International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

SCALABLE PARALLEL PREFIX SOLVERS FOR DISCRETE ORDINATES TRANSPORT

Shawn Pautz[*]
Sandia National Laboratories[†]
Albuquerque, NM 87185-1179
sdpautz@sandia.gov

Tara Pandya and Marvin Adams
Department of Nuclear Engineering
Texas A&M University
College Station, TX 77843-3133
tarapandya@tamu.edu; mladams@tamu.edu

ABSTRACT

The well-known "sweep" algorithm for inverting the streaming-plus-collision term in first-order deterministic radiation transport calculations has some desirable numerical properties. However, it suffers from parallel scaling issues caused by a lack of concurrency. The maximum degree of concurrency, and thus the maximum parallelism, grows more slowly than the problem size for sweeps-based solvers.

We investigate a new class of parallel algorithms that involves recasting the streaming-plus-collision problem in prefix form and solving via cyclic reduction. This method, although computationally more expensive at low levels of parallelism than the sweep algorithm, offers better theoretical scalability properties. Previous work has demonstrated this approach for one-dimensional calculations; we show how to extend it to multidimensional calculations. Notably, for multiple dimensions it appears that this approach is limited to long-characteristics discretizations; other discretizations cannot be cast in prefix form.

We implement two variants of the algorithm within the radlib/SCEPTRE transport code library at Sandia National Laboratories and show results on two different massively parallel systems. Both the "forward" and "symmetric" solvers behave similarly, scaling well to larger degrees of parallelism then sweeps-based solvers. We do observe some issues at the highest levels of parallelism (relative to the system size) and discuss possible causes. We conclude that this approach shows good potential for future parallel systems, but the parallel scalability will depend heavily on the architecture of the communication networks of these systems.

Key Words: deterministic transport, sweeps, cyclic reduction, parallel prefix, scalability

---

[*] To whom correspondence should be addressed

[†] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

1. INTRODUCTION

Numerical solutions to discrete ordinates discretizations of the first-order form of the Boltzmann transport equation are commonly obtained by means of the method of "sweeps". In the sweep process the streaming-plus-collision term is numerically inverted for successive elements along the direction of particle travel; the algorithm "sweeps" across the spatial mesh from the incoming boundary to the outgoing boundary. By respecting the numerical dependencies among tasks induced by the physics of particle streaming this method obtains some desirable iterative properties. For example, in a pure absorber a single set of sweeps will immediately converge to the discrete solution.

Unfortunately the same conditions that make the sweep process effective also limit its parallel scalability. Dependencies among tasks imply a lack of concurrency; the maximum parallelism is determined by the maximum degree of concurrency of the sweep tasks. For example, in one-dimensional slab geometry a spatial mesh with N elements will have N sweep tasks per angle, but a maximum concurrency of only one; it will require a minimum of N computational steps to sweep each angle regardless of the number of available processors. Similar concurrency limits occur in multidimensional geometries [1,2].

In this paper we investigate an alternative to parallel sweeps, the method of cyclic reduction. This method has been successfully applied elsewhere for "prefix" problems; in some cases first-order transport discretizations may be recast as prefix problems. Although the method of cyclic reduction requires more computational operations than the method of sweeps, it has better theoretical parallel scaling properties.

The rest of this paper is organized as follows. A discussion of parallel scaling in general and the scaling of sweeps in particular is given in Section 2. A presentation of prefix problems, the method of cyclic reduction, and their application to one-dimensional transport problems is found in Section 3. In Section 4 we extend and analyze this approach for multidimensional problems, and in Section 5 we present computational results. We give some final discussion and conclusions in Section 6.

2. PARALLEL SCALING AND TRANSPORT SWEEPS

There are several factors that may affect the scalability of a parallel algorithm. Load imbalance, a difference in the total amount of work assigned to each processor, will degrade the parallel efficiency of an algorithm, since some processors will take longer than the ideal time to complete their tasks. Communication costs reduce the parallel efficiency, since they are a form of overhead not found in a serial calculation. In addition, dependencies among tasks may force some processors to remain idle due to a lack of concurrency, which reduces efficiency. The design of effective parallel algorithms involves tradeoffs among these and possibly other factors.

