A GENERAL PERFORMANCE MODEL FOR PARALLEL SWEEPS ON ORTHOGONAL GRIDS FOR PARTICLE TRANSPORT CALCULATIONS

A Thesis

by

MARK MICHAEL MATHIS

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2000

Major Subject: Computer Science

A GENERAL PERFORMANCE MODEL FOR PARALLEL SWEEPS ON ORTHOGONAL GRIDS FOR PARTICLE TRANSPORT CALCULATIONS

A Thesis

by

MARK MICHAEL MATHIS

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Nancy Amato
(Chair of Committee)

Lawrence Rauchwerger
(Member)

Marvin Adams
(Member)

Wei Zhao
(Head of Department)

December 2000

Major Subject: Computer Science

ABSTRACT

A General Performance Model for Parallel Sweeps on Orthogonal Grids for Particle Transport Calculations. (December 2000)

Mark Michael Mathis, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Nancy Amato

There is a growing need to accurately simulate physical systems whose evolution depends on the transport of subatomic particles. It has long been recognized that the huge computational demands of the transport problem mean that practical solution times will be obtained only by the efficient utilization of parallel processing. For example, since estimates place the time devoted to particle transport in multi-physics simulations at 50-80% of total execution time, parallelizing deterministic particle transport calculations is an important problem in many applications targeted by the Accelerated Strategic Computing Initiative of the United States Department of Energy. One common approach to deterministic particle transport calculations is the *discrete-ordinates* method, whose most time consuming step is the transport sweep which involves multiple sweeps through the spatial grid, one for each direction of particle travel. The efficient parallel implementation of the transport sweeps is the key to parallelizing the discrete-ordinates method.

The key contribution of this thesis is a new general model that can be used to compare the running times of transport sweeps on three–dimensional orthogonal grids for various mappings of the grid cells to processors. Our model, which includes machine–dependent parameters such as computation cost and communication latency, can be used to analyze and compare the effects of various spatial decompositions on the running time of the transport sweep. Insight obtained from the model yields two significant contributions to the theory of optimal transport sweeps on orthogonal

grids. First, our model provides a theoretical basis that explains why, and under what circumstances, the column decomposition of the current standard KBA algorithm is superior to the 'balanced' decomposition obtained by classic domain decomposition techniques. Second, our model enables us to identify a new decomposition, which we call *Hybrid*, that proves to be almost as good as and often better than the current standard KBA method. We obtain expressions for the completion time and discuss theoretical results.

To Star and Baby Mathis

ACKNOWLEDGMENTS

I would like to thank Nancy Amato, Marvin Adams, Lawrence Rauchwerger, Paul Nelson, Ping An, Jack Perdue and the rest of the ASCI group at Texas A&M for useful discussions regarding this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

There is a growing need to accurately simulate physical systems whose evolutions depend on the transport of subatomic particles (such as neutrons, gammas, thermal radiation, and charged particles) coupled with other complex physics (such as hydrodynamics). For example, photons of optical and near-optical wavelengths have long been used for remote sensing (e.g., in satellite imagery and lidar probing), and, more recently, have found increasing potential in important applications such as non-invasive medical diagnosis or detection of biological aerosols. Moreover, in many simulations, particle transport calculations consume the majority of the computational resources. For example, estimates place the time devoted to particle transport in certain multi–physics simulations of practical interest at 50-80% of total execution time [Hoisie et al. 1998; Hoisie et al. 1999]. It has long been recognized that the huge computational demands of the transport problem mean that practical solution times will be obtained only by the efficient utilization of parallel processing. Indeed, parallelizing deterministic particle transport calculations is recognized as an important problem in many applications targeted by the Accelerated Strategic Computing Initiative of the United States Department of Energy.

A. Discrete–Ordinates Method

One common approach to deterministic particle transport calculations is the *discrete-ordinates* method [Koch et al. 1992; Lewis and Miller 1993]. This method for dis-

---

This thesis follows the style and format of *The Journal of the ACM.*

cretizing the first–order form of the transport equation discretizes the energy variable E (usually using energy 'groups'), the angular directions Ω (using a quadrature set), and the spatial domain R (using a grid), and solves for the angular flux Ψ of particles for each spatial cell $r \in R$ for each group $g \in E$ and each direction $d \in \Omega$. The resulting large system of equations is solved iteratively. In the best sequential algorithm each iteration involves *source formation*, a *transport sweep*, and usually an *acceleration* step. In three dimensions, the number of unknowns needed to accurately characterize the transport solution can easily exceed 100,000 per particle species per spatial cell in each iteration. For example, a trilinear discontinuous finite-element spatial discretization requires 8 unknowns per hexahedral cell, a standard S_{16} discrete-ordinates angular discretization has 288 unknowns, and a typical calculation might require 50 energy groups. This rather ordinary discretization leads to $8 \times 288 \times 50 = 115,200$ unknowns per cell per particle species per time step.

In a parallel implementation of the best sequential algorithm, the most difficult step of a discrete-ordinates computation is the transport sweep which involves an ordered traversal of the spatial grid in each direction of particle travel (i.e., a sweep for each $d \in \Omega$). In this thesis, we address the particular, but important, case of regular orthogonal grids. In this case, there are only eight distinct sweep orderings in three dimensions, corresponding to the diagonal planes sweeping the grid from each of the corners. That is, if all directions in a given octant are processed together, then eight distinct sweeps through the orthogonal spatial grid must be performed in each iteration. Although each such sweep is sequential in nature, all spatial cells on the diagonal sweep plane are independent and can be processed in parallel.

The efficient parallel implementation of the transport sweeps is the key to parallelizing the best sequential algorithm for solving the discrete–ordinates equations. As the sweeps themselves must be performed in a particular order (dictated by the

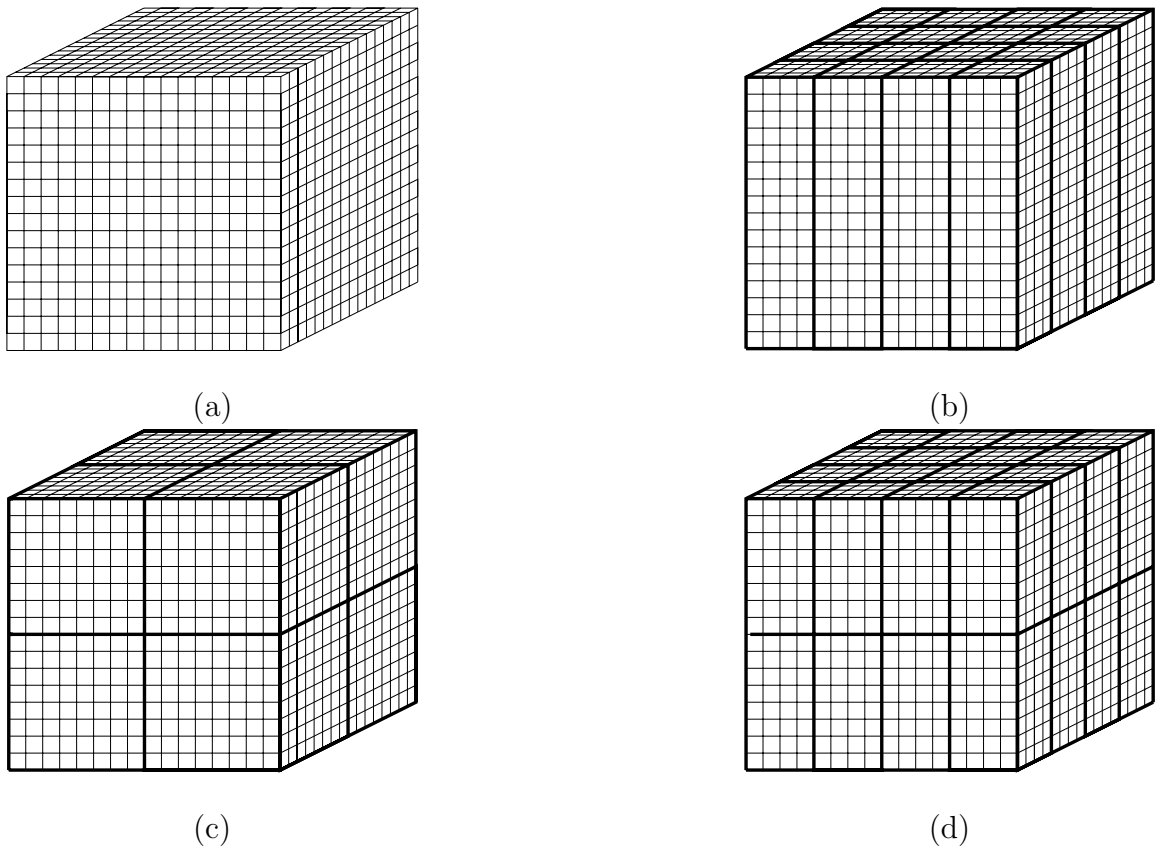


Fig. 1. Example base grids. Figure (a) is a diagram of an example base grid ($x = y = z = 16$). Figures (b), (c), and (d) demonstrate how KBA ($\phi_x = \phi_y = 4, \phi_z = 1, p = 16$), Volumetric ($\phi_x = \phi_y = \phi_z = 2, p = 8$), and Hybrid ($\phi_x = \phi_y = 4, \phi_z = 2, p = 32$) partition the base grid in (a) into 16, 8, and 32 spatial cells, respectively.

direction of particle travel), the main controllable parameter is the mapping of the spatial domain onto the processors, i.e., which spatial cells are assigned to which processors. Note that communication of data between dependent cells assigned to the same processor will have essentially no cost. While it might appear that this is an instance of the standard domain decomposition problem (see, e.g., [Karypis and Kumar 1995]), this is not the case. For example, an ideal domain decomposition would partition an orthogonal grid into sub-domains that are as 'balanced' as possible in all dimensions (since this minimizes boundary area, which directly corresponds to the amount of inter-processor communication, as in Figure 1(c)). In contrast, the Koch–

Baker–Alcouffe (KBA) algorithm [Koch et al. 1992], which is commonly considered to be the best parallel discrete-ordinates method for orthogonal grids, partitions the grid into columns (which in fact substantially increases boundary area, as in Figure 1(b)). Intuitively, the reason that classic domain decomposition methods do not work well on this problem is due to the directional dependencies of the transport sweep, which are not considered by classic domain decomposition methods.

B. Contribution

The key contribution of this thesis is a new general model that can be used to compare the running times of transport sweeps on regular orthogonal grids for any regular mapping of the grid cells to processors. In particular, our model accounts for machine-dependent parameters such as computation and communication/latency costs (assumed constant for all grid cells) and is parameterized by p, the number of processors; (x, y, z), the dimensions of the underlying spatial transport grid; and (ϕ_x, ϕ_y, ϕ_z), the dimensions of the coarse–grid processor overlay (which determines the dimensions of the sub-domain assigned to each processor). Thus, our model can be used to analyze and compare the effects of various spatial decompositions on the running time of the transport sweep.

Insight obtained from the model yields the following contributions to the theory of optimal transport sweeps on orthogonal grids.

- Our model provides a theoretical basis that explains why, and under what circumstances, the column decomposition of the current standard KBA algorithm [Koch et al. 1992] is superior to the 'balanced' decomposition' obtained by classic domain decomposition techniques.

- Our model enables us to identify a new decomposition that proves to be almost

as good as and often better than the current standard KBA method. We call this new decomposition the *Hybrid* method because it incorporates positive aspects of both the KBA and the balanced decomposition.

- A more minor (but still potentially valuable) contribution of our work is a theoretical expression for the optimal 'block size' parameter in the sweep method (the number of cells a processor should process before communicating).

Some of the results contained in this thesis appear in [Mathis et al. 2000].

C. Previous Work

To our knowledge, this is the first general model for transport sweeps on orthogonal grids. Previous modeling efforts have all concentrated on specific methods of mapping the spatial domain to processors. For example, the KBA algorithm has been analyzed in several papers [Baker and Alcouffe 1997; Koch et al. 1992]. A general predictive performance model for wavefront algorithms implemented using message passing on 2–dimensional logical processor arrays is presented in [Hoisie et al. 1998; Hoisie et al. 1999]. The assumption of 2D logical processor arrays restricts the model to column decompositions such as the KBA method, where ours allows for 3D logical processor topologies. Also, the model does not allow for multiple simultaneous sweeps where ours does. [Hoisie et al. 2000] extends the model of [Hoisie et al. 1999] to clusters of SMPs using a "pipeline with bottlenecks" abstraction, which is something that we do not consider. The authors validate their model using Sweep3D [Owens 1999], a benchmark implementation of the KBA method.

D. Outline of Thesis

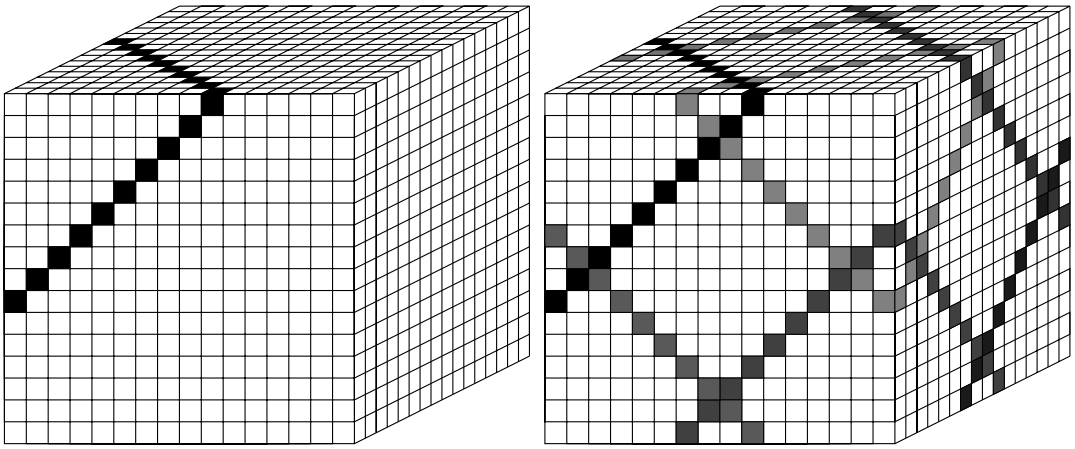
This thesis is organized as follows. In Chapter II, we define parallel transport sweeps on orthogonal grids for discrete-ordinates computations. We also introduce the three spatial decomposition methods analyzed throughout the thesis: the KBA column decomposition, a *Volumetric* decomposition, and the *Hybrid* decomposition. In Chapters III–V we develop the model. In Chapter VI we give theoretical results for our model. Chapter VII summarizes our findings.

CHAPTER II

TRANSPORT SWEEPS

As mentioned in Chapter I, the transport sweep is the most difficult step to parallelize in the best sequential algorithm for solving the *discrete-ordinates* equations [Koch et al. 1992; Lewis and Miller 1993] in deterministic particle–transport calculations. The discrete–ordinates method for solving the first-order form of the transport equation discretizes the energy variable E (usually using energy 'groups'), the angular directions Ω (using a quadrature set), and the spatial domain R (using a grid), and solves for the angular flux Ψ of particles for each spatial cell $r \in R$ for each group $g \in E$ and each direction $d \in \Omega$. The angular flux is defined such that its integral over a spatial volume, energy interval, and cone of directions is the rate at which particles are making tracks.

During each iteration, an ordered traversal through the spatial grid is performed for each direction of particle travel (i.e., a sweep for each $d \in \Omega$). A sweep of the grid begins at one of the eight corners, depending on the direction desired. On an arbitrary grid, there may be as many unique sweep orderings as there are directions in Ω. However, for the orthogonal grids considered in this thesis, there are only eight unique sweep orderings, one for each octant of directions. That is, if all directions in a given octant are processed together, then eight distinct traversals of the spatial grid must be performed in each iteration. The sweep for a given octant progresses from its starting corner across the grid to the opposite corner following a diagonal trajectory (Figure 2(a)). The constraining factor is that every cell behind a diagonal plane must be calculated before any cell ahead of the diagonal plane can be calculated. To simultaneously process all directions $d \in \Omega$, there is one sweep plane for each octant of directions (Figure 2(b)). It would appear that this inherently sequential nature



(a) (b)

Fig. 2. Parallel sweeps on orthogonal grids. In (a) and (b) we show snapshots of sweeps in progress for one sweep only and all distinct sweeps, respectively ($x = y = z = 16$, $i = 10$).

of the sweep would seriously inhibit any parallel implementation of the basic sweep algorithm. However, while the progress of the diagonal sweep plane is sequential, every cell inside the sweep plane is independent of all other cells in the same plane. Therefore, all cells in a sweep plane may be processed in parallel. Furthermore, the sweeps for each direction are independent and can be processed in parallel.

A. Spatial Decomposition

The key factor determining the parallel efficiency of the sweeps is the mapping of the spatial domain onto the processors, that is, the assignment of the spatial cells to the processors. In this thesis, we develop a model that can be used to analyze transport sweeps on regular orthogonal grids for any 'regular' mapping of the grid cells to processors. While the theory is general, our analysis will concentrate on three different methods for performing this decomposition: the column decomposition of the KBA algorithm [Koch et al. 1992] (which is the current standard), a *Volumetric*

method which uses a 'balanced' decomposition (obtained by classic domain decomposition techniques), and then a *Hybrid* approach that combines aspects of the first two methods. We have selected these three decompositions since they represent the two extreme cases (the KBA's columns vs. the Volumetric's balanced sub-domains) and an intermediate case (Hybrid).

