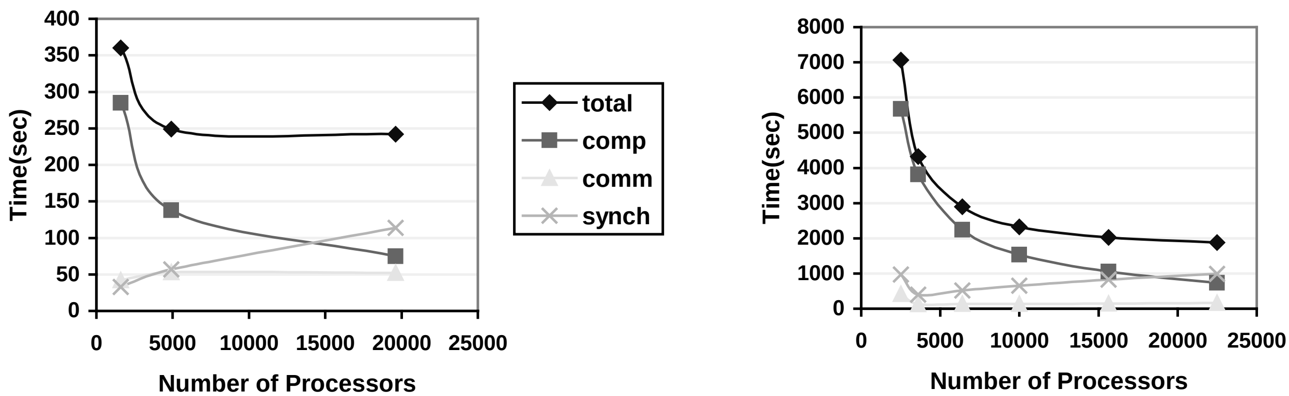
Figure 7 shows the projected execution time of Sweep3D with a fixed problem size per processor, as the system is scaled to the thousands of processors that are expected to be available at ASCI sites in the near future. Two fixed per-processor problem sizes are considered: 6×6×1000 and 14×14×255. In both cases, the model predictions have been validated to 2500 processors using simulation (not shown). The measured execution times for the 6×6×1000 case illustrate an unexplained system anomaly in which measured execution time suddenly increases for a given small increase in the number of processors. This anomaly has occurred for only a couple of the fixed per-processor grid sizes we have examined. Note that the anomaly occurs even though the problem size per processor is fixed, and thus it seems unlikely that it can be explained by cache behavior or message size. One of the hazards of modeling (analytic or simulation) is that such anomalous system behavior cannot be predicted. However, the model estimates show that the jump in execution time is not due to expected communication or synchronization costs. Detailed examination of the system implementation is required to discover, and hopefully correct, the cause of the anomaly.

As in figure 6, figures 7(a) and (b) predict excellent scaling in the case where memory usage per processor is kept constant. Nevertheless, solving the 10^9 problem size with 6×6×1000 grid points per processor requires 27,000 processors. The results in Figure 7(a) suggest that the execution time, scaled up to 30 energy groups and 10,000 time steps will be prohibitive for this problem configuration.



(a) 20 Million Grid Points **(b) 1 Billion Grid Points**

Figure 8: Projected Sweep3D Execution Times, Fixed Total Problem Size
(One Time Step, 30 Energy Groups, Fortran Code)



(a) 20 Million Grid Points **(b) 1 Billion Grid Points**

Figure 9: Projected Sweep3D Execution Time: Communication and Synchronization Costs
(One Time Step, 30 Energy Groups, Fortran, mmi=3)

Figure 8 gives the projected execution time of Sweep3D as the system is scaled to 20,000 processors, for two different *total* problem sizes of interest to the application developers. In this case, the projected execution times, for a single time step involving 12 iterations, are scaled up by a factor of 30 to reflect the fact that the computation of interest to the scientists involves 30 energy groups rather than one. Note that the problem size per processor decreases as the number of processors increases, and thus the Sweep3D configurations with larger *mk* have higher performance. The LogGP model can be used to determine which values of the Sweep3D configuration parameters (*i.e., mmi* and *mk*) yield the lowest execution time for given processor configurations and problem sizes.