A lack of concurrency is the primary factor affecting the scalability of parallel sweep algorithms. To illustrate this we depict the sweep process in one-dimensional Cartesian (slab) geometry for one angle in Figure 1. Here ψ_n is the unknown flux exiting element n, with ψ_0 the known incident boundary flux. In the first step the known incident boundary flux (ψ_0) is combined with

data for element 1 (attenuation coefficients and sources) to produce the flux exiting the element (ψ_1). In the second step the flux entering element 2 (ψ_1) is used to compute the exiting flux (ψ_2). This process continues until the solutions for all elements have been determined. The curved arrows in each step indicate dataflow, which may involve communication for a parallel application.

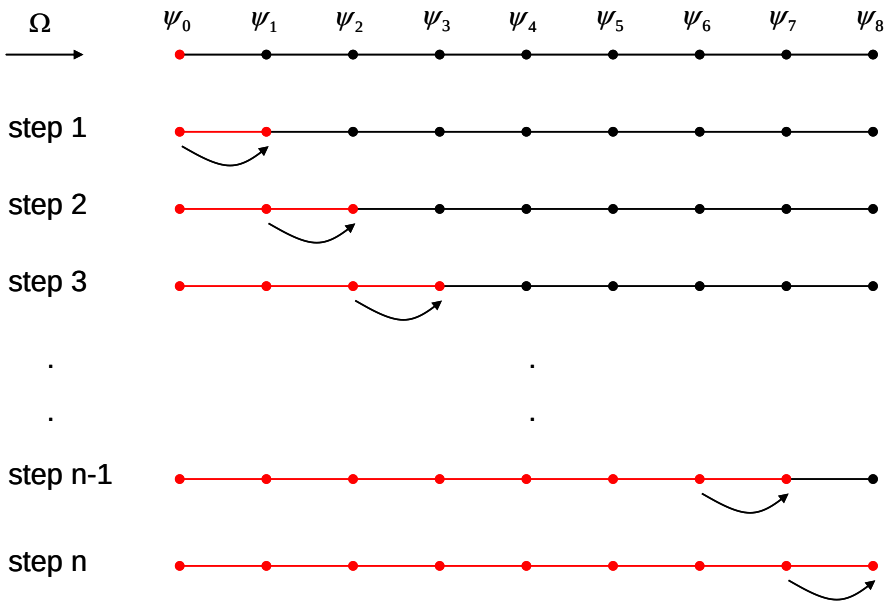

Figure 1: Sweep solver.

We note that the sweep algorithm proceeds in the above sequential manner not merely out of simplicity but out of necessity. At any given step there is only one task with sufficient data to compute the associated outgoing flux. An equivalent statement is that the maximum number of concurrent tasks for one angle in slab geometry is one. This is true regardless of the number of processors available or how the work is divided. Thus no amount of parallelization will result in computational speedup for this example.

The situation is similar in multidimensional geometries, but with additional complexities. In slab geometry each element depends on no more than one other element for a given angle; in multidimensional problems each element may depend on more than one other element. In slab geometry there are no concurrent tasks for the same angle; in multidimensional geometry there usually is some limited concurrency. In either case the potential parallel speedup is limited by the depth of the sweep task graph; no parallel sweep algorithm can complete the process in fewer steps than that depth.

3. PREFIX-BASED SOLVERS FOR SLAB GEOMETRY TRANSPORT

In the previous section we noted that sweep-based approaches to transport are subject to concurrency limits that in turn place upper bounds on the amount of effective parallelism that can be achieved. In this section we turn our attention to a class of problems called prefix problems, for which effective parallelization techniques exist. We then show how slab geometry transport problems can be (and have been) recast as prefix problems, thereby overcoming the concurrency issues of sweep-based solvers. We define two different cyclic reduction solvers for slab geometry transport that will later be extended to multidimensional transport problems.