Each method differs in the manner in which the underlying grid is partitioned among the available processors. Consider the $x \times y \times z$ grid shown in Figure 1(a) (where x, y, and z are the width, depth, and height of the grid, respectively) as the base grid. The processor partitions can be represented as a *coarse grid overlay*. For KBA (Figure 1(b)) each processor is allocated an $x/\sqrt{p} \times y/\sqrt{p} \times z$ set of spatial cells of the base grid, where p is the number of processors. Alternatively, for the Volumetric method (Figure 1(c)) each processor operates on an $x/\sqrt[3]{p} \times y/\sqrt[3]{p} \times z/\sqrt[3]{p}$ set of spatial cells. For the Hybrid method (Figure 1(d)) each processor is given an $x/\sqrt{p/2} \times y/\sqrt{p/2} \times z/2$ set.

Inter-processor communication will be required to move data from an 'upstream' cell on one processor to an adjacent 'downstream' cell on another processor, where which cells are upstream and downstream depends on the sweep direction. During a sweep, each processor will calculate one or more contiguous planes of cells in its partition before it processes the next set of cells in its partition and before the next processor can process its adjacent set of cells. It is important to note that the time to process one block of cells is highly dependent on the partitioning method being used.

CHAPTER III

A GENERAL MODEL FOR PARALLEL TRANSPORT SWEEPS

In this chapter, we develop our performance model for sweeps in three-dimensional $x \times y \times z$ regular grids, where x, y, and z are the width, depth, and height of the grid, respectively. We will develop the model considering multiple simultaneous sweeps (all octants of directions). The single–sweep case (one octant of directions) is a simplification of this general scenario.

It is useful to define a function of coarse grid overlay structure based on the number of processors and the partitioning method used:

$$\phi = (\phi_x, \phi_y, \phi_z) = \begin{cases} (\sqrt{p}, \sqrt{p}, 1) & \text{KBA,} \\ (\sqrt{p/2}, \sqrt{p/2}, 2) & \text{Hybrid,} \\ (\sqrt[3]{p}, \sqrt[3]{p}, \sqrt[3]{p}) & \text{Volumetric.} \end{cases} \tag{3.1}$$

Here, ϕ_x, ϕ_y, and ϕ_z are the width, depth, and height of the coarse grid overlay, respectively, such that $\phi_x \times \phi_y \times \phi_z = p$. Note that ϕ is not the number of cells each processor is assigned, but rather the overall dimensions of the coarse grid overlay. Now, in terms of ϕ, each processor is allocated an $x/\phi_x \times y/\phi_y \times z/\phi_z$ block of cells of the base grid. Also, while we only show three cases, we note that ϕ actually represents a family of $\sqrt[3]{p}$ partitions corresponding to $(\phi_x, \phi_y, \phi_z) = (\sqrt{p/i}, \sqrt{p/i}, i)$, $i = 1, 2, \ldots, \sqrt[3]{p}$.

A. Completion Time

The total time required by the sweep algorithm, or *completion time*, T, is the collective sum of all computation and communication required on the critical path.

$$T = T_{comp} + T_{comm}. \tag{3.2}$$

B. Computation Time

The computation time is the summation of the computation required for each individual step or *work quanta*, t_i.

$$T_{comp} = \sum_{i=1}^{z/k+\phi_x+\phi_y} t_i, \tag{3.3}$$

where

k = block size (amount of computation done before communicating).

The upper limit of the summation is equal to the critical path length of the grid, as partitioned by each method. This is the number of steps required to complete a sweep of the underlying grid. Although the sweep does not proceed as a series of 'bulk synchronous' [Valiant 1990] steps, the summation is still a valid approximation of the computation required. The reason is that the limit of the summation represents a critical path of *work quanta* that must be computed in order to complete the sweep. Note that since the limit of the summation depends on ϕ_x and ϕ_y, the critical path length will depend on the chosen spatial decomposition.

We allow for *plane blocking* by introducing a block size parameter, k. When $k = 1$, each processor's block partition will be equal to one plane of the base grid. Increasing k will allow a processor to calculate multiple planes of its subdomain before communicating. This can be advantageous since large messages can often be sent as easily as small messages. Careful selection of k allows the processors to calculate more grid cells before communicating, thus minimizing the time spent communicating and

consequently minimizing the completion time. Intuitively, k should be small when communication is inexpensive (it is not necessary to conserve messages when they are free) and larger when communication is relatively expensive. Note that k has a natural upper limit of z/ϕ_z, the number of planes in a processor's partition.

The time to complete a step or *work quanta* is the time to do all local computation for one block of cells in a processor's partition. The amount of local computation required is equal to the number of cells in a block times the amount of work required to compute one cell. These assumptions give us an expression for t_i.

$$t_i = \rho\omega k \frac{x}{\phi_x}\frac{y}{\phi_y}, \tag{3.4}$$

where

$\omega =$ local computation (work per base grid cell),

$\rho =$ sweep density (number of sweeps crossing a processor's partition).

The work per base grid cell, ω, is often referred to as the *grind time* in the transport literature. The grind time is the effective time to update a spatial cell for one energy group and one direction. Note that it must be the case that $x \geq \phi_x$ and $y \geq \phi_y$ so that the inequalities, $x/\phi_x \geq 1$ and $y/\phi_y \geq 1$, hold. In other words, the width and depth of a processor's partition must be at least one.

Since we are developing the model allowing for multiple simultaneous sweeps of different directions, a parameter must be included to account for the additional computation involved. Specifically, each processor will be required to sequentially calculate every block in its partition through which a wavefront is passing in a given step. In the single–sweep case, there is only one wavefront, so only one set of cells in a processor's partition is processed at any given step. However, for d octants proceeding simultaneously there can be as many as d sets of cells. The ρ parameter is designed

to include this information in the model. For example, the values we have chosen to use in our analysis are

$$\rho = \begin{cases} 4 & \text{KBA,} \\ 2 & \text{Volumetric \& Hybrid.} \end{cases}$$

The meaning of ρ and the justifications for the values chosen for each method are explained in detail in Chapter V. At this point it is worth noting that ρ does not necessarily equal d and may be different for different methods. Intuitively, since Volumetric and Hybrid partition the grid horizontally, sweeps for directions starting at the top of the grid do not interfere with sweeps for directions starting at the bottom of the grid. Since KBA does not partition the grid horizontally, this will not be the case.

C. Communication Time

The communication time is the summation of all outstanding communication required to perform the transport sweep. This assumption limits T_{comm} to cross–processor communication that cannot be pipelined.

$$T_{comm} = (1-\alpha)\sum_{i=1}^{z/k} L + \sum_{i=1}^{\phi_x+\phi_y+\phi_z} L \tag{3.5}$$

where

L = cost of interprocessor communication (Latency)

α = fraction of communication that can be hidden ($0 \leq \alpha \leq 1$)

The upper limits of the summations are designed to individually capture the effects of communication latencies that might be hidden on particular architectures and those that cannot be hidden on any architecture. For instance, assuming blocking commu-

nications ($\alpha = 0$), a processor must wait for the communication event to complete after computing a block of cells in its partition. This situation will occur z/k times (the number of blocks in a column of the grid). Alternatively, if non–blocking communications are available ($\alpha \approx 1$), these communication events can be hidden. Note that this term is dependent on the block size, k, but not at all on the partitioning method used. This fact will become important when we try to calculate an optimal block size. Therefore, the first summation accounts for communication events that may or may not be expensive, depending on the features of the communication subsystem. The upper limit of the second summation is derived similarly. The second summation is designed specifically to explain the effect of communication latencies that cannot be hidden, even if non–blocking communication is available. There must be at least $\phi_x + \phi_y + \phi_z$ of these, since there are at least this many cross processor boundaries. Note that this quantity is greatest for KBA and least for Volumetric. In fact, this is a lower bound on the number of communication events that cannot be hidden that favors methods with $\phi_z > 1$ (Volumetric and Hybrid). The actual number may be larger depending on the method used and the number of directions considered. However, as we will discuss in detail in Chapter VI, the model is precise for both KBA and Hybrid. Also note that the limits of the summations are mutually exclusive (i.e. no communication event is counted in both summations).

Practically, L represents the amount of time it takes for a message to pass from one processor to another. The amount of time required for communication overall is determined by α. If we assume we have access to only blocking communication primitives, α will be approximately zero. That is, essentially none of the communication can be hidden. After a processor computes one block it will produce a message containing the results of the computation. Then it must wait until the message has been consumed before continuing. However, if non–blocking communication is available, α

will be much larger, ideally approaching one. In this case, after a processor computes one block it may send the results and immediately start the next block.

Note that the amount of information communicated at each step also depends on k. However, we allow a uniform cost for communication, regardless of the amount of information. This assumption is based on the observation that the startup time associated with sending a message often outweighs the transmission time. Also note that this model is derived considering multiple simultaneous sweeps. Without this assumption, there would actually be more communication that cannot be hidden than is accounted for by our model. This extra communication is related to the startup time, that is the time it takes to get all the processors working. Basically, there is some delay that cannot be hidden when data is sent to an idle processor. Our model counts only the startup costs incurred at the beginning of the computation. It may be the case that additional 'startup' costs will be incurred during the computation. The number of these 'restarts' is related to the number of horizontal partitions, ϕ_z, and the size of the input grid, x, y, and z. The potential effects of this extra communication on the partitioning methods will be explained further in Chapter VI.

D. Substitute and Reduce

Now, substituting the expressions for T_{comp} and T_{comm} into the summation and simplifying gives us an expression for the completion time, T.

$$\begin{aligned} T &= T_{comp} + T_{comm} \\ &= \sum_{i=1}^{z/k+\phi_x+\phi_y} \rho\omega k \frac{x}{\phi_x}\frac{y}{\phi_y} + (1-\alpha)\sum_{i=1}^{z/k} L + \sum_{i=1}^{\phi_x+\phi_y+\phi_z} L \\ &= (\frac{z}{k}+\phi_x+\phi_y)(\rho\omega k \frac{x}{\phi_x}\frac{y}{\phi_y}) + (1-\alpha)L(\frac{z}{k}) + L(\phi_x+\phi_y+\phi_z) \\ &= \rho\omega xy(\frac{z}{\phi_x\phi_y} + \frac{k}{\phi_x} + \frac{k}{\phi_y}) + (1-\alpha)L(\frac{z}{k}) + L(\phi_x+\phi_y+\phi_z) \end{aligned} \tag{3.6}$$

We find it useful to normalize the equation by dividing through by ω.

$$\frac{T}{\omega} = \rho xy(\frac{z}{\phi_x \phi_y} + \frac{k}{\phi_x} + \frac{k}{\phi_y}) + (1-\alpha)\frac{L}{\omega}(\frac{z}{k}) + \frac{L}{\omega}(\phi_x + \phi_y + \phi_z) \tag{3.7}$$

CHAPTER IV

OPTIMAL BLOCK SIZE

The performance of any method can be tuned using the block size parameter, k. Specifically, we want to choose a k such that the completion time is minimized. The optimal value of k (k_{opt}) can easily be found by minimizing Equation 3.7 with respect to k.

$$\begin{aligned}
\frac{\partial(T/\omega)}{\partial k} &= \rho xy(\frac{1}{\phi_x} + \frac{1}{\phi_y}) - (1-\alpha)(\frac{L}{\omega})(\frac{z}{k^2}) \\
0 &= \rho xy(\frac{\phi_x + \phi_y}{\phi_x \phi_y}) - (1-\alpha)(\frac{L}{\omega})(\frac{z}{k_{opt}^2}) \\
(1-\alpha)(\frac{L}{\omega})(\frac{z}{k_{opt}^2}) &= \rho xy(\frac{\phi_x + \phi_y}{\phi_x \phi_y}) \\
k_{opt}^2 &= (\frac{(1-\alpha)(L/\omega)z}{\rho xy})(\frac{\phi_x \phi_y}{\phi_x + \phi_y}) \\
k_{opt} &= \sqrt{(\frac{(1-\alpha)(L/\omega)z}{\rho xy})(\frac{\phi_x \phi_y}{\phi_x + \phi_y})}
\end{aligned} \tag{4.1}$$

Of course, we are only interested in positive integer values of k_{opt}. Therefore, k_{opt} should be rounded and we must be content with an *almost* optimal value of k. Note that k_{opt} depends on the number of processors used. This means that k_{opt} can only be chosen once the number of processors to be used is known. Also, since k_{opt} depends on ϕ_x and ϕ_y, as well as ρ, k_{opt} will be highly dependent on the method used.

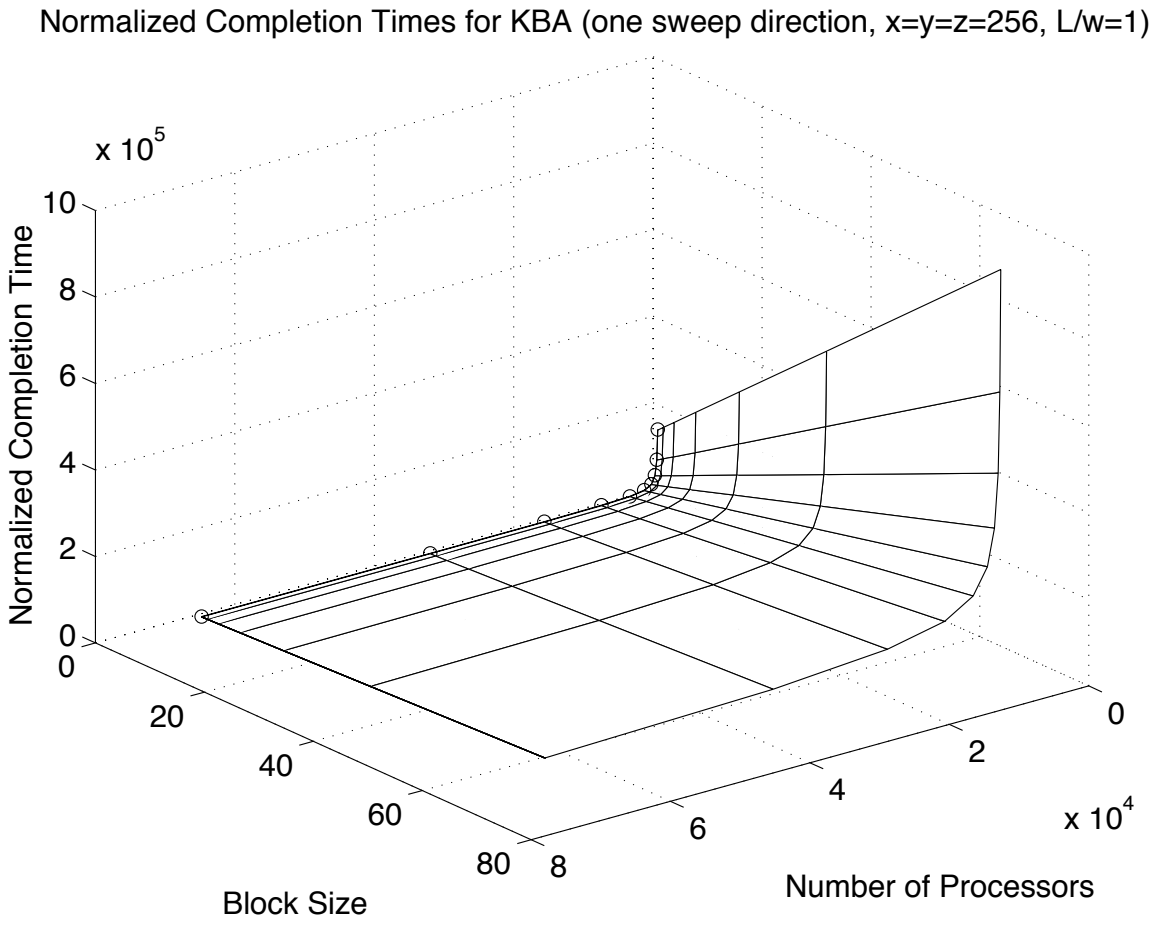
Examples of the relationship between T/ω (using Equation 3.7) and k for each method for different values of L/ω (a *machine parameter* representing the relative cost of communication vs. computation) are displayed in Figure 3 assuming blocking communications, $\alpha = 0$. The minimal values of T/ω derived using k_{opt} (from Equation 4.1) are shown as lines on the three–dimensional surfaces. In general, T/ω does not change much with k and it is generally best to choose a small value for k.

However, as L/ω increases (as communication becomes relatively more expensive), it becomes advantageous to choose a slightly larger value of k, i.e., to have fewer (larger) communications (recall our assumption that communication cost does not depend on message size). This is particularly advantageous when using a very large number of processors.

The information in these figures follows logically from Equation 3.7. For example, let us assume that L/ω is large (communication is relatively more expensive than computation). In this situation, the magnitude of L/ω will increase the contribution of the second and third (communication) terms in Equation 3.7. In order to decrease the effect of L/ω we deduce that it may be advantageous to increase k. However, although increasing k will decrease the effect of the communication terms, it will also increase the contribution of the first (computation) term. So, although we may want to increase k, we will probably only increase k slightly. We see from Figure 3(g-i) that this is indeed the case. In fact, the only time it seems profitable to choose a very large k is when ϕ_x and ϕ_y are also very large, which also follows from Equation 3.7.

For another example, let us assume that L/ω is small (communication is relatively inexpensive). Now, the magnitude of the communication terms will be much smaller than the magnitude of the computation term. In this case, it will probably not be useful to choose a very large k. Figure 3(a-c) confirm our suspicions.

For non–blocking communications, $\alpha \approx 1$, k_{opt} is always one. This fact follows from Equation 4.1. As α approaches one, the term $(1 - \alpha)$ approaches zero, and consequently k_{opt} also approaches zero. However, the block size cannot be fractional and it must be at least one. This fact also makes sense physically. If it does not cost anything to send messages, why not send as many messages as one wishes? Selecting the block size is designed to minimize the effect of communication on the completion time and is therefore unnecessary if the effect of communication is already negligible.