One key observation from the results in Figure 8 is that there is a point of greatly diminishing improvement in execution time as the number of processors is increased beyond one or two thousand. A second key observation from Figure 8(b) is that even for optimal values of the Sweep3D configuration parameters and an unlimited number of processors, solving the billion grid point problem for 10,000 time steps appears to require a prohibitive execution time using the current algorithm.

To investigate the causes of the limited scalability in Figures 7 and 8, Figure 9 shows a breakdown of the execution time for each of the problem sizes in Figure 8. This breakdown shows how much of the critical path execution time is due to computation, non-overlapped synchronization, and non-overlapped communication. A key observation is that as the system is scaled up, synchronization delays become a significant and then dominant factor in execution time. (These synchronization delays are modeled by the (m-1)L and (n-1)L terms in equations (7) and (8) of Table 2.) Modifications that reduce the synchronization costs would be highly desirable for solving the very large problems of interest. For example, a simple modification that might be explored is to use a non-blocking form of MPI-send. However, more fundamental algorithmic changes that reduce synchronization delays may be needed. Figure 10 shows that this could yield greater benefit than improved processor technology, due to the difficulty of speeding up communication latencies.

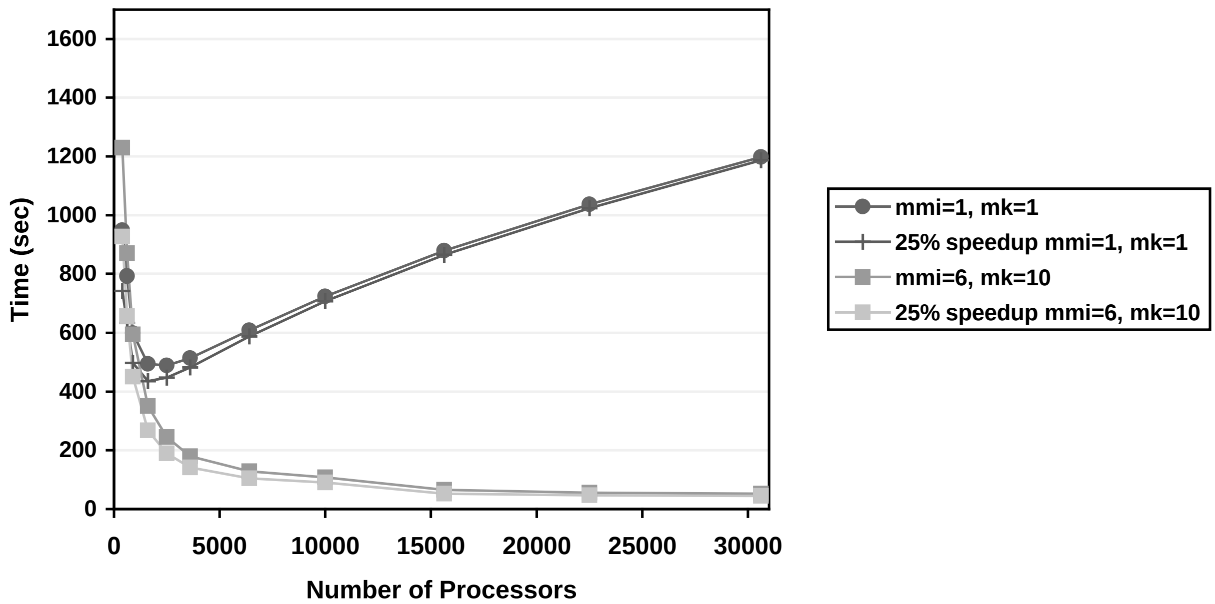


Figure 10: Projected Impact of 25% Increase in Computational Speed

6. Conclusions

The principal contribution of this research is the LogGP model for analyzing and projecting the performance of an important application that has a complex synchronization structure. For this wavefront application, the LogGP equations that capture the principal synchronization costs also elucidate the basic pipelined synchronization structure, illustrating an abstraction capability in this domain that is comparable to the simplicity of the communication parameters (L, o, and G). This research provides a case study in which the model validates extremely well against measured application performance, and further illustrates the potential of the LogGP model for analyzing a wide variety of interesting applications including the important class of wavefront applications.