3.1. Prefix Problems

First we define a prefix problem. A *prefix computation* (or *prefix sum*) is defined [3] in terms of a binary, associative operator $\otimes$. The computation takes as input a sequence $\langle x_0, x_1, \cdots, x_n \rangle$ and produces as output a sequence $\langle y_0, y_1, \cdots, y_n \rangle$ such that

$$\begin{aligned} y_0 &= x_0 \\ y_k &= y_{k-1} \otimes x_k = x_0 \otimes x_1 \otimes \cdots \otimes x_k \end{aligned}. \tag{2}$$

A prefix problem may be solved sequentially, with each operation yielding one of the y_k, as indicated by the parentheses below:

$$\left[\left(\left\{\left(x_0 \otimes x_1\right) \otimes x_2\right\} \otimes \cdots\right) \otimes x_k\right], \tag{3}$$

The innermost operation above yields y_1, the next operation yields y_2, etc. After *n* operations we obtain the entire sequence $\langle y_0, y_1, \cdots, y_n \rangle$.

The key to producing efficient parallel solvers is to exploit the property of associativity. This property allows us to regroup the operations in various ways; one possible grouping is

$$\left[\left(\left\{x_0 \otimes x_1\right\} \otimes \left\{x_2 \otimes x_3\right\}\right) \otimes \left(\left\{x_4 \otimes x_5\right\} \otimes \left\{x_6 \otimes x_7\right\}\right) \otimes \cdots \otimes x_n\right]. \tag{4}$$

Note that at each level of nested parentheses there are numerous independent operations that may be computed concurrently. By exploiting the property of associativity numerous parallel prefix algorithms have been created by computer scientists. Although the details vary, they typically yield solutions after $O(\log(n))$ parallel steps, which is asymptotically less than the $O(n)$ steps required by the sequential approach.

3.2 Slab Geometry Transport as a Prefix Problem

In our earlier description of slab geometry transport we showed that the solution to the streaming-plus-collision operator within some element depends on the solution to at most one other element as well as known information such as fixed sources. We now recognize that this is a particular example of a prefix problem (assuming that the transport operator is associative, which is true for linear discretizations). For some element *k* the incoming angular flux maps to y_{k-1}, the outgoing flux maps to y_k, the fixed sources map to x_k, and the discretized streaming-plus-collision operator maps to $\otimes_k$. Thus in principle we should be able to apply one of the existing parallel prefix algorithms to create an efficient parallel transport algorithm for slab geometry.

As examples of the application of prefix solvers to slab-geometry transport problems we define two different cyclic reduction algorithms. The first, which we denote as the "forward" method, is depicted in Figure 2. In step 0 we express each exiting flux ψ_i algebraically in terms of known sources and the (generally unknown) local incident flux (i.e. $\psi_i = \tau_i \psi_{i-1} + Q_i$). We then recursively communicate this functional dependence to progressively more distant processors, allowing ψ_i to be expressed in terms of $\psi_{i-2^{j-1}}$ at step *j*. The dataflow at each step consists of two quantities: the accumulated attenuation coefficients for a portion of the mesh (e.g. $\tau_1 \tau_2$) and the accumulated sources over the same region (e.g. $\tau_2 Q_1 + Q_2$). After a logarithmic number of steps the solution throughout the entire mesh has been obtained. This process may be applied simultaneously for every angle in the problem.

In Figure 3 we depict a variant algorithm we denote as "symmetric". Whereas in the forward algorithm every element propagated its functional dependence to its downstream neighbor during the first step, in the symmetric algorithm only the odd-numbered ones do. In the second step we implicitly renumber the elements that received data in the first step, and again only the odd-numbered ones propagate data. This continues for $O(\log(N))$ steps, and then we reverse the process, switching the roles of even and odd elements. This algorithm takes twice as many steps as the forward algorithm, but the volume of dataflow is substantially reduced.

3.3 Parallelization of Prefix Solvers for Slab Geometry Transport

In the above description of cyclic reduction we have not discussed actual parallelism, only an alternative set of operations which yield the same solution as sweeps. The total number of operations is greater than that of sweeps, but all of the operations in a given cycle (step) are completely concurrent. It is this concurrency that we can use to construct a parallel algorithm. As an extreme limit of parallelism we can map *n* processors to each of *n* elements and complete the algorithm in $O(\log(n))$ steps. This is in fact what is depicted in Figures 2-3, provided that we interpret each element as the sole member of a processor domain and each of the dataflow arrows as messages between processors. Assuming that the underlying communication network is sufficiently fast this algorithm will be asymptotically faster than a sweep-based algorithm.