(a)

Fig. 3. Effect of k on the normalized completion time. These 3D surfaces demonstrate the effect of the block size parameter k on the normalized completion times of KBA ((a), (d), and (g)), Volumetric((b), (e), and (h)), and Hybrid((c), (f), and (i)). These plots are for a sweep of a single direction in a cubic three–dimensional spatial domain ($x = y = z = 256$) over $1 \leq k \leq 64$. We look at three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-c)), communication is moderately expensive ($L/\omega = 10$, (d-f)), and communication is relatively expensive ($L/\omega = 100$, (g-i)). The normalized completion times obtained using k_{opt} are traced out as lines on the 3D surfaces. All plots are for blocking communication ($\alpha = 0$).

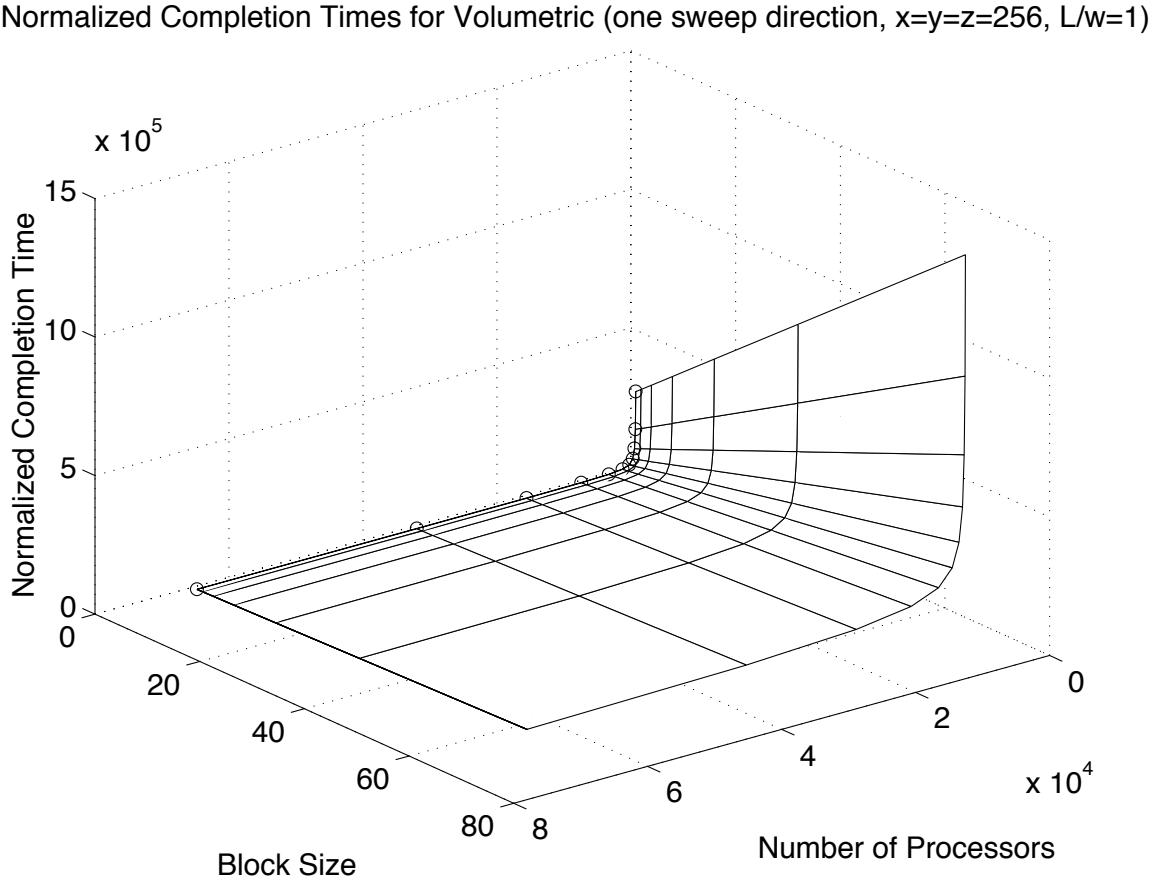


Fig. 3 Continued (b)

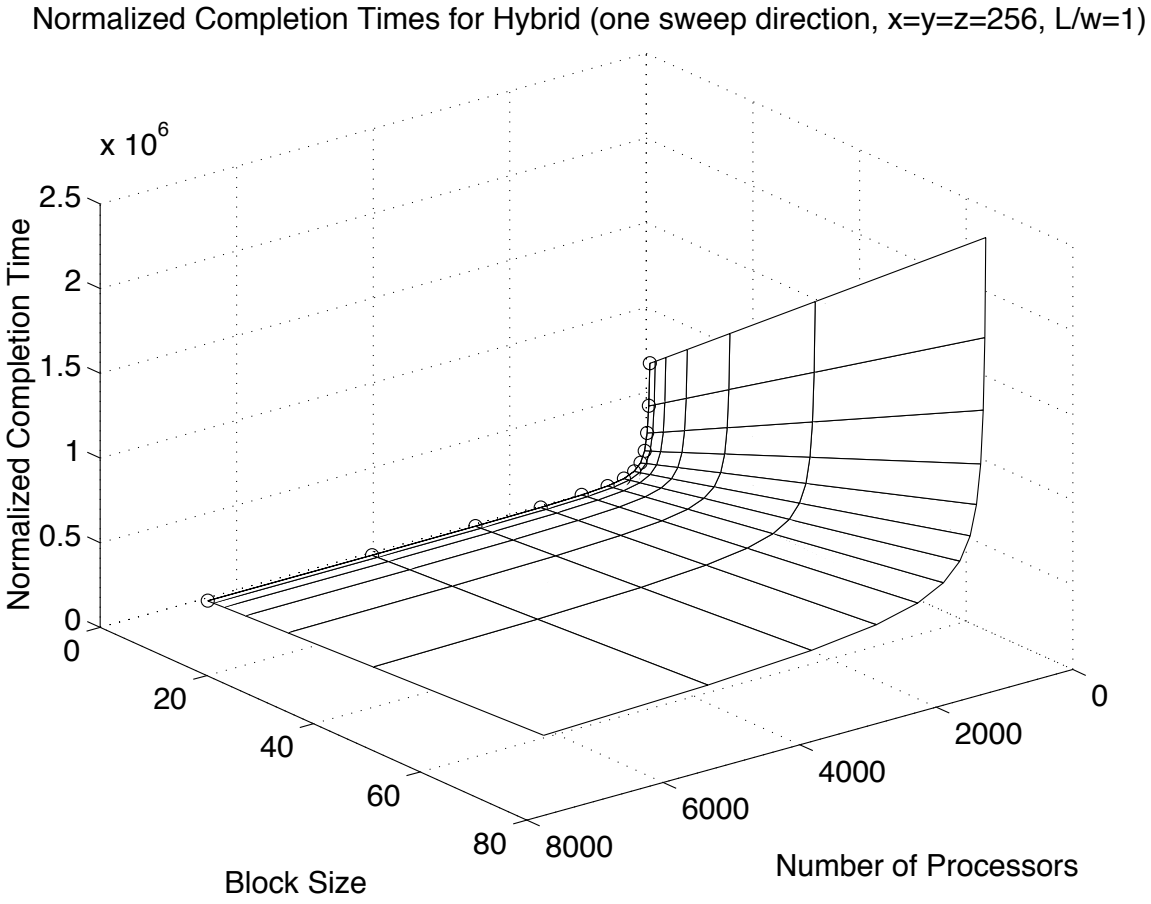


Fig. 3 Continued (c)

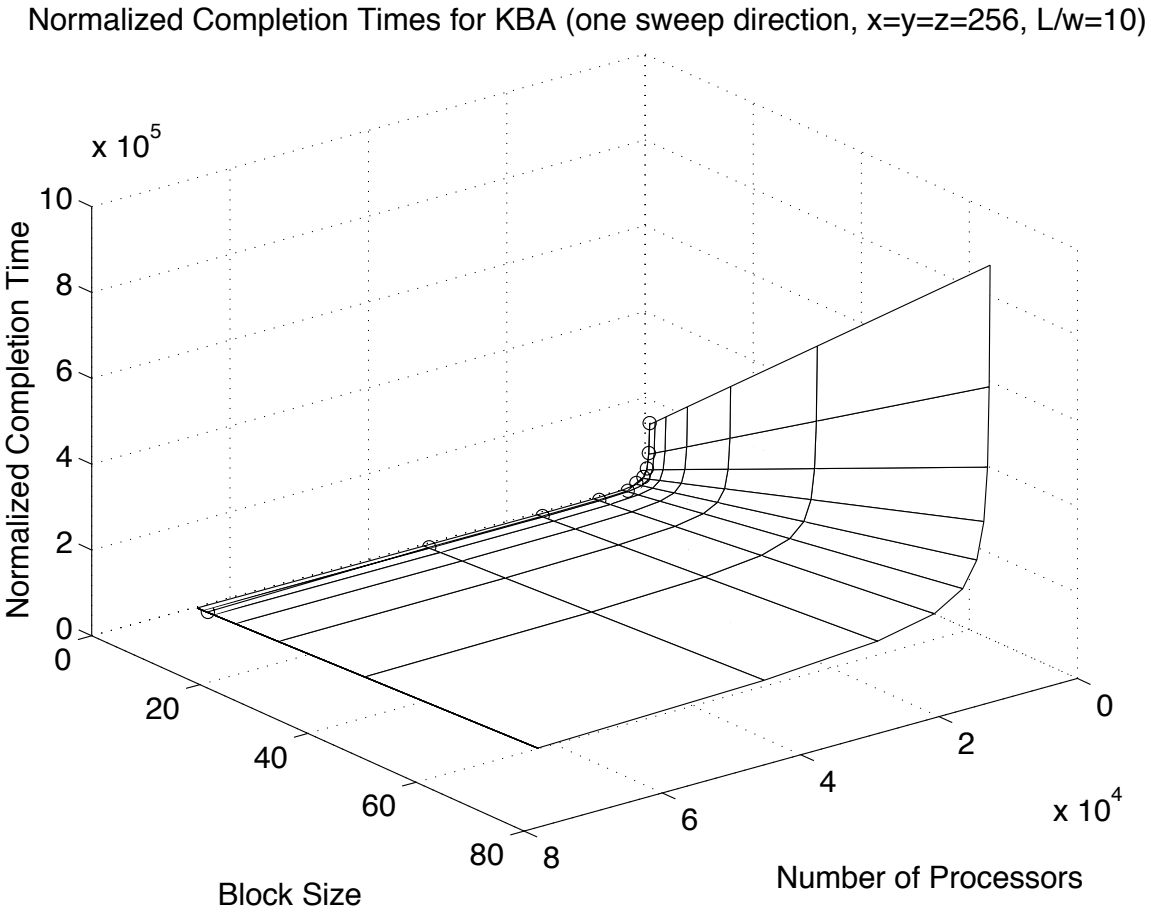


Fig. 3 Continued (d)

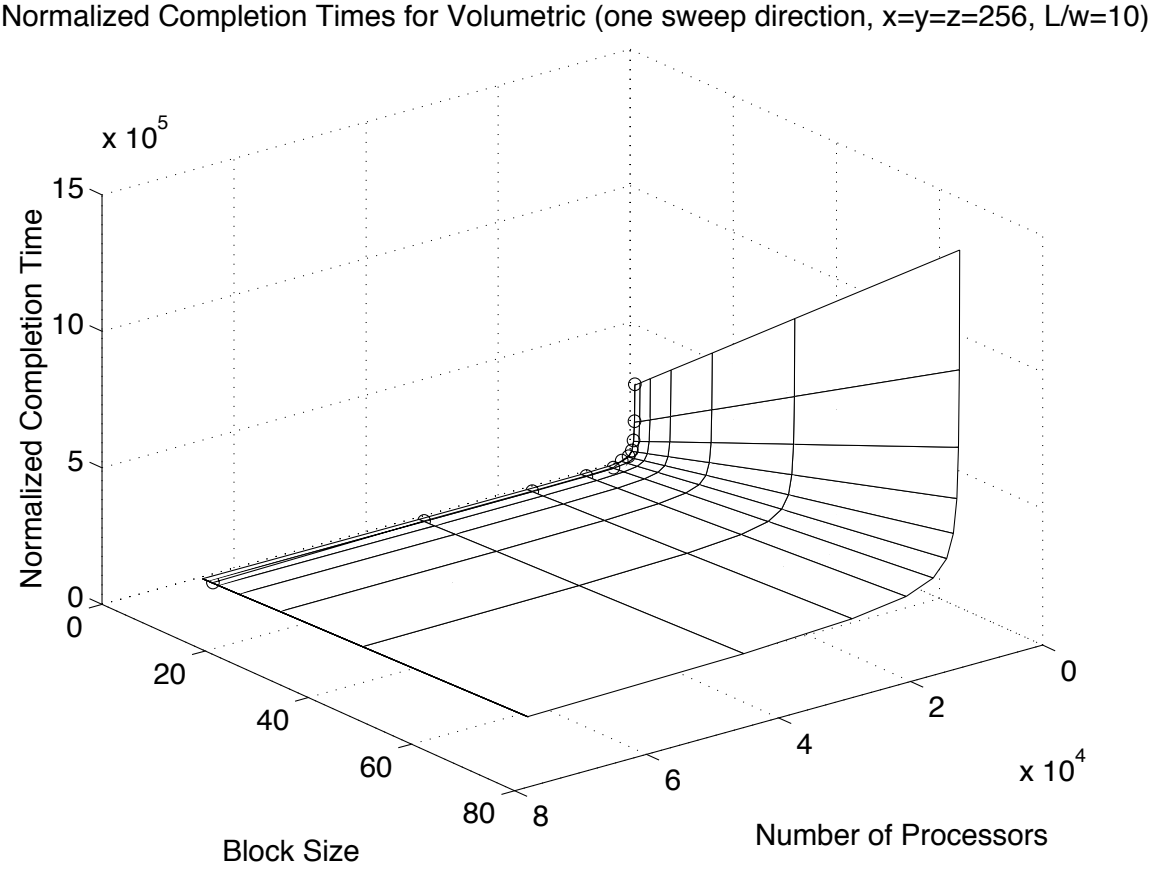


Fig. 3 Continued (e)

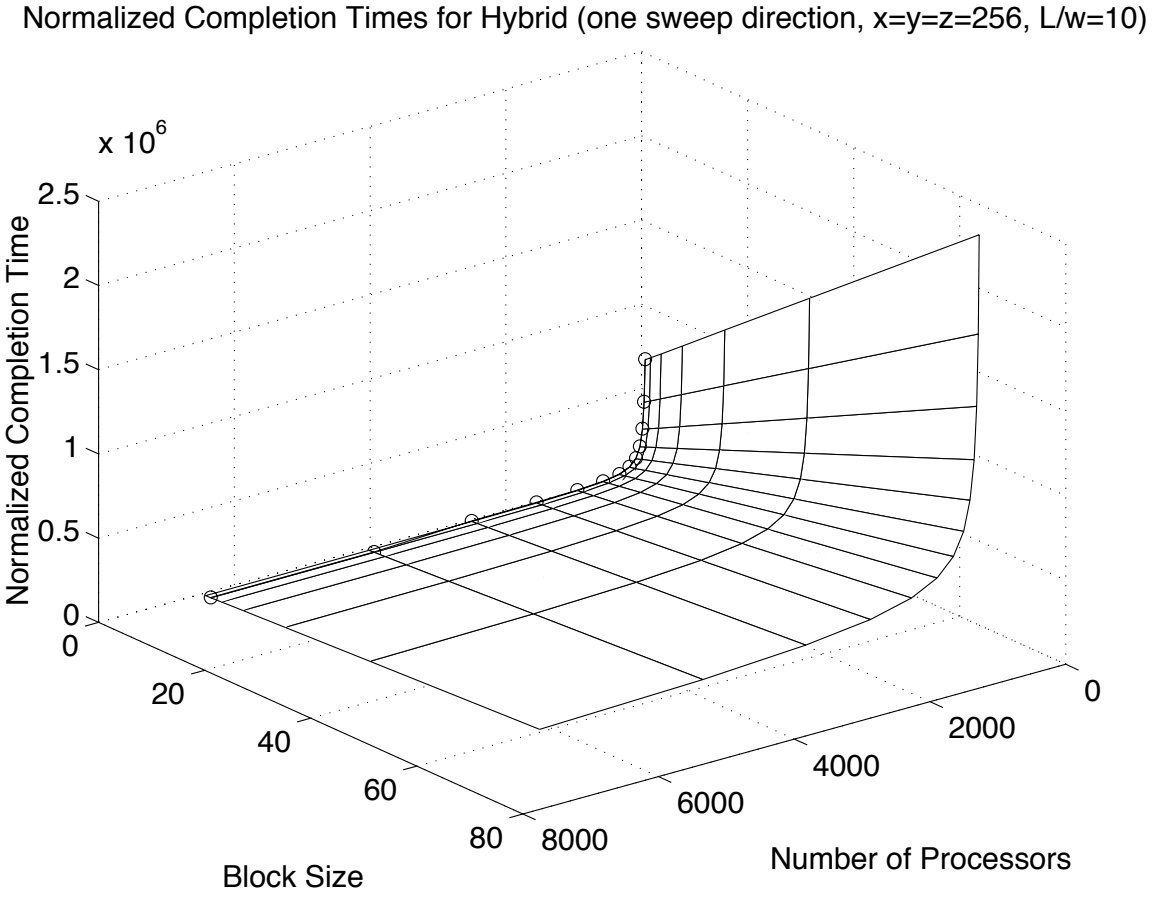


Fig. 3 Continued (f)