The most significant results obtained for the Sweep3D application studied in this paper are as follows. First, scaling beyond one or two thousand processors yields greatly diminished returns in terms of improving execution time, even for very large problem sizes. Second, solving problem sizes on the order of 10 grid points with 30 energy groups and 10,000 time steps appears to be impractical with the current algorithm. Finally, synchronization overhead is a principal factor in limiting scalability of the application.

Future work includes generalizing the model presented in this research to create a *re-usable* analytic model of wavefront applications executing on production parallel architectures, developing a model to the shared-memory version of Sweep3D, and developing LogGP models of applications with other complex synchronization structures.

References

[1] Adve, V. S., "Analyzing the Behavior and Performance of Parallel Programs", *Ph.D. Thesis,* Technical Report #1201, Computer Science Dept, Univ. of Wisconsin – Madison, October 1993.

[2] Alexandrov, A., M. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: Incorporating Long Messages into the LogP Model", *Proc. 7th Ann. ACM Symp. on Parallel Algorithms and Architectures,* Santa Barbara, CA, July 1995.

[3] Bagrodia, R., and E. Deelman, private communication.

[4] Culler, D., R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. Von Eiken, "LogP: Towards a Realistic Model of Parallel Computation", *Proc. 4th ACM SIGPLAN Symp. On Principles and Practice of Parallel Programming (PPoPP '93),* San Diego, CA, May 1993.

[5] Deelman, E., A. Dube, A. Hoisie, Y. Luo, R. Oliver, D. Sundaram-Stukel, H. Wasserman, V. S. Adve, R. Bagrodia, J. C. Browne, E. Houstis, O. Lubeck, J. Rice, P. Teller, M. K. Vernon, "POEMS: End-to-end Performance Design of Large Parallel Adaptive Computational Systems", to appear in *Proc. 1st Int'l. Workshop on Software and Performance,* Santa Fe, October 1998.

[6] Dusseau, A., D. E. Culler, K. E. Schauser, and R. P. Martin, "Fast Parallel Sorting Under LogP: Experience with the CM-5", *IEEE Transactions on Parallel and Distributed Systems,* Volume 7, No. 8, 1996.

[7] Frank, M. I., A. Agrawal, and M. K. Vernon, "LoPC: Modeling Contention in Parallel Algorithms", *Proc. 6th ACM SIGPLAN Symp. On Principles and Practice of Parallel Programming (PPoPP '97),* Las Vegas, NV, June 1997.

[8] Harzallah, K. and K. C. Sevcik, "Predicting Application Behavior in Large Scale Shared-memory Multiprocessors, *Proc. 1995 ACM/IEEE Supercomputing Conf. (Supercomputing '95),* San Diego, CA, December 1995.

[9] Holt, C., M. Heinrich, J. P. Singh, E. Rothberg, and J. Hennessy, "The Effects of Latency, Occupancy, and Bandwidth in Distributed Shared Memory Multiprocessors", Technical Report CSL-TR-95-660, Stanford Computer Systems Laboratory, January 1995.

[10] Koch, K. R., R. S. Baker, and R. E. Alcouffe, "Solution of the First-Order Form fo the 3-D Discrete Orginates Equation on a Massively Parallel Processor", *Trans. Amer. Nuc. Soc.,* Vol. 65, 1992.

[11] Martin, R. P., A. M. Vahdat, D. E. Culler, and T. E. Anderson, "Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture", *Proc. 24th Int'l. Symp. on Computer Architecture,* Denver, June 1997.

[12] Moritz, C. A., and M. I. Frank, "LoGPC: Modeling Network contention in Message Passing Programs", *Proc. ACM Sigmetrics '98/Performance '98 Joint Conf.,* Madison, June 1998