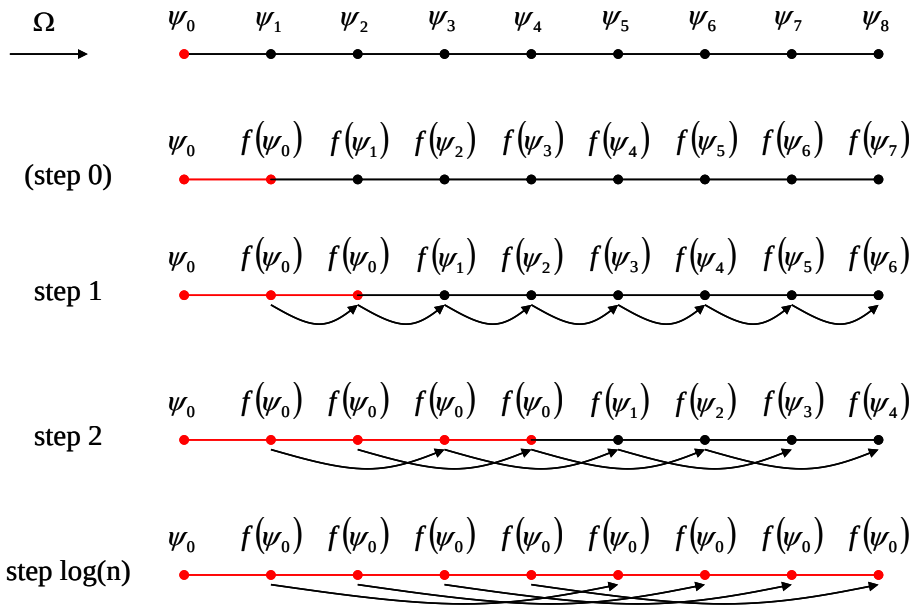

Figure 2: "Forward" cyclic reduction solver.

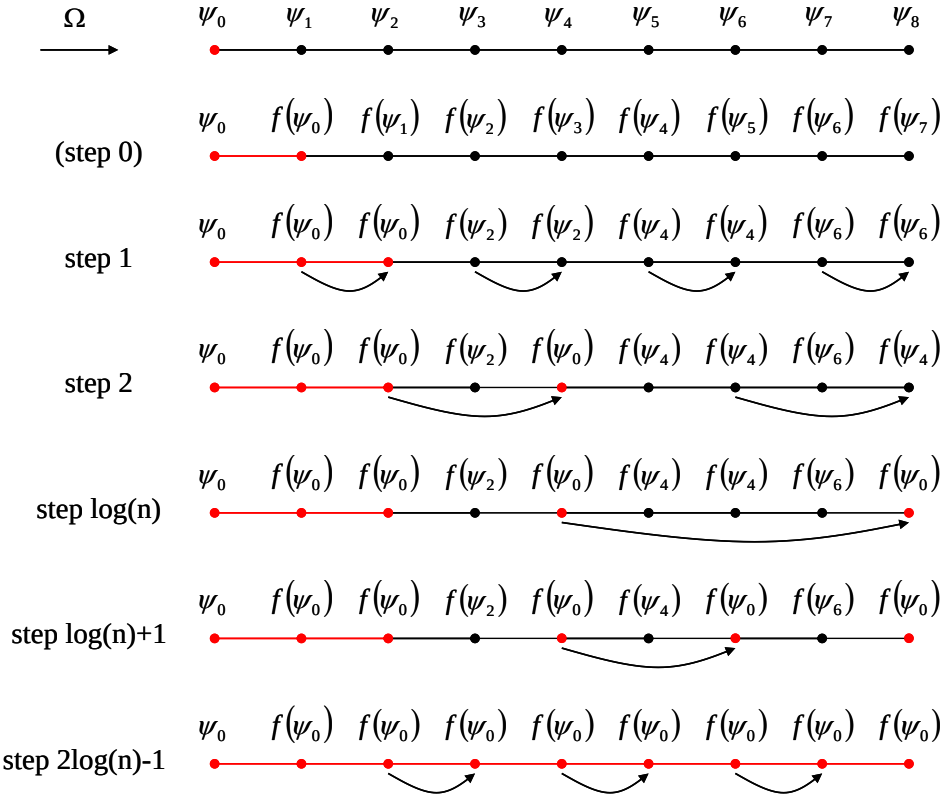

Figure 3: "Symmetric" cyclic reduction solver.