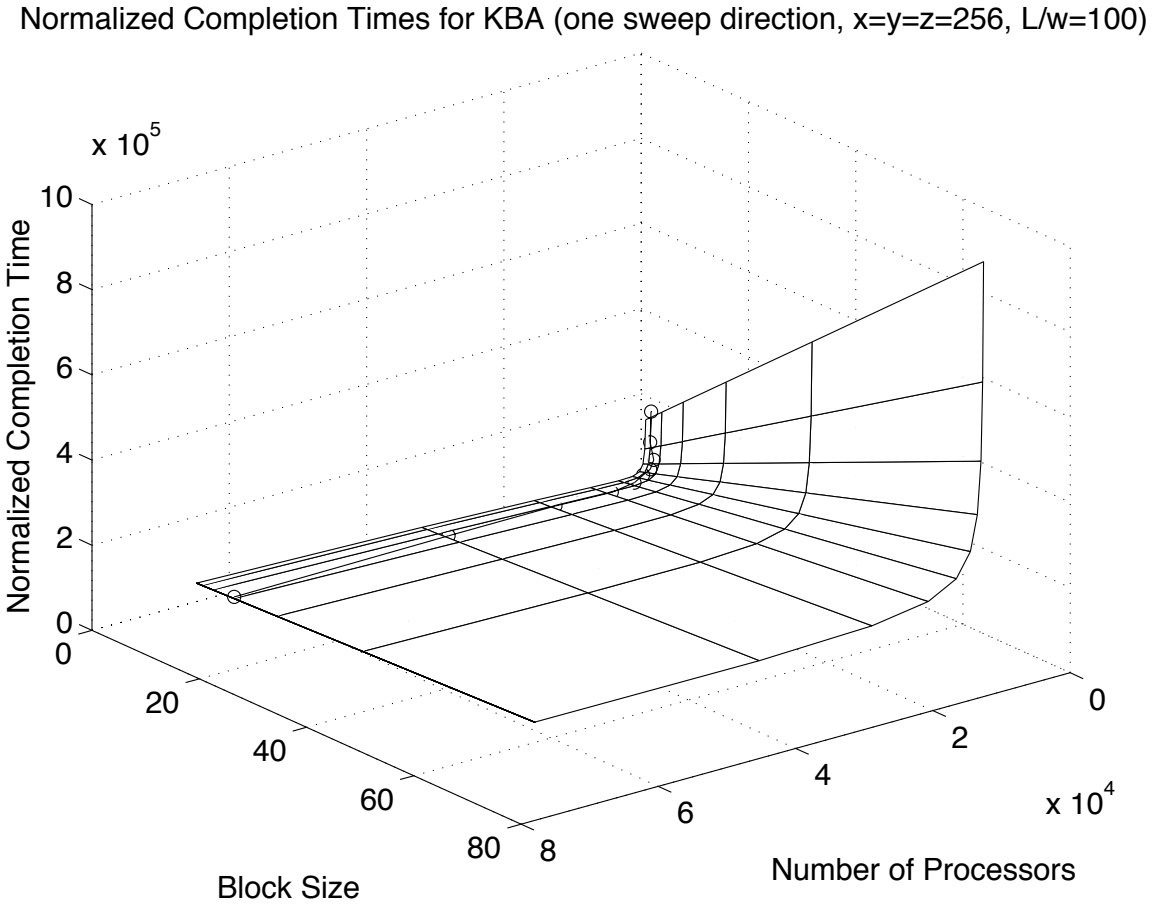


Fig. 3 Continued (g)

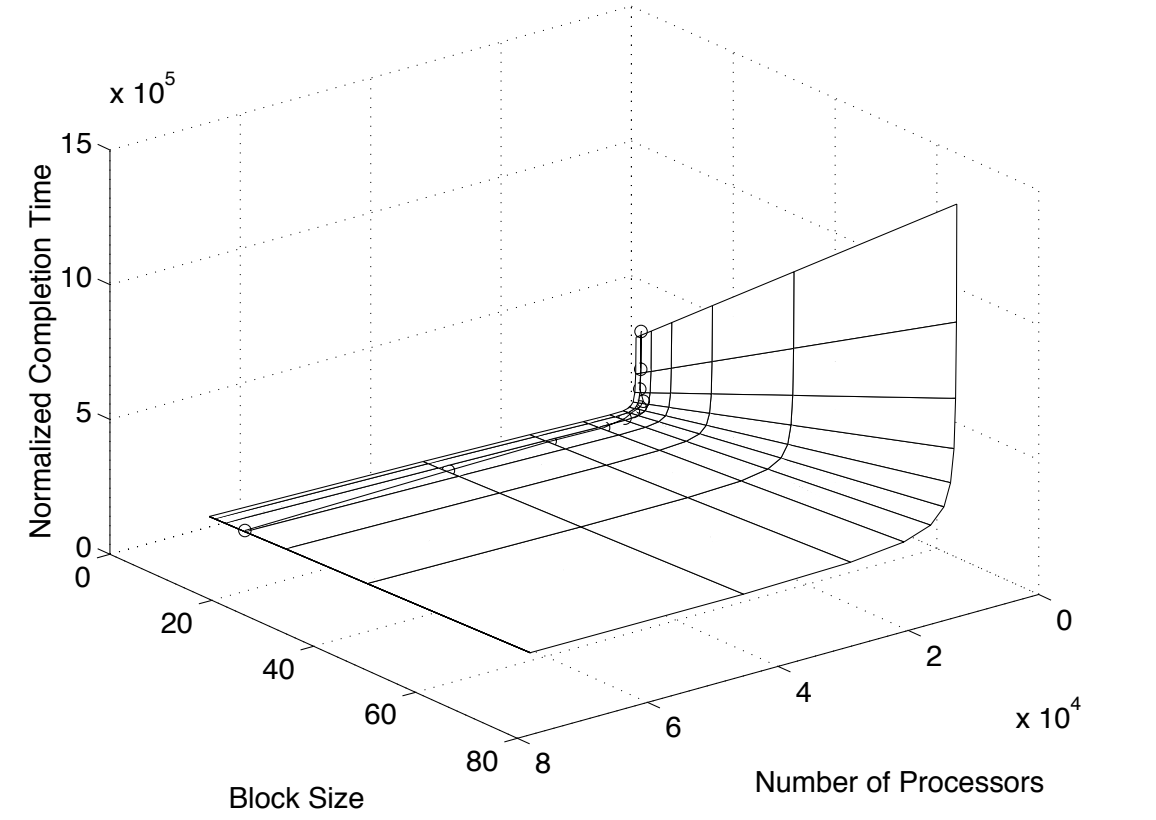


Fig. 3 Continued (h)

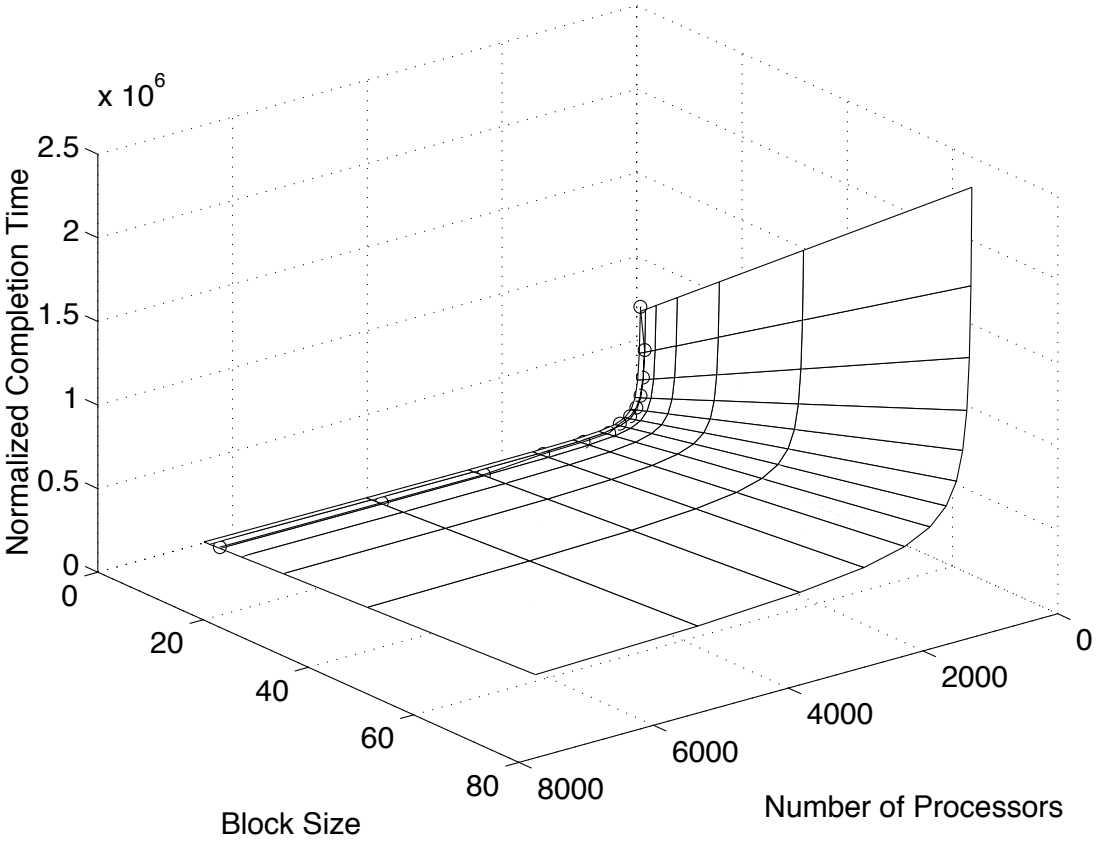


Fig. 3 Continued (i)

CHAPTER V

SWEEP DENSITY

The ρ function represents the number of distinct sweeps that must be computed on each processor during each step. We could arbitrarily set ρ equal to the number of distinct sweeps being computed, but that would not accurately reflect the situation. It is certainly not the case that every processor participates in each sweep at every step. Therefore, we should derive an expression for ρ that accounts for the asymmetry of the sweeps.

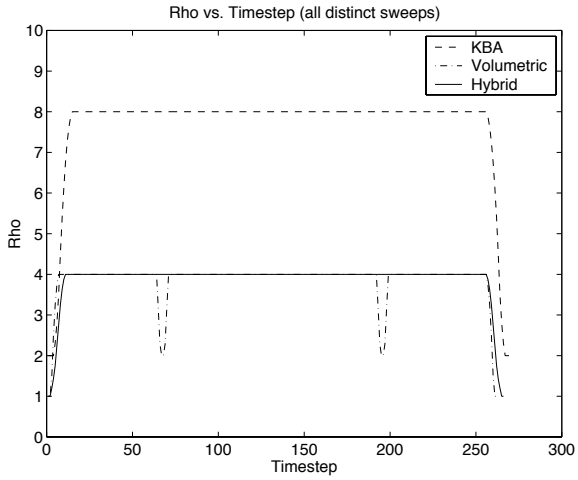


Fig. 4. Calculation of ρ. This plot shows values of ρ for the case of all distinct sweep directions, one for each octant of directions, in three dimensions. Here we see that KBA typically has a ρ equal to the number of directions, while Volumetric and Hybrid typically have ρ values equal to one half of the number of directions. This example assumes a representative grid of size $x = y = z = 256$ with $p = 64$ processors.

The exact value of ρ for any individual step can easily be determined by inspecting the grid. For a given step, ρ is precisely the maximum number of wavefronts crossing any processor's partition. We can use this method of inspection to calculate values of ρ for every step for each method. The results for a representative grid are displayed in Figure 4.

It is clear from these figures that ρ follows a regular and predictable pattern. Most of the time is spent on the plateau representing the steady-state operation of the sweep. The values of ρ that we use above in Equation 3.4 are the steady-state values. The remainder of the time is spent in either pipeline fill or empty stages. It is worth noting that the amount of time spent in these stages is less for Volumetric and Hybrid than for KBA. The startup time for a method is related to the sum of the dimensions of the coarse grid overlay, $\phi_x + \phi_y + \phi_z$. So, Volumetric should have the shortest startup time, KBA should have the longest, and Hybrid should fall somewhere in between. We see from Figure 4 that this is indeed the case. Note that Volumetric seems to have some difficulty keeping its pipeline full. These are the restarts mentioned in Chapter III that our model does not account for. For the validity of our results, it is important to note that KBA and Hybrid do not have any restarts. In particular, KBA does not have any restarts because it has only one horizontal partition. Hybrid does not either because, even though it has two horizontal partitions, all the processors are already busy by the time the boundary between the horizontal partitions is reached. The potential effects of these restarts will be explained further in Chapter VI.

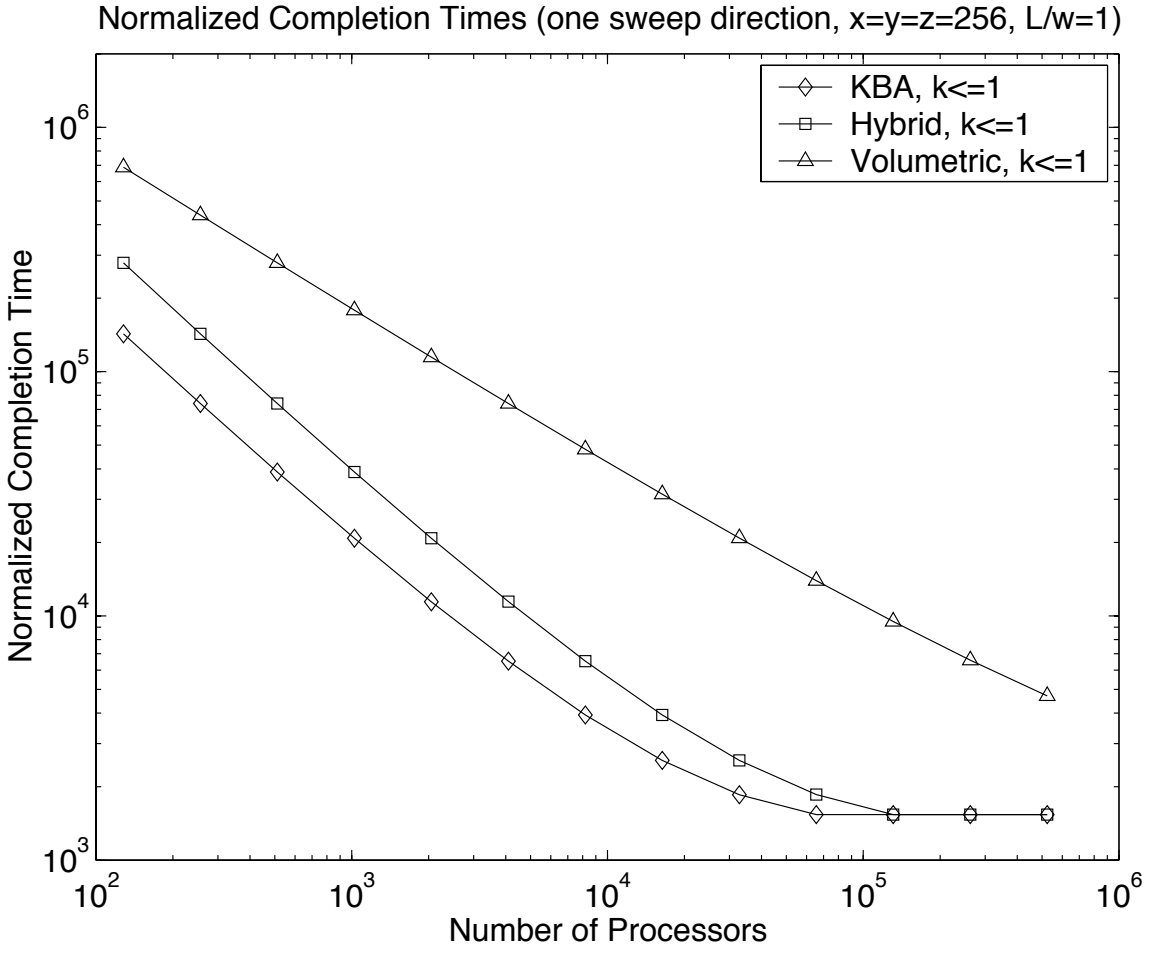
During steady-state operation, the ρ value for Volumetric and Hybrid is (generally) half that of KBA. The reason for this is that both of these methods subdivide the grid horizontally as well as vertically, while KBA does not. Therefore, wavefronts crossing the top of the grid will not interfere with wavefronts crossing the bottom of the grid, and ρ can be *at most* half the total number of wavefronts. Alternatively, since KBA does not subdivide the grid horizontally, ρ must be *at least* half the total number of wavefronts and is generally *equal* to the number of wavefronts.

CHAPTER VI

THEORETICAL RESULTS

Normalized completion times using Equation 3.7 for a variety of grid sizes, latencies, and number of directions are displayed in Figures 5-6. For these graphs we assume we have blocking communications ($\alpha = 0$). The behavior of the models with non-blocking communications ($\alpha \approx 1$) is similar. Each method is plotted out until it cannot use any more processors. After that point it is plotted using the maximum number of processors it can use. KBA can use up to $x \times y$ processors, Volumetric can use up to $x^2 \times y$ processors, and Hybrid can use up to $2 \times x \times y$ processors. In each scenario, k_{opt} is chosen individually for each number of processors using Equation 4.1. The k_{opt} values displayed next to the method names in the legend are the largest values of k_{opt} used by that method for any number of processors. We look first at a single sweep (for one octant of directions) and then at multiple sweeps (one for each octant of directions).

For one direction (Figure 5) KBA does very well, as we expect. Although the other methods usually surpass KBA for higher values of p, KBA does better over the lower, more reasonable values of p. This follows from Equation 3.7. Basically, since $\rho = 1$ for all methods, the advantage of having a lower ϕ_x and ϕ_y (which decreases the effect of the second and third terms) does not outweigh the advantage of having a higher ϕ_x and ϕ_y (which decreases the effect of the first term). Also notice that Hybrid and Volumetric catch up with KBA faster when L/ω is large (communication is relatively expensive). This also predicts that the cost of local computation, ω, would have a greater impact on KBA than the other methods.



(a)

Fig. 5. Sweeps in one direction only. These plots consider a sweep of a single direction in a cubic three–dimensional spatial domain ($x = y = z = 256$, ((a), (d), and (g)), $x = y = z = 128$, ((b), (e), and (f)), and $x = y = 128$, $z = 1024$, ((c), (f), and (i))) and study the relative performance of the three decomposition methods analyzed (KBA, Volumetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-c)), communication is moderately expensive ($L_c/\omega = 10$, (d-f)), and communication is relatively expensive ($L/\omega = 100$, (g-i)). All plots are for blocking communication ($\alpha = 0$).

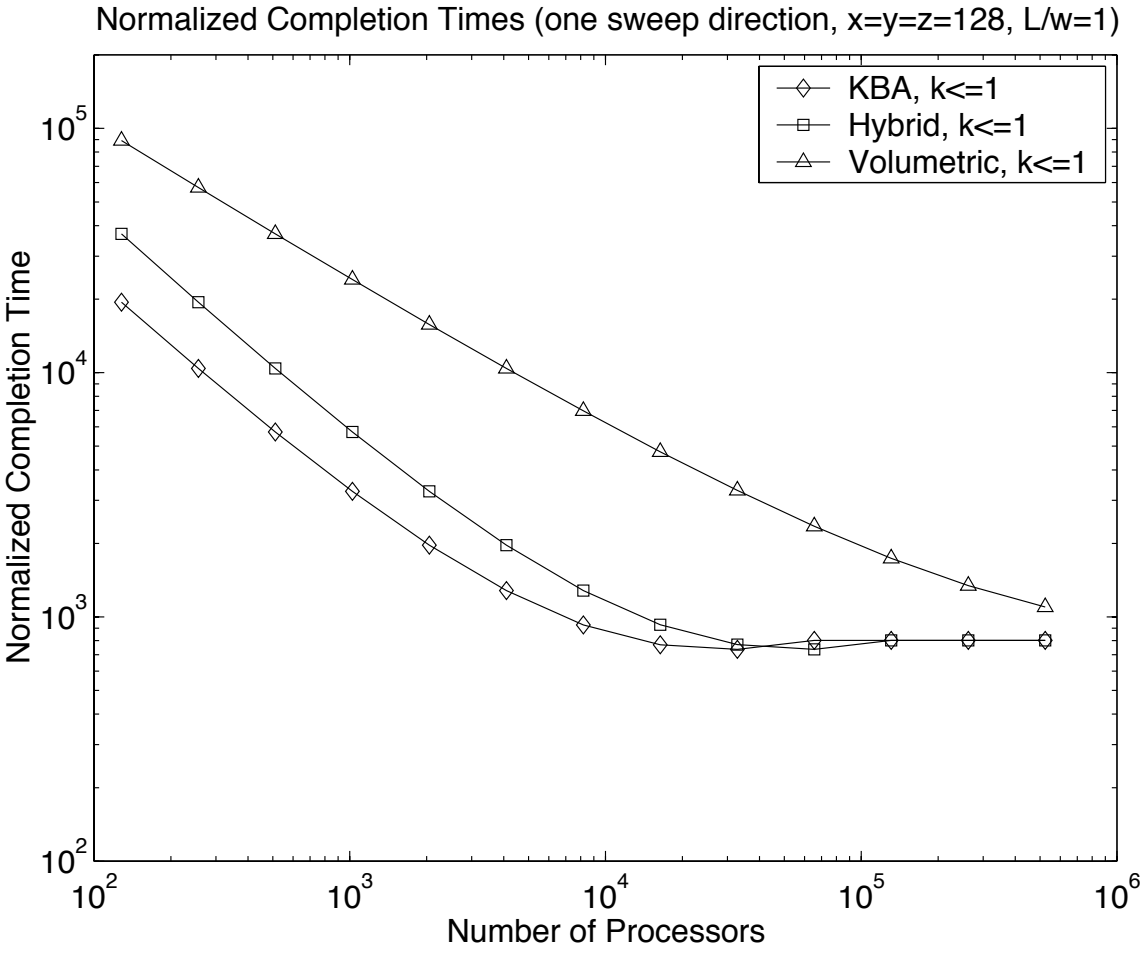


Fig. 5 Continued (b)

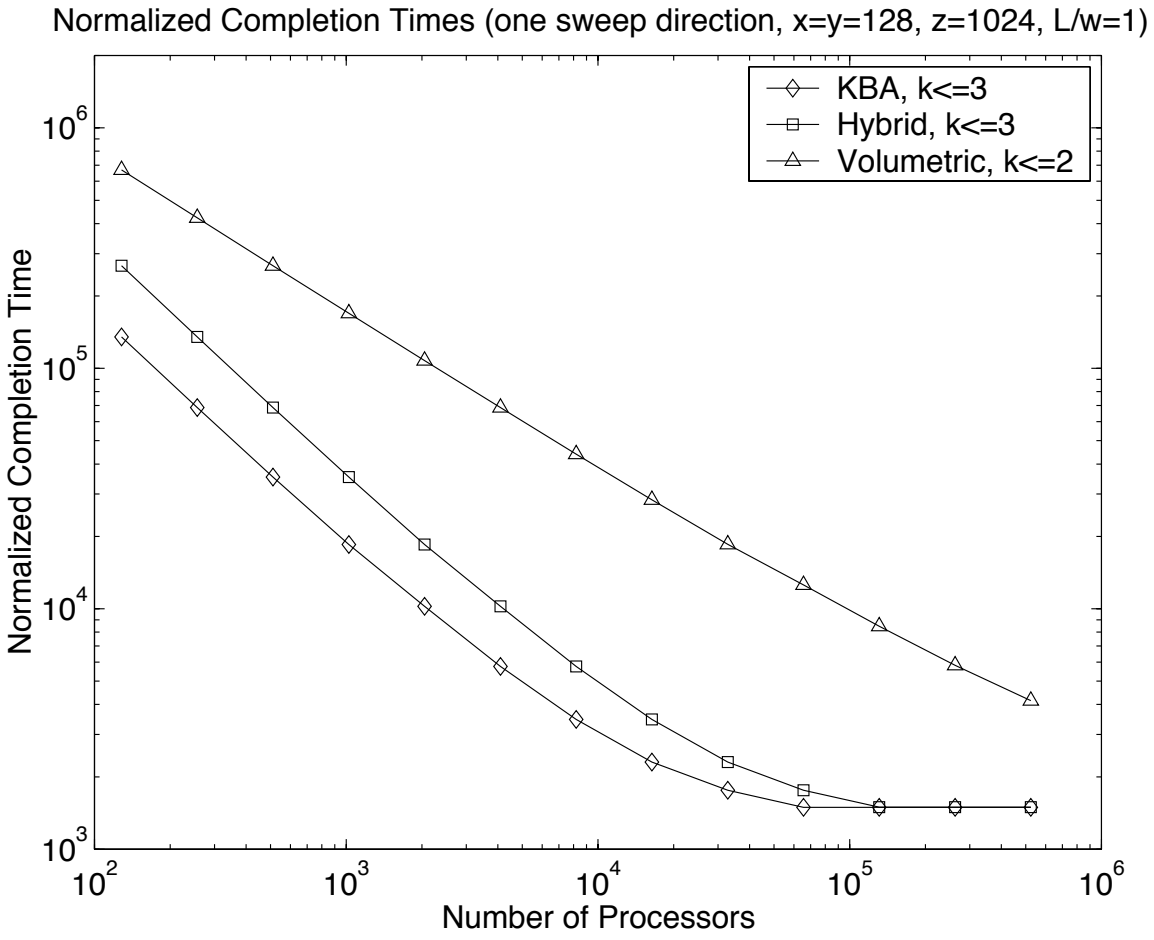


Fig. 5 Continued (c)

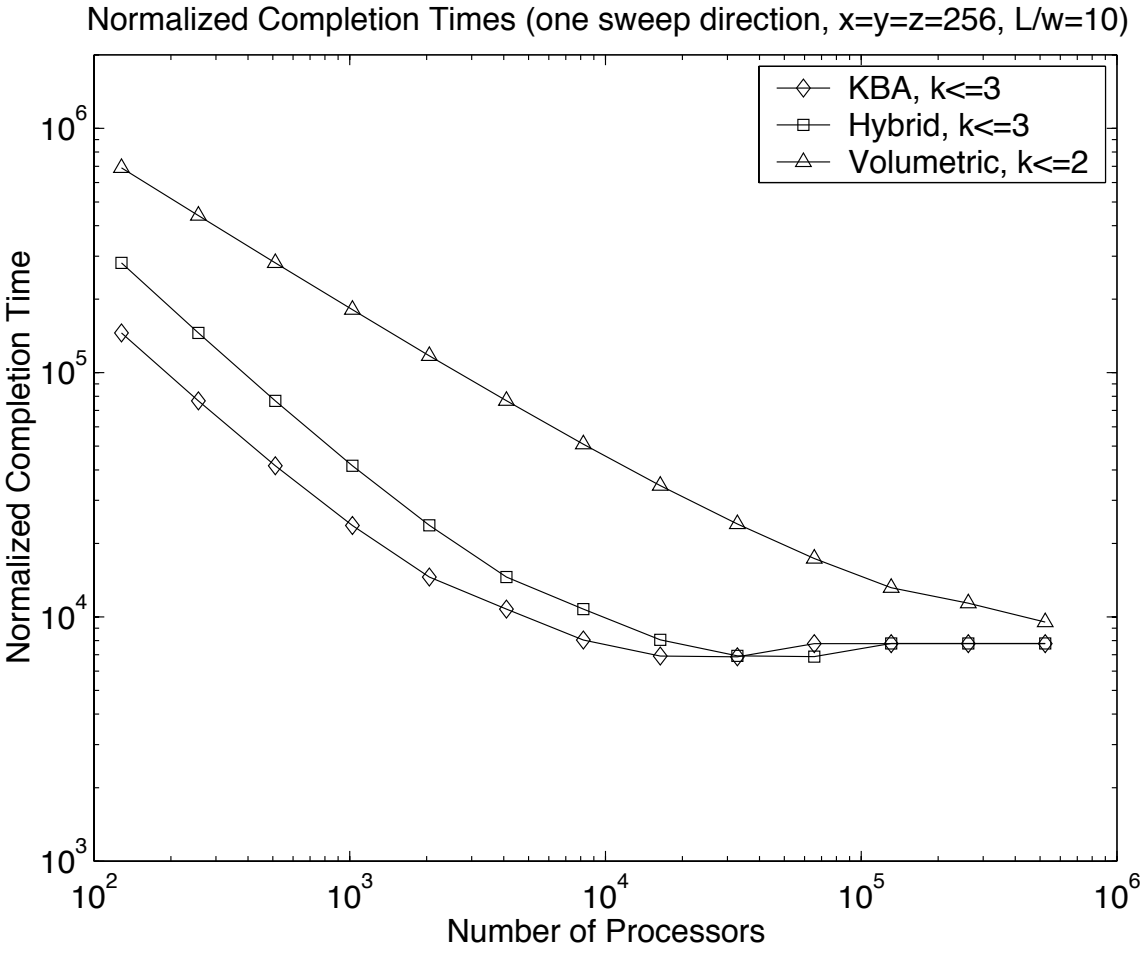


Fig. 5 Continued (d)

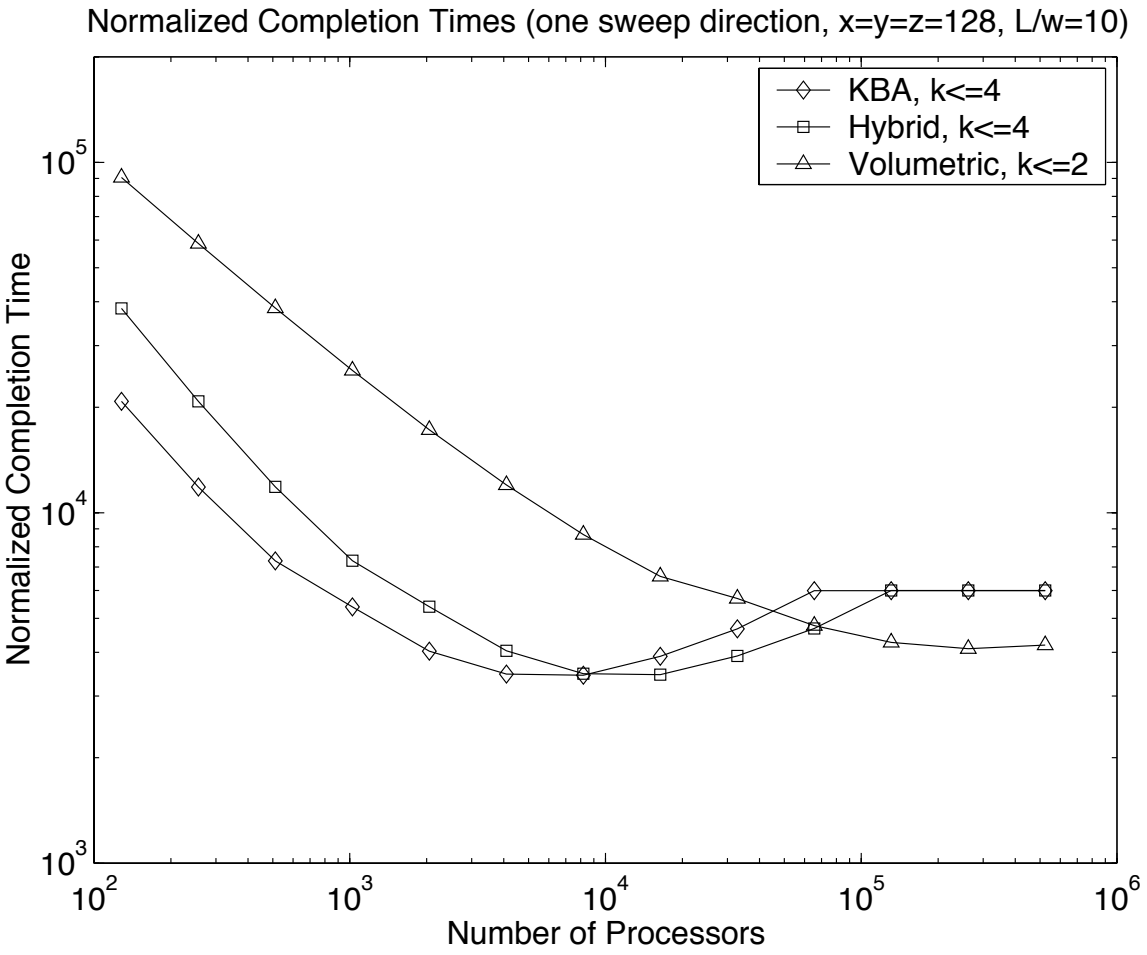


Fig. 5 Continued (e)

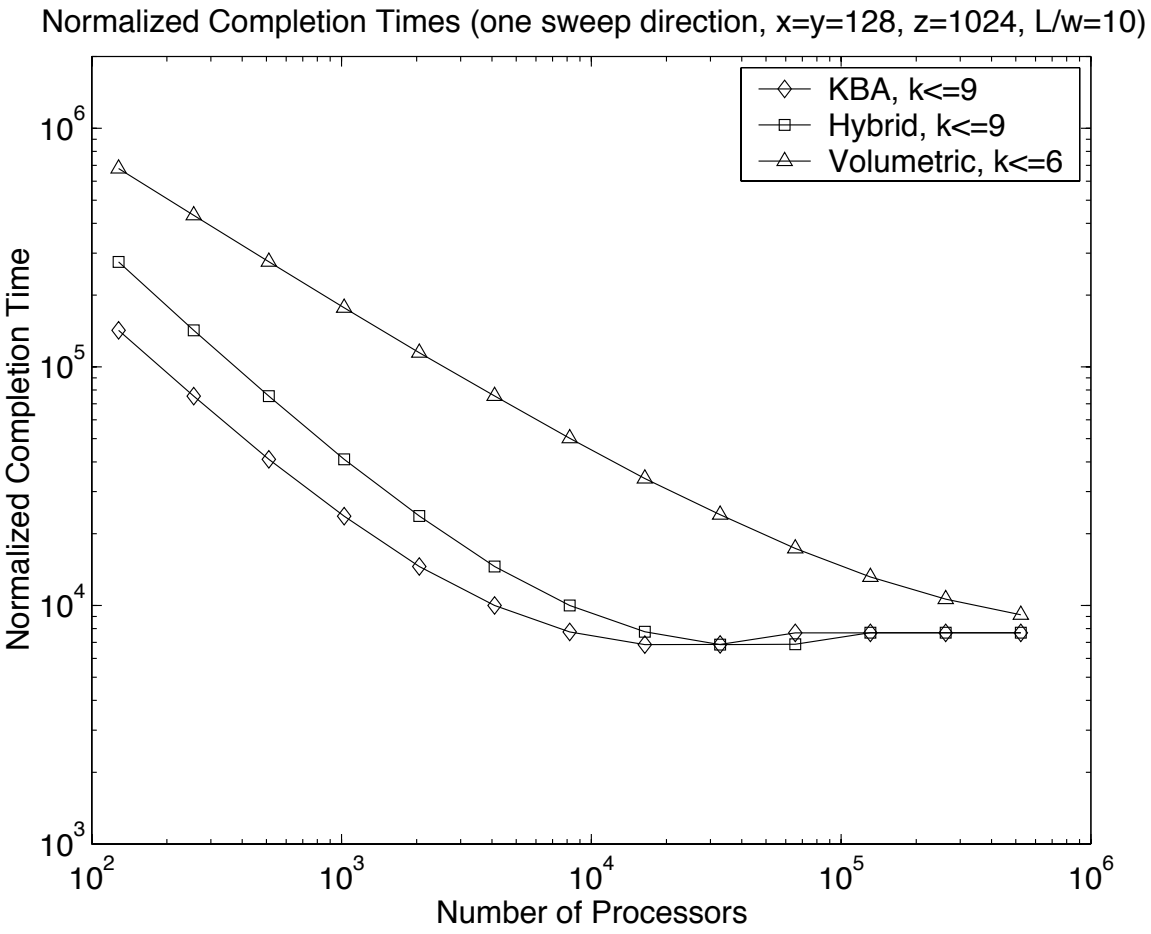


Fig. 5 Continued (f)