In general we will not have the same number of processors as mesh elements. Certainly we could emulate such a situation with a smaller number of processors, which would require all of the concurrent operations that are mapped to the same physical processor to be executed sequentially. In this situation the extra operations of the cyclic reduction algorithm will have purchased more concurrency than we can effectively use. Here we note however that the cyclic reduction algorithm may be modified to create only as much concurrency as needed, resulting in fewer total operations than the extreme case of complete parallelization but more operations than a fully serial (sweep-based) algorithm. The modified algorithm proceeds as follows:

1. Perform a local (concurrent) set of sweeps on each processor domain to determine attenuation coefficients and source contributions across the entire domain (e.g. $\tau_{p0} = \tau_1\tau_2\tau_3$). This is equivalent to performing two separate block-Jacobi sweep iterations, one with a unit incident boundary flux in the absence of internal sources in order to obtain the domain attenuation coefficients, and then another with no incident fluxes but with the internal sources in order to determine the source contribution from the whole domain.
2. Perform the usual cyclic reduction algorithm using the above domain data rather than individual element data in order to solve for the fluxes exiting each domain (and thus entering the domain immediately downstream).
3. Perform a final set of block-Jacobi sweeps with the known fluxes incident on each domain in order to solve for each individual element.

Since each local sweep takes $O(n/p)$ time, and the cyclic reduction step takes $O(\log(p))$ time, the overall parallel algorithm will complete in $O(3n/p)+O(\log(p))$ time. For small p this is larger than the $O(n)$ time of sweeps, but for large p it is asymptotically smaller.

Our description of a parallel prefix method for slab geometry transport is not unique or original. At least two other researchers independently constructed similar algorithms based on equivalent reasoning over a decade ago [4-5]. However, it was not obvious whether these approaches could be extended to multidimensional calculations, for reasons discussed in the next section.

4. PREFIX-BASED SOLVERS FOR MULTIDIMENSIONAL TRANSPORT

4.1 Extension of Prefix Solvers to Multidimensional Transport Problems

The results for slab geometry transport are not easily extended to multidimensions. Most discretizations (including the familiar class of finite element methods) yield a set of discrete equations that do not define a prefix problem. Prefix problems require that each angular flux variable depend on at most one other variable, whereas in many transport discretizations these variables have as many upstream dependencies as there are incoming element faces. We have recently recognized that there is one notable exception to this trend: the class of long characteristics (LC) discretizations. The flux exiting an element along one of these characteristics depends only on known source terms and on the (generally unknown) flux entering the element on the same characteristic. Thus LC discretizations define a set of prefix problems, which we postulate may be solved more efficiently by a parallel prefix solver than by a parallel sweeps approach.

The extension of the cyclic reduction solvers to LC discretizations is conceptually straightforward. Each characteristic defines an effectively one-dimensional streaming-plus-collision problem; multidimensional effects are restricted to the calculation of source terms. A depiction of one step of the forward algorithm is shown in Figure 4, where the dataflow for just one characteristic is illustrated. This process may be conducted concurrently with other characteristics and other angles. Parallelization is also implemented in a manner analogous to the slab geometry case.