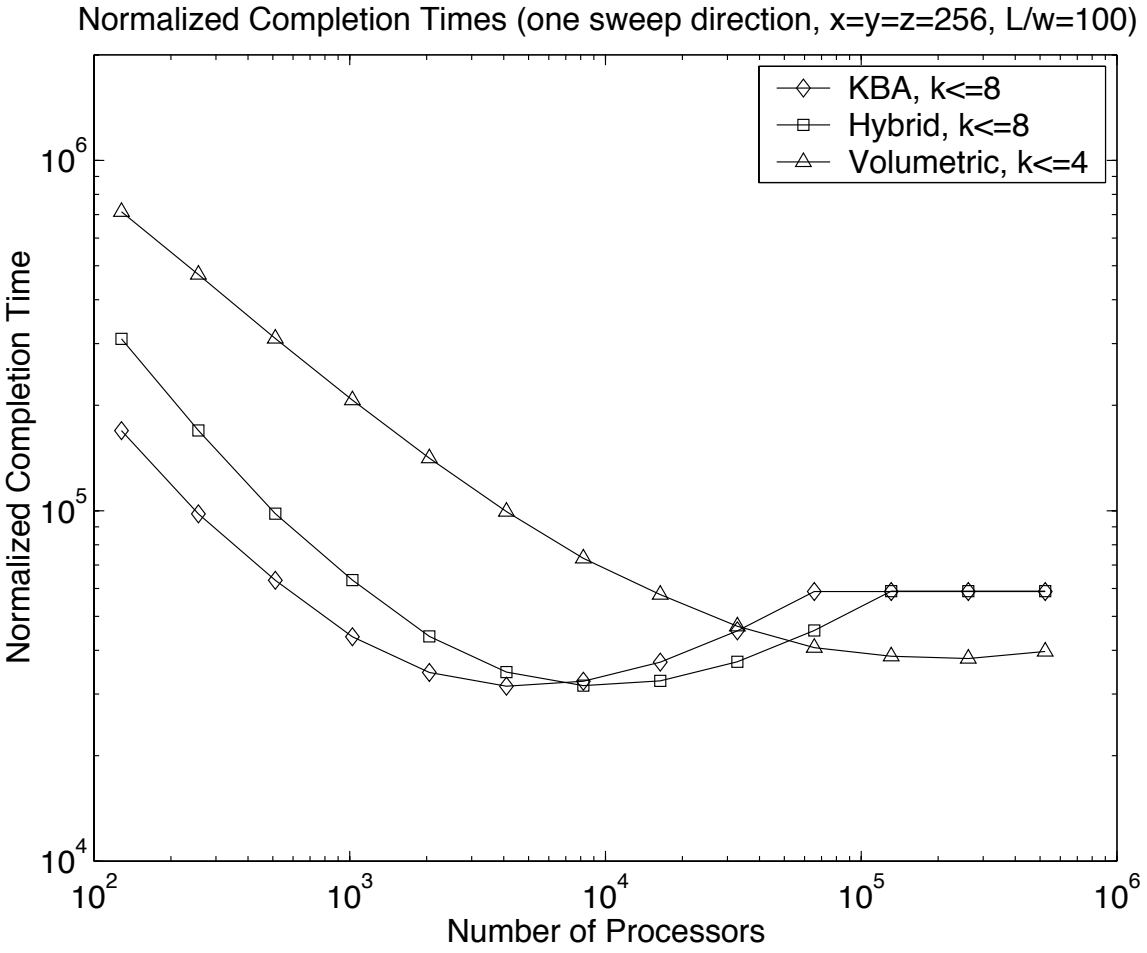


Fig. 5 Continued (g)

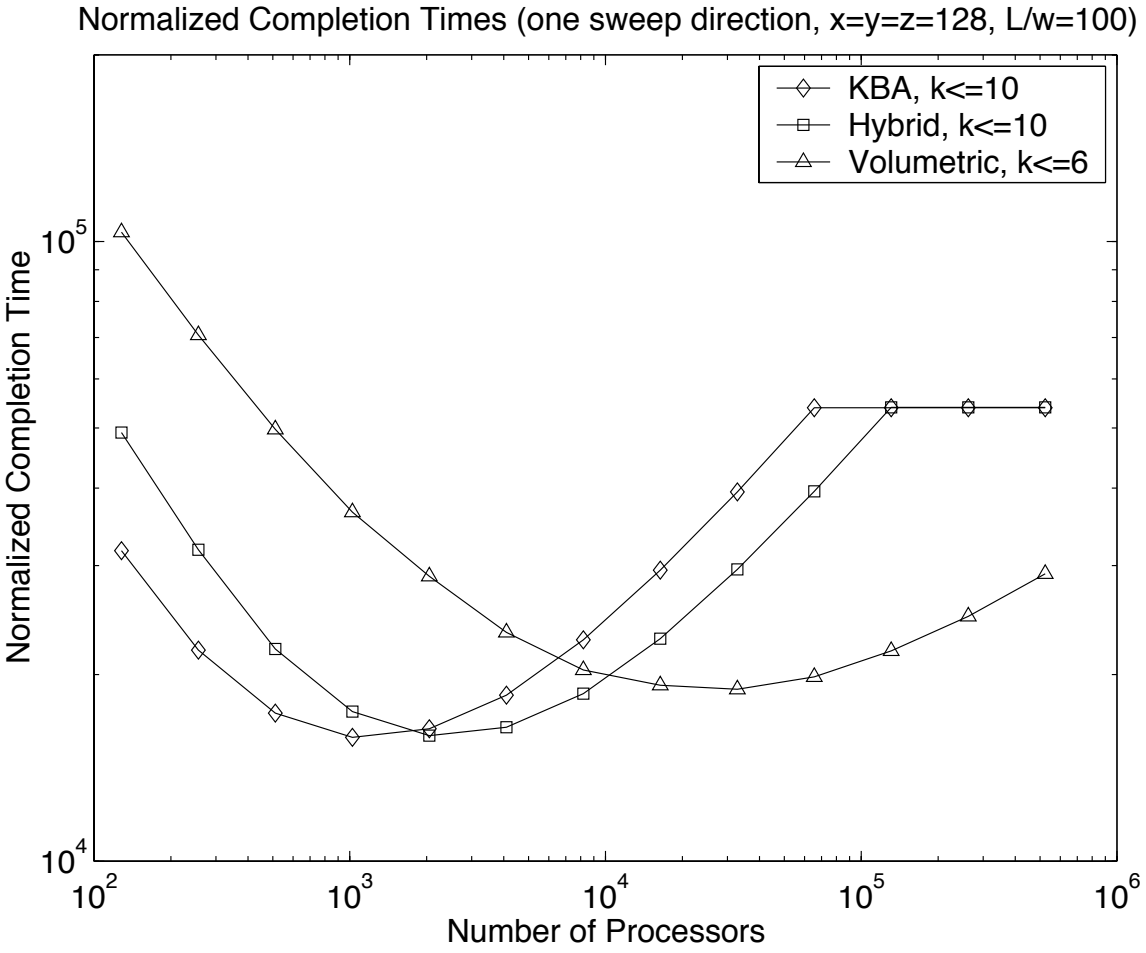


Fig. 5 Continued (h)

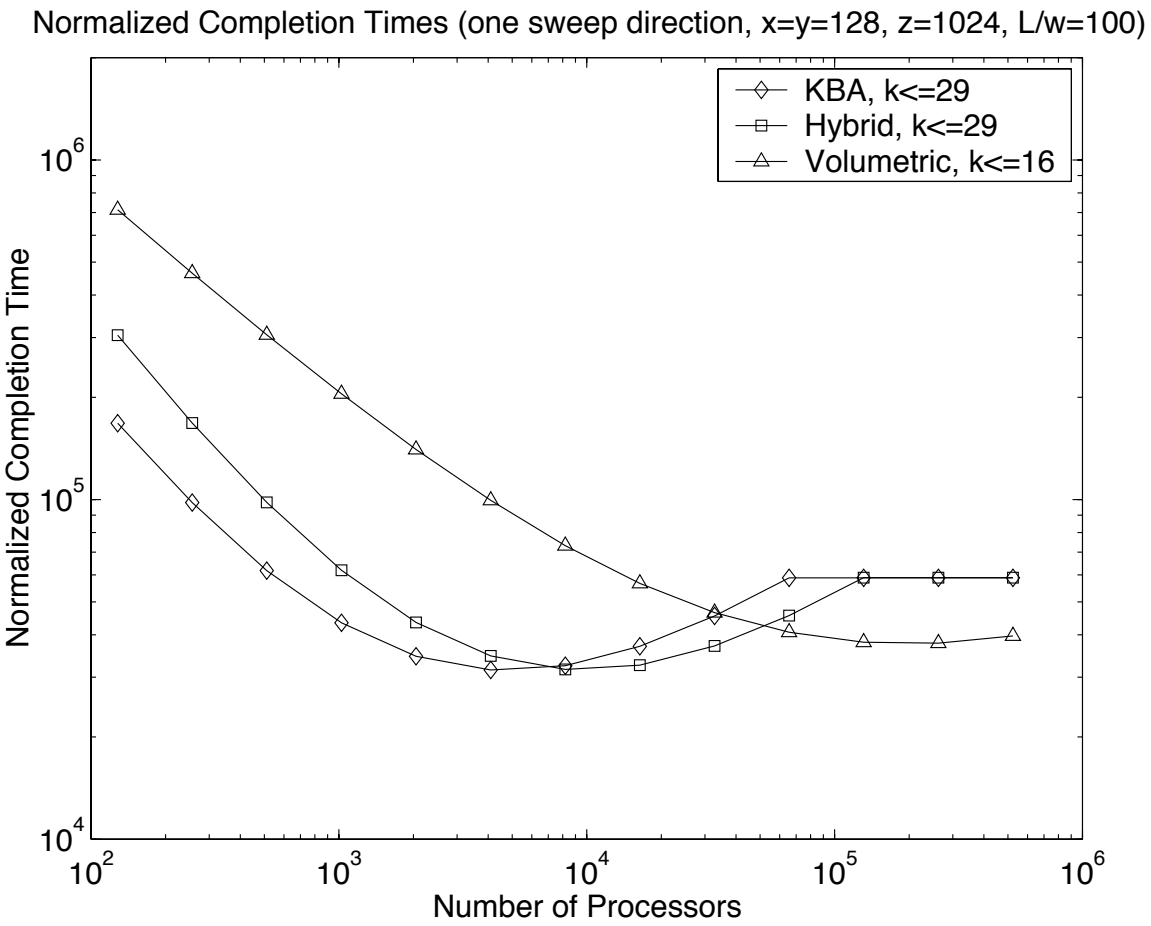
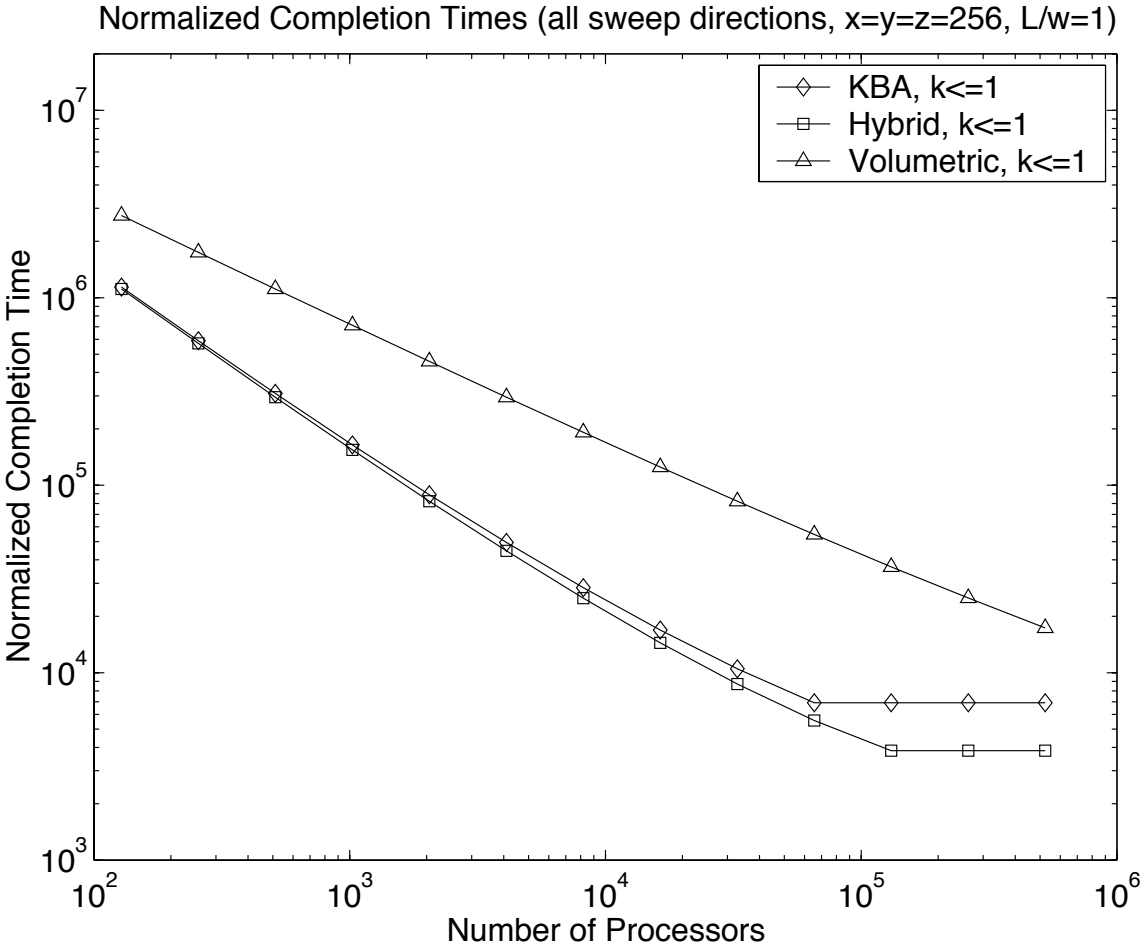


Fig. 5 Continued (i)



(a)

Fig. 6. Sweeps in all directions. These plots consider a sweep in all simultaneous directions in a cubic three–dimensional spatial domain ($x = y = z = 256$, ((a), (d), and (g)), $x = y = z = 128$, ((b), (e), and (f)), and $x = y = 128$, $z = 1024$, ((c), (f), and (i))) and study the relative performance of the three decomposition methods analyzed (KBA, Volumetric, and Hybrid) for three different cases of the machine parameter L/ω, representing machines where communication is relatively inexpensive ($L/\omega = 1$, (a-c)), communication is moderately expensive ($L/\omega = 10$, (d-f)), and communication is relatively expensive ($L/\omega = 100$, (g-i)). All plots are for blocking communication ($\alpha = 0$).

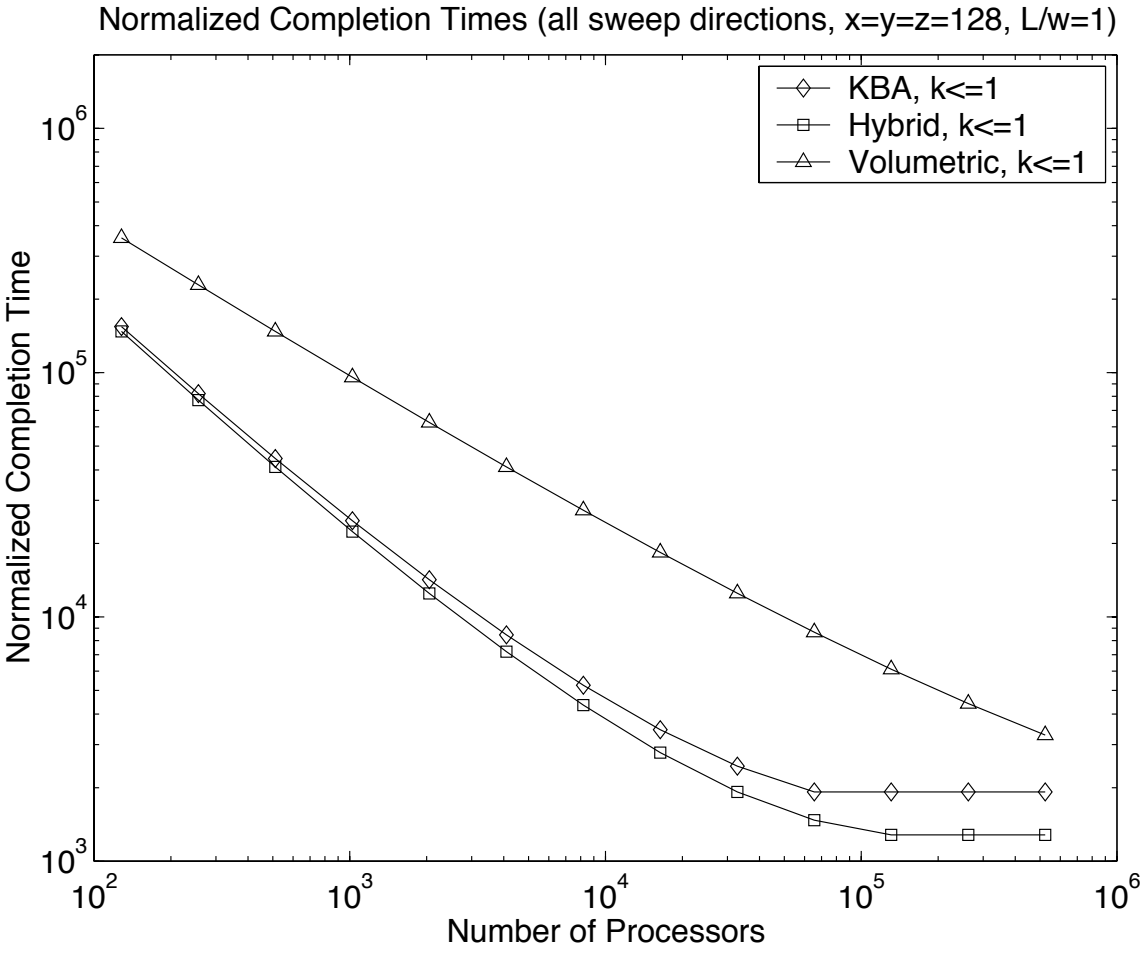


Fig. 6 Continued (b)