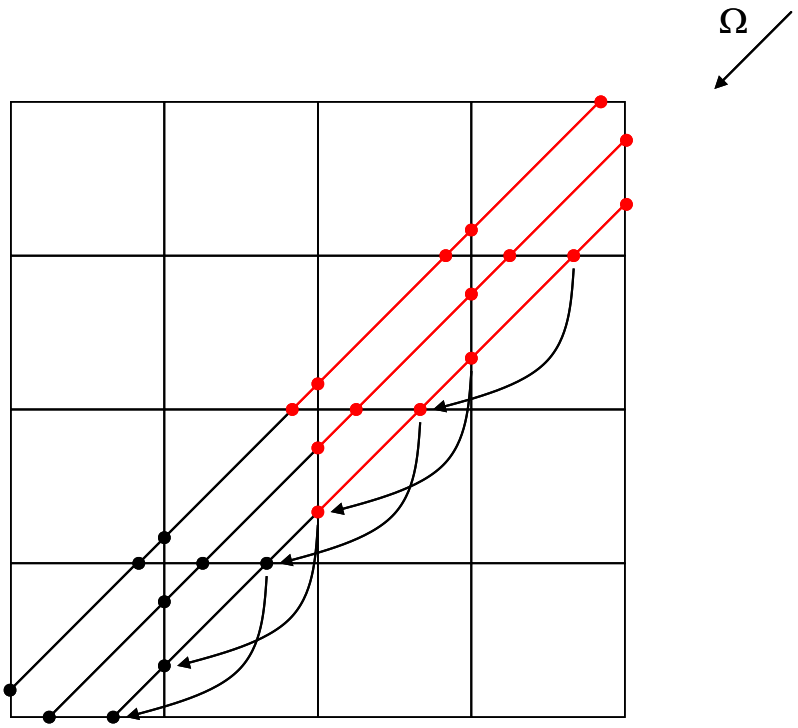

Figure 4: Cycle 2 of "forward" cyclic reduction solver applied to LC problem.

4.2 Efficiency Analysis of Multidimensional Parallel Cyclic Reduction Solvers

We have analyzed the theoretical costs of the proposed parallel cyclic reduction algorithms for two-dimensional LC discretizations. We are particularly interested in their asymptotic behavior as the problem size and processor count increase, which we wish to compare with the corresponding expressions for the cost of parallel sweep-based solvers.

For the purpose of this analysis we restrict our attention to a mesh of quadrilateral elements laid out in a structured IxJ grid. We examine two different situations. In the general/worst case we make no assumptions about the mesh partitioning other than load balance, we assume that the communication network of the host computer system is mesh-based with a diameter of $\sqrt{P}$, and we assume that messages are sent with store-and-forward routing. In the more restricted case we assume that the domains form a P_xxP_y grid (i.e. rectangular domains), the communication network is tree-based with a diameter of $2\log_2(P/2)$, and that cut-through routing is used. In

both cases we assume that there is no network contention; the time to send and receive a message is independent of the existence of other message traffic. We also make the following definitions:

R: average number of rays intersecting an element in each angle
M: number of angles per quadrant
db: number of bytes per double-precision variable
ω: time to invert the streaming-plus-collision operator per ray segment per element
T_s: message startup time
T_w: communication bandwidth
T_{hop}: time to forward message at each node of the network

For the sake of space we simply report the final results in the extreme limit in which P=IJ. The efficiencies of the forward solver in the general case, the forward solver in the restricted case, and the method of sweeps [1] are:

$$\varepsilon_{gen} = \left[3 + 2\log_2\left(\sqrt{P}\right)\frac{T_s}{R\omega} + 2\sqrt{P}\log_2\left(\sqrt{P}\right)\frac{dbT_w}{\omega} + \sqrt{P}\log_2\left(\sqrt{P}\right)\frac{T_{hop}}{R\omega} \right]^{-1}$$

$$\varepsilon_{res} = \left[3 + 2\log_2\left(2\sqrt{P}\right)\frac{T_s}{R\omega} + 2\log_2\left(2\sqrt{P}\right)\frac{dbT_w}{\omega} + 2\log_2\left(P/2\right)\log_2\left(2\sqrt{P}\right)\frac{T_{hop}}{R\omega} \right]^{-1} \qquad (10)$$

$$\varepsilon_{sweep} = \left[1 + \frac{\sqrt{P}}{2M} + \frac{\sqrt{P}}{2M}\frac{T_{comm}}{\omega} \right]^{-1}$$

Here we see clearly the asymptotic drop-off in efficiency of parallel sweeps, which decays as $P^{-1/2}$ regardless of communication speed. We also see poor asymptotic efficiency for a parallel cyclic reduction algorithm when executed on a mesh-based architecture; in this case a communication sweep through a network has replaced a computational sweep through the finite element mesh. The efficiency in the restricted case, however, displays only logarithmic decay. We note that this improvement comes strictly as a result of the network architecture and does not depend on the partitioning, since there is only one possible partitioning when P=IJ.