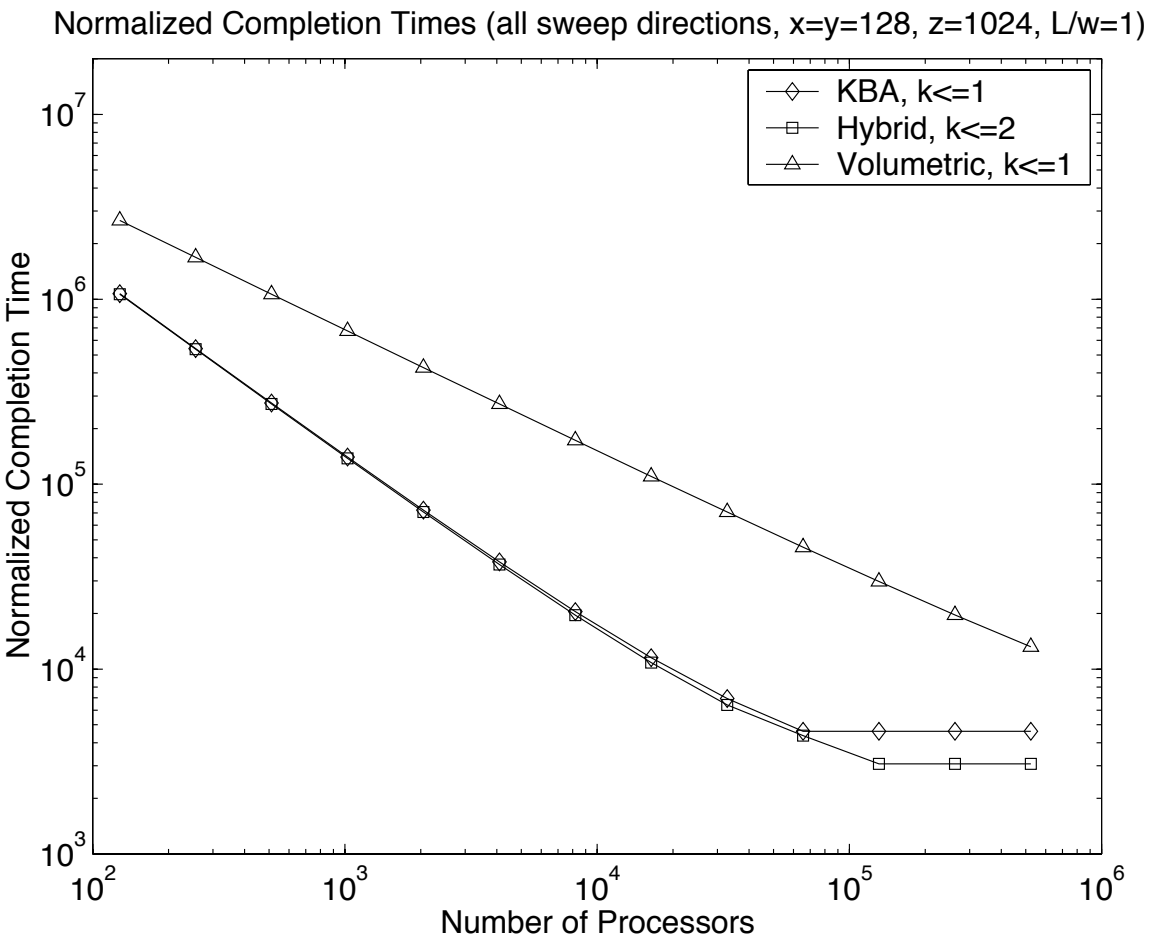


Fig. 6 Continued (c)

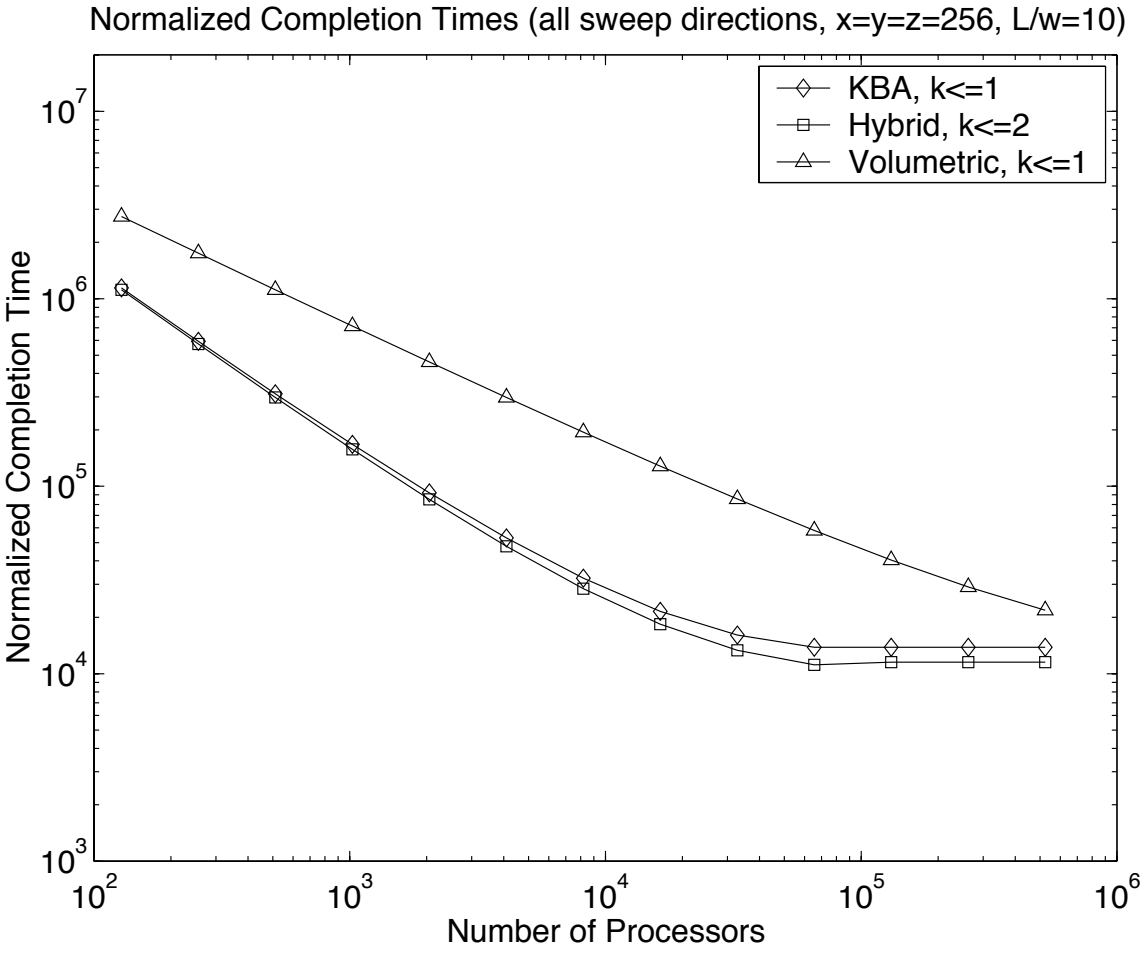


Fig. 6 Continued (d)

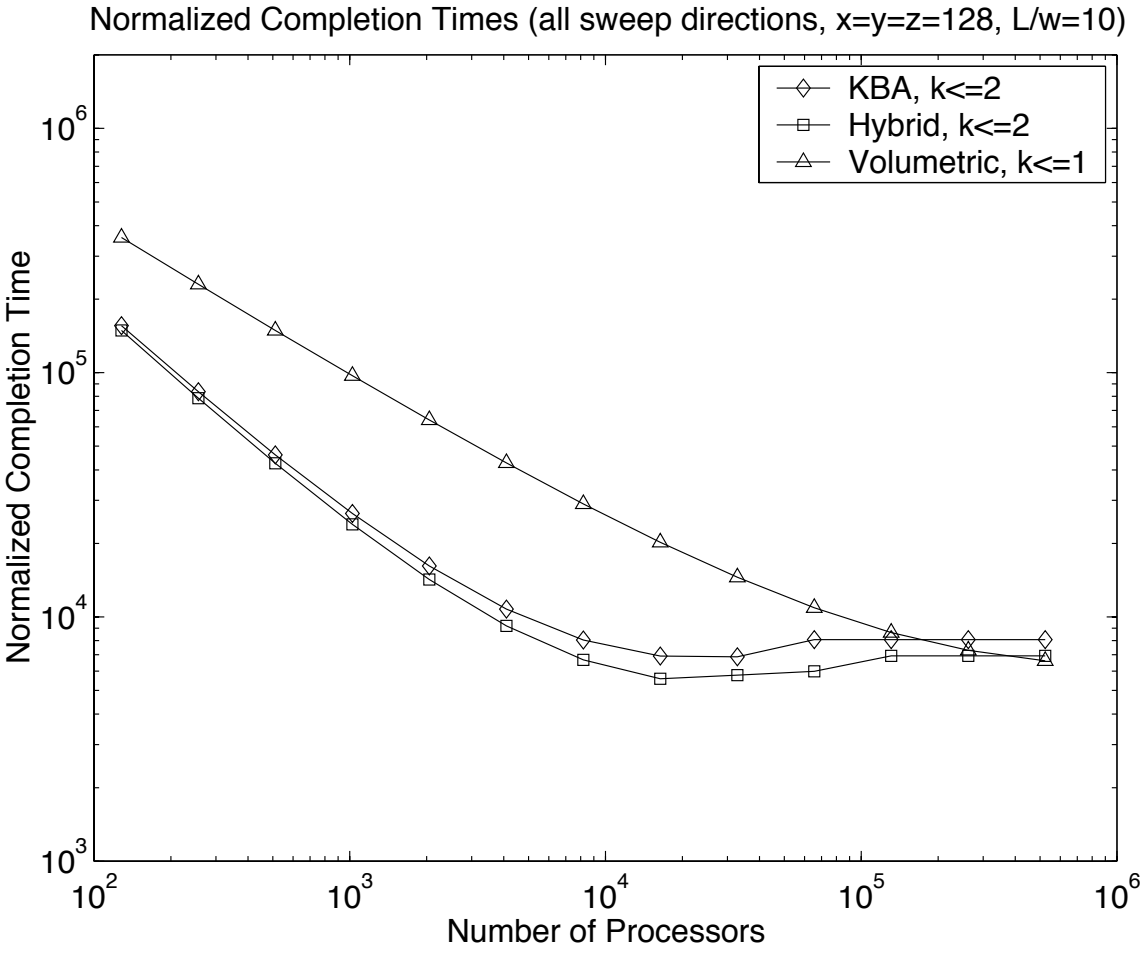


Fig. 6 Continued (e)

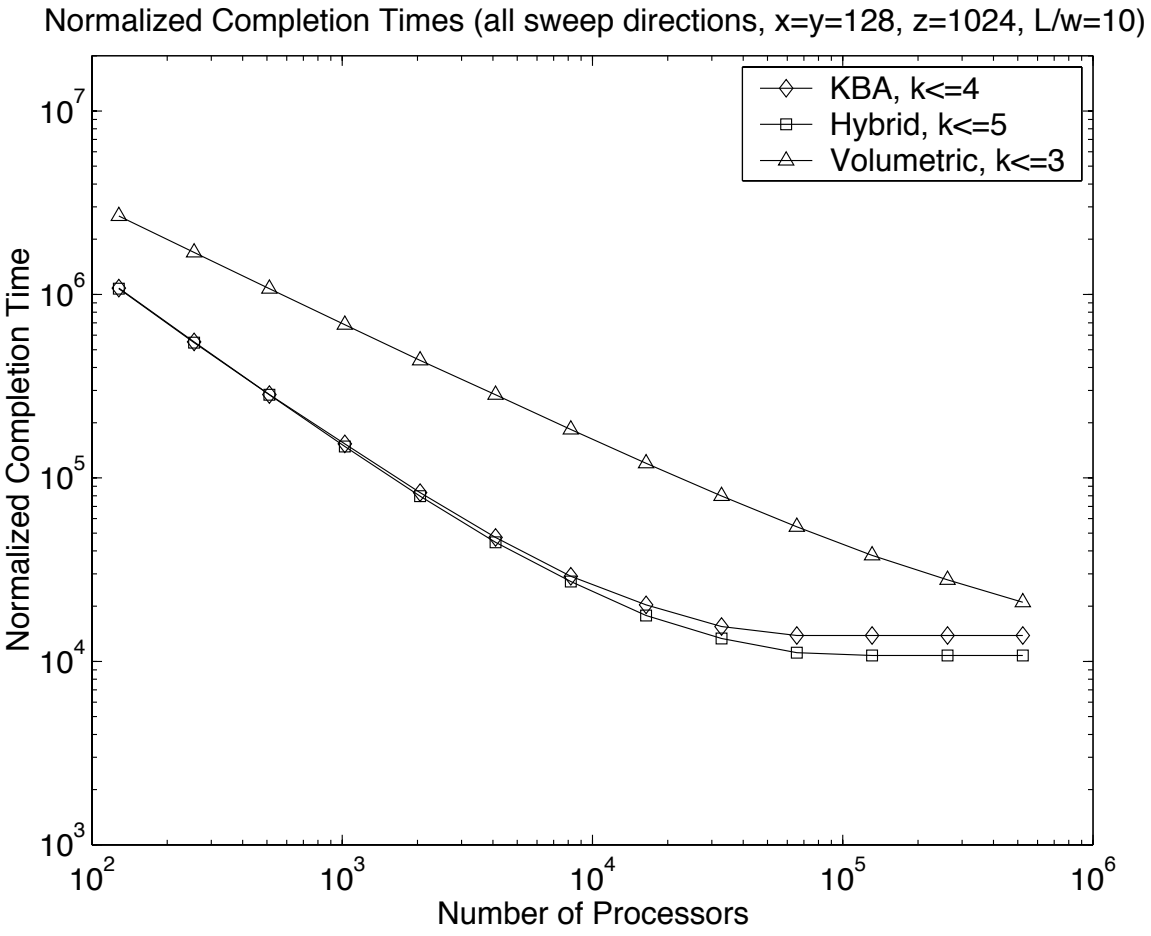


Fig. 6 Continued (f)

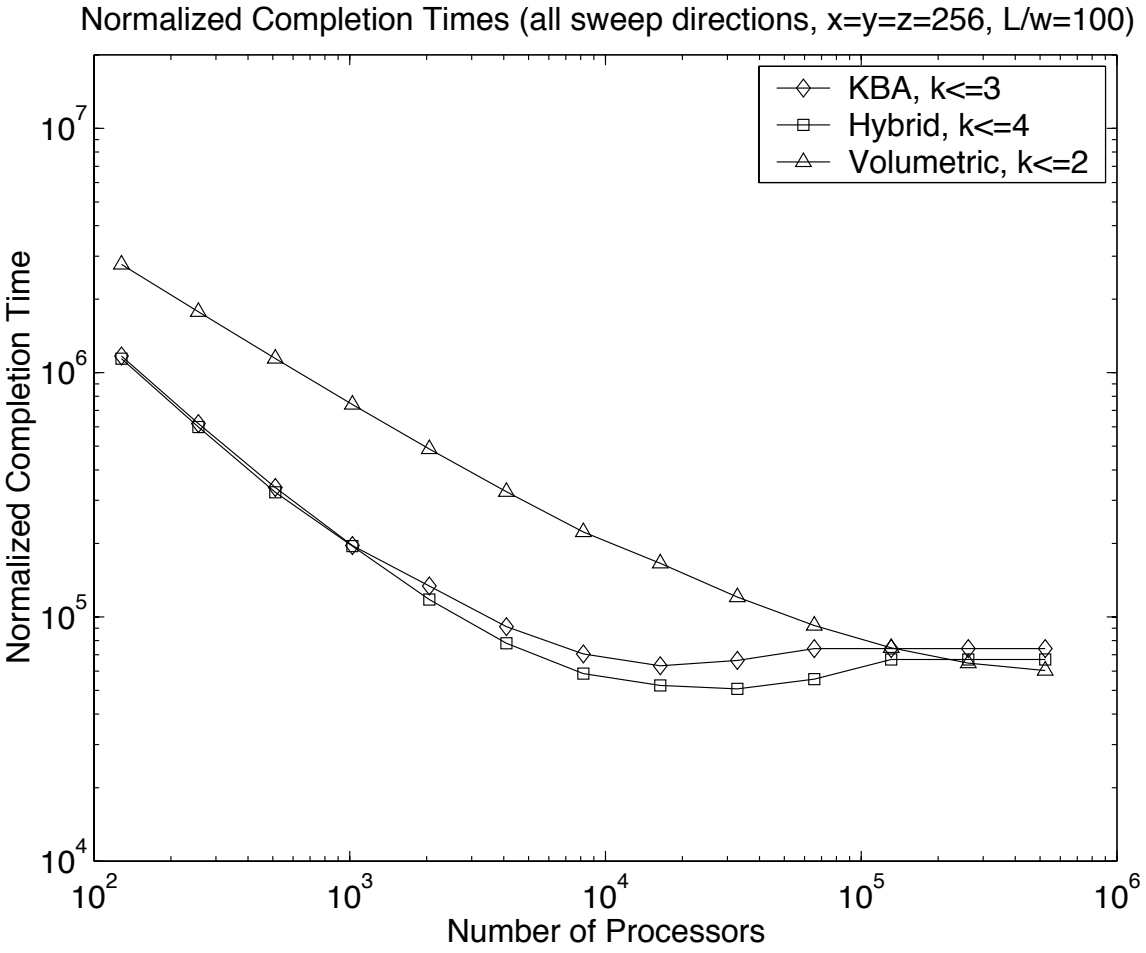


Fig. 6 Continued (g)