5. RESULTS

To test our analysis we implemented a simple LC discretization within Sandia's radlib/SCEPTRE code base[6]. We also implemented the forward and symmetric cyclic reduction algorithms as alternatives to SCEPTRE's existing sweeps solver. Various scaling studies were performed on Red Storm (12,960 dual-core AMD Opteron processors) and on Thunderbird (4480 dual core Intel EM64T processors). Timings on Red Storm and Thunderbird for S_8 quadrature for several quadrilateral meshes are shown in Figures 5-8. In these studies we show sweeps results for different quadrature "mappings" that improve cache reuse at the expense of iterative effectiveness, so sweeps results for S_2 and S_4 mappings will generally be worse than implied when the overall time to solution is compared. These and other results generally confirm our theoretical analysis: sweeps-based approaches are faster than cyclic reduction for low processor counts, but the reverse is true for sufficiently high parallelism.

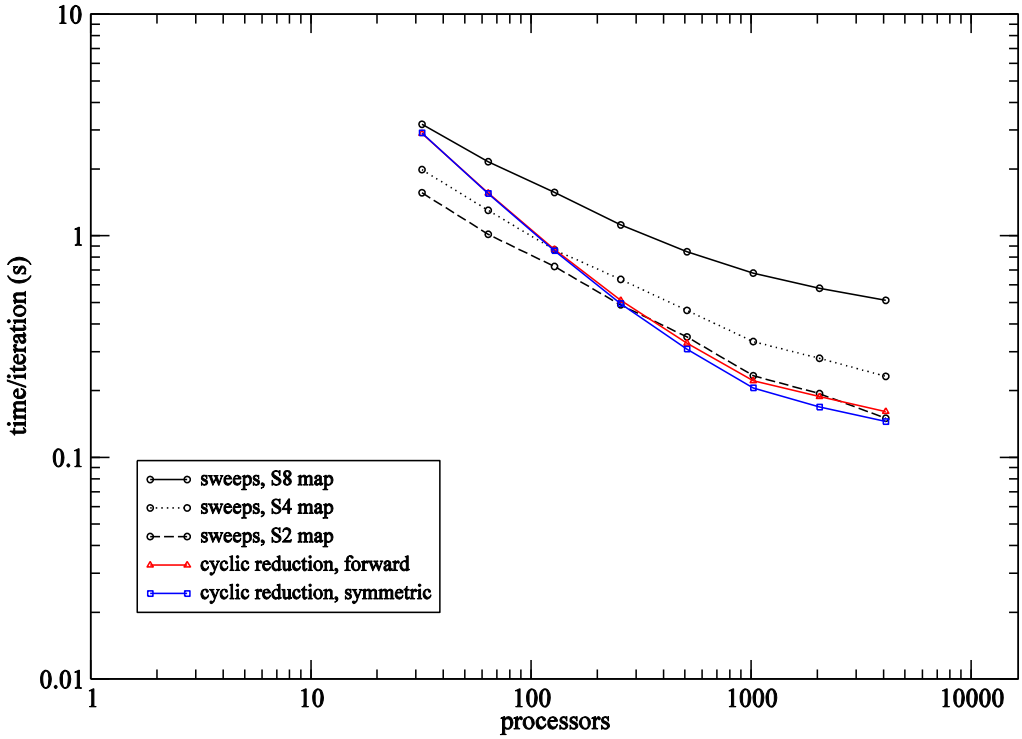

Figure 5: Red Storm scaling, 512x512 quad4 mesh.