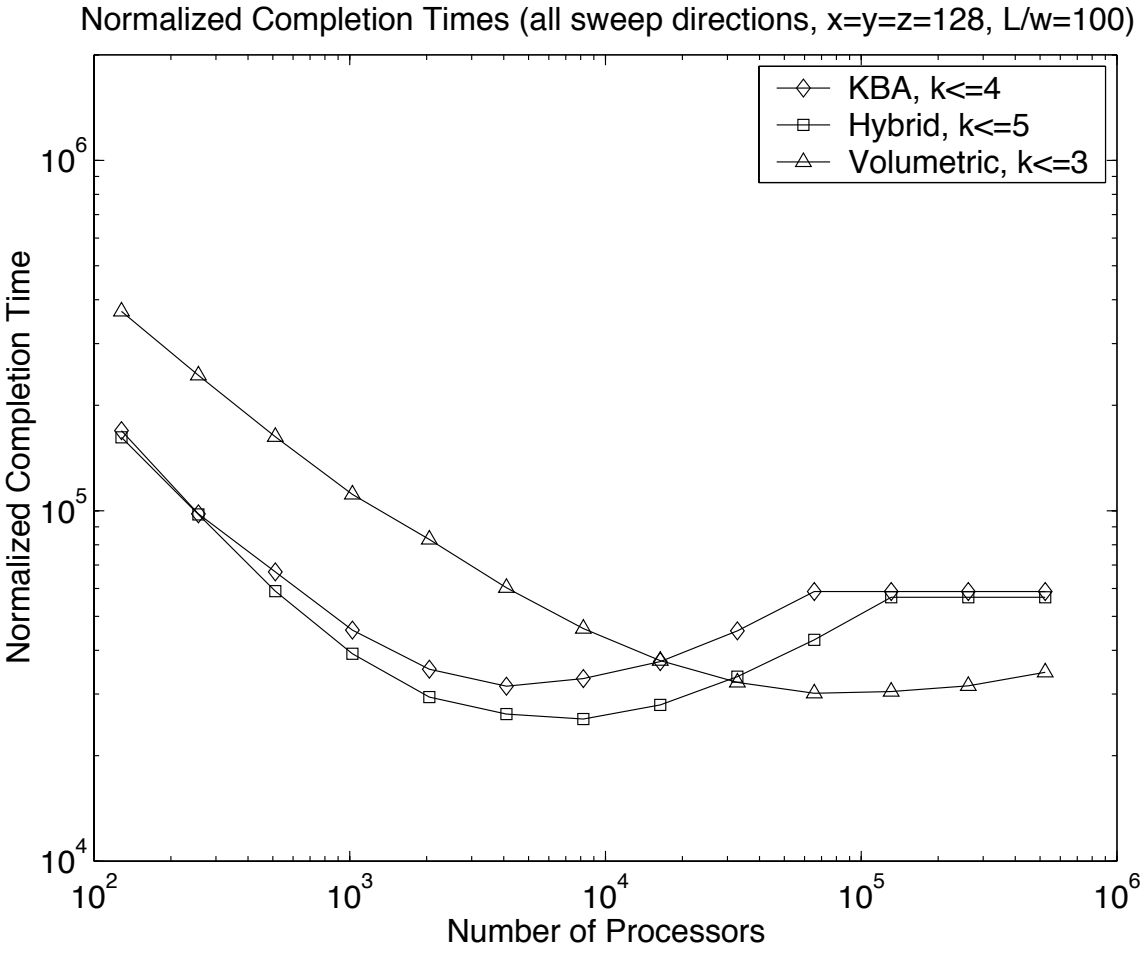


Fig. 6 Continued (h)

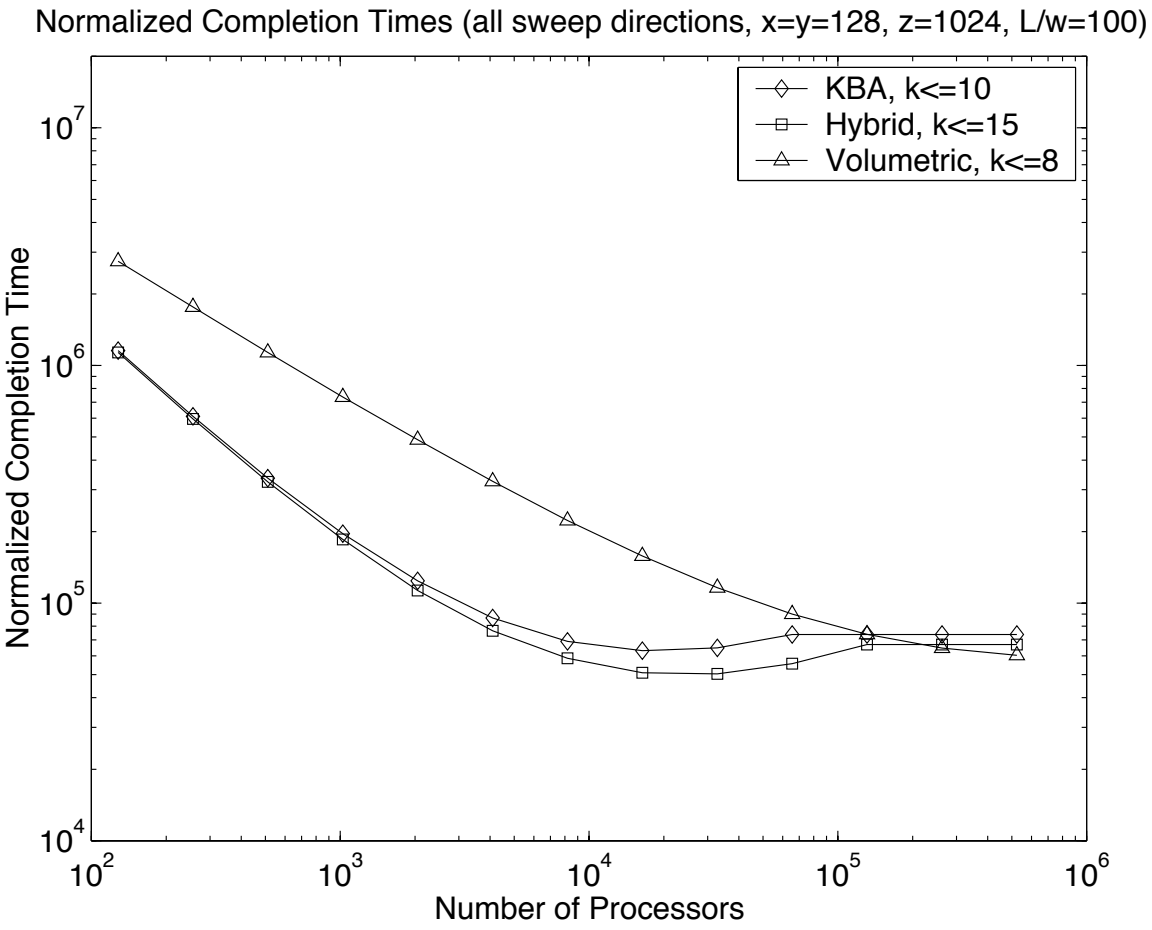


Fig. 6 Continued (i)

Table I. Summary of differences between KBA and Hybrid.

	ρ	ϕ_x	ϕ_y	ϕ_z
KBA	8	$\sqrt{p}$	$\sqrt{p}$	1
Hybrid	4	$\sqrt{p/2}$	$\sqrt{p/2}$	2

For multiple directions (Figure 6) the Hybrid method is always better than KBA. Now, having a higher ϕ_x and ϕ_y (which decreases the effect of the first term) does not pay off because it comes at the cost of a higher ρ. In this situation we see that the Hybrid and Volumetric methods do much better, although Volumetric is still not generally useful. One may wonder, "Does Hybrid always beat KBA in multiple directions?" To see that it does we will look at the ratio of the models for KBA over Hybrid. If the ratio is greater than one, then Hybrid will always outperform KBA.

In order to compare the models for the two different methods we must be able to express one in terms of the other. Table I summarizes the differences between the parameters of KBA and Hybrid. Fortunately, ϕ_x, ϕ_y, and ϕ_z for KBA can be easily expressed in terms of Hybrid's parameters.

$$\phi_x^{KBA} = \sqrt{2}\phi_x^{Hybrid}$$
$$\phi_y^{KBA} = \sqrt{2}\phi_y^{Hybrid}$$
$$\phi_z^{KBA} = \frac{1}{2}\phi_z^{Hybrid}$$

We also need to express k_{opt} for one method in terms of k_{opt} for the other method.

This can be accomplished using Equation 4.1. Note that ϕ_x, ϕ_y, and ϕ_z are for Hybrid.

$$\begin{aligned}\frac{k_{opt}^{KBA}}{k_{opt}^{Hybrid}} &= \frac{\sqrt{(\frac{(1-\alpha)(L/\omega)z}{8xy})(\frac{\sqrt{2}\phi_x\sqrt{2}\phi_y}{\sqrt{2}\phi_x+\sqrt{2}\phi_y})}}{\sqrt{(\frac{(1-\alpha)(L/\omega)z}{4xy})(\frac{\phi_x\phi_y}{\phi_x+\phi_y})}} \\ &= \frac{1}{2^{1/4}}\end{aligned}$$

Therefore,

$$k_{opt}^{KBA} = \frac{1}{2^{1/4}} k_{opt}^{Hybrid}$$

Now we can begin to look at the ratio of the two models. Since there are three major terms in Equation 3.7, we will look at each term individually. If the ratios for each term are greater then one, then the ratio for all three terms together would also be greater than one. Note that in addition to ϕ_x, ϕ_y, and ϕ_z, k_{opt} is for Hybrid as well.

First Term:

$$\begin{aligned}\frac{T_{KBA}^1}{T_{Hybrid}^1} &= \frac{8xy(\frac{z}{2\phi_x\phi_y}+\frac{k_{opt}}{2^{3/4}\phi_x}+\frac{k_{opt}}{2^{3/4}\phi_y})}{4xy(\frac{z}{\phi_x\phi_y}+\frac{k_{opt}}{\phi_x}+\frac{k_{opt}}{\phi_y})} \\ &= \frac{(\frac{z}{\phi_x\phi_y}+\frac{2^{1/4}k_{opt}}{\phi_x}+\frac{2^{1/4}k_{opt}}{\phi_y})}{(\frac{z}{\phi_x\phi_y}+\frac{k_{opt}}{\phi_x}+\frac{k_{opt}}{\phi_y})} \\ &> 1\end{aligned} \tag{6.1}$$

Second Term:

$$\begin{aligned}\frac{T_{KBA}^2}{T_{Hybrid}^2} &= \frac{(1-\alpha)\frac{L}{\omega}(\frac{2^{1/4}z}{k_{opt}})}{(1-\alpha)\frac{L}{\omega}(\frac{z}{k_{opt}})} \\ &= 2^{1/4} \\ &> 1\end{aligned} \tag{6.2}$$

Third Term:

$$\begin{aligned}\frac{T^3_{KBA}}{T^3_{Hybrid}} &= \frac{L/\omega(\sqrt{2}\phi_x + \sqrt{2}\phi_y + \frac{1}{2}\phi_z)}{L/\omega(\phi_x + \phi_y + \phi_z)} \\ &= \frac{(\sqrt{2}\phi_x + \sqrt{2}\phi_y + \frac{1}{2}\phi_z)}{(\phi_x + \phi_y + \phi_z)}\end{aligned} \tag{6.3}$$

which is greater than one if $\phi_x + \phi_y > \frac{1}{\sqrt{2}-1} \approx 2.4$. This is true for all but the most trivial examples (i.e. $p \leq 2$). Therefore, Hybrid will always perform better than KBA for multiple simultaneous sweeps.

We now return to the issue of the extra communication or 'restarts' mentioned in Chapter III. For KBA there is only one horizontal partition, $\phi_z = 1$, so the model is correct for KBA in either the single sweep case or the multiple simultaneous sweeps case. We see in the single sweep case that KBA usually does the best. This is in spite of the fact that Hybrid and Volumetric actually benefit from the omission of 'restarts' in our model. Therefore, if we were to include restarts, the relative performance of the different partitioning methods would not change. KBA would perform the same and Hybrid and Volumetric would only get worse.

In the multiple simultaneous sweeps case we see that Hybrid is always better than KBA. Does this result still hold if we count restarts in our model? Recall that we developed our model for the case of multiple simultaneous sweeps. In this case Hybrid does not have any restarts because the processors on the top (bottom) of the grid are already busy when the processors on the bottom (top) of the grid are ready to communicate data with them. Therefore, the model is accurate for KBA and Hybrid in the multiple directions case and the relationship between the two methods still holds. Volumetric on the other hand may have some restarts even in the multiple simultaneous sweeps case. However, Volumetric is not competitive with the other methods and including restarts would only serve to decrease its performance.

CHAPTER VII

CONCLUSION

The key contribution of this thesis is a new general model which can be used to predict the running time of parallel sweeps on orthogonal grids for any regular mapping of the grid cells to processors. In particular, our model accounts for machine-dependent parameters such as computation and communication/latency costs (assumed constant for all grid cells) and is parameterized by p, the number of processors, (x, y, z), the dimensions of the underlying spatial transport grid, and (ϕ_x, ϕ_y, ϕ_z), the dimensions of the coarse grid processor overlay (which determines the dimensions of the sub-domain assigned to each processor). Thus, our model can be used to analyze and compare the effects of various spatial decompositions on the running time of the transport sweep.

Insight obtained from the model yields the following contributions to the theory of optimal transport sweeps on orthogonal grids.

- Our model provides a theoretical basis that explains why, and under what circumstances, the column decomposition of the current standard KBA algorithm [Koch et al. 1992] is superior to the 'balanced' decomposition' obtained by classic domain decomposition techniques.

- Our model enables us to identify a new decomposition that proves to be almost as good as and often better than the current standard KBA method. We call this new decomposition the *Hybrid* method because it incorporates positive aspects of both the KBA and the balanced decomposition.

- A more minor (but still potentially valuable) contribution of our work is a theoretical expression for the optimal 'block size' parameter in the sweep method (the number of cells a processor should process before communicating).

The methods we have analyzed represent a family of algorithms for three dimensional sweeps. Given an input grid and a parallel computer system (i.e., number of processors and estimates of computation and communication costs for each grid cell), one can select the best method for the given configuration. Furthermore, one can minimize the completion time by selecting an optimal block size. Future work on this subject will involve correctly accounting for restarts in the execution of the transport sweep.

REFERENCES

Baker, R. S. and Alcouffe, R. E. 1997. Parallel 3D S_N Performance for DANTSYS/MPI on the Cray T3D. In *Proc. Int. Conf. on Mathematical Methods and Supercomputing for Nuclear Applications*, Volume 1 (Oct. 1997), pp. 377–393.

Hoisie, A., Lubeck, O., and Wasserman, H. 1998. Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications. Technical Report LAUR-98-3316 (Aug.), Los Alamos National Laboratory.

Hoisie, A., Lubeck, O., and Wasserman, H. 1999. Scalability analysis of multidimensional wavefront algorithms on large-scale SMP clusters. In *Proceedings of Frontiers '99: The 7th Symposium on the Frontiers of Massively Parallel Computation* (Annapolis, MD, Feb. 1999), pp. 4–15. IEEE Computer Society.

Hoisie, A., Lubeck, O., Wasserman, H., Petrini, F., and Alme, H. 2000. A general predictive performance model for wavefront algorithms on clusters of smps. In *Proceedings of ICPP 2000* (Toronto, Canada, Aug. 2000), pp. 20–25.

Karypis, G. and Kumar, V. 1995. METIS, unstructured graph partitioning and sparse matrix ordering system. version 2.0. Technical report (Aug.), University of Minnesota, Department of Computer Science, Minneapolis, MN.

Koch, K. R., Baker, R. S., and Alcouffe, R. E. 1992. Solution of the first-order form of the 3D discrete ordinates equation on a massively parallel processor. *Transactions of the American Nuclear Society 65*, 198–199.

Lewis, E. E. and Miller, W. F. 1993. *Computational Methods of Neutron*

Transport. American Nuclear Society, LaGrange Park, IL.

Mathis, M. M., Amato, N. M., and Adams, M. L. 2000. A general performance model for parallel sweeps on orthogonal grids for particle transport calculations. In *Proc. ACM Int. Conf. Supercomputing (ICS)* (May 2000), pp. 255–263.

Owens, J. 1999. *The ASCI SWEEP3D README File*. On the web at www.llnl.gov/asci_benchmarks/asci/limited/sweep3d/sweep3d_readme.html: Lawrence Livermore National Laboratories.

Valiant, L. 1990. Bridging model for parallel computation. *Comm. ACM 33*, 8, 103–111.

VITA

Mark Michael Mathis was born August 31st, 1976, in Fort Worth, Texas to Joe Michael and Emily Jane Mathis. He graduated from Borger High School in 1994. He received his B.S. in Computer Engineering at Texas A&M University in 1999. Mr. Mathis's permanent address is 2306 Yellowstone, Bryan, TX, 77803. He can also be reached via electronic mail at mmathis@cs.tamu.edu. More information on his past and present work can be found at his home page available at http://www.cs.tamu.edu/people/mmathis/.