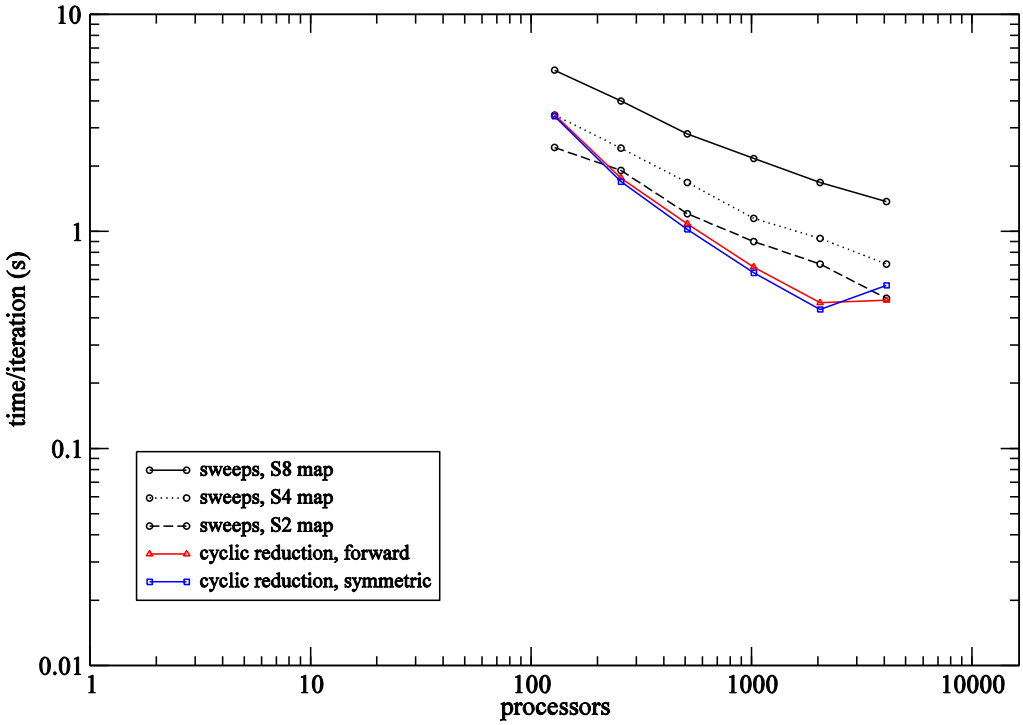

Figure 6: Red Storm scaling, 1024x1024 quad4 mesh.

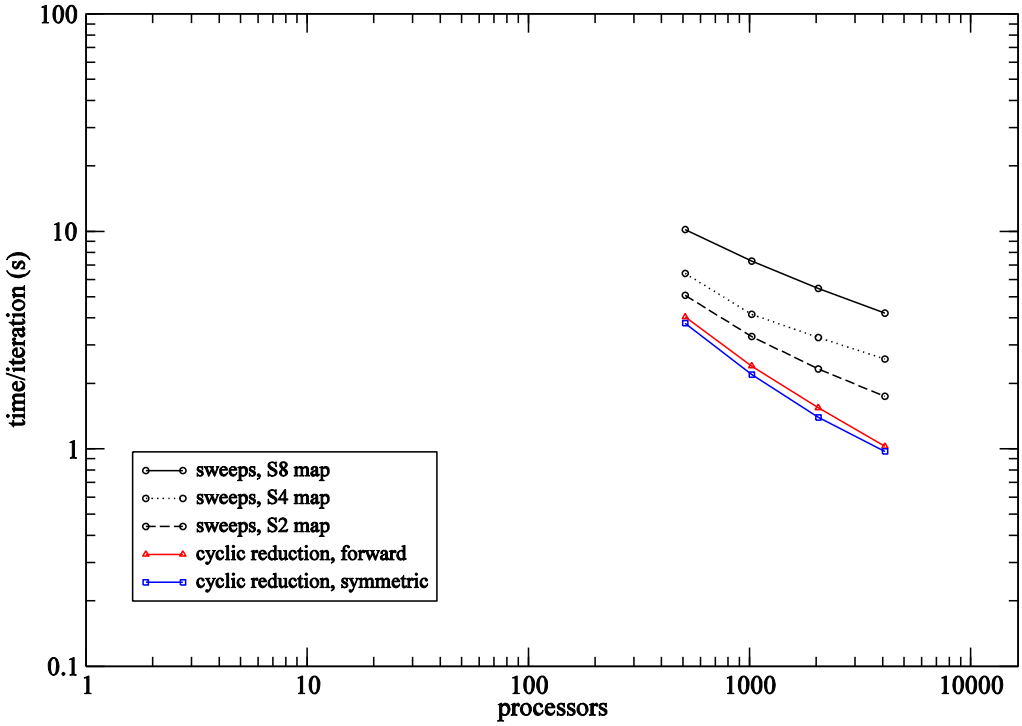

Figure 7: Red Storm scaling, 2048x2048 quad4 mesh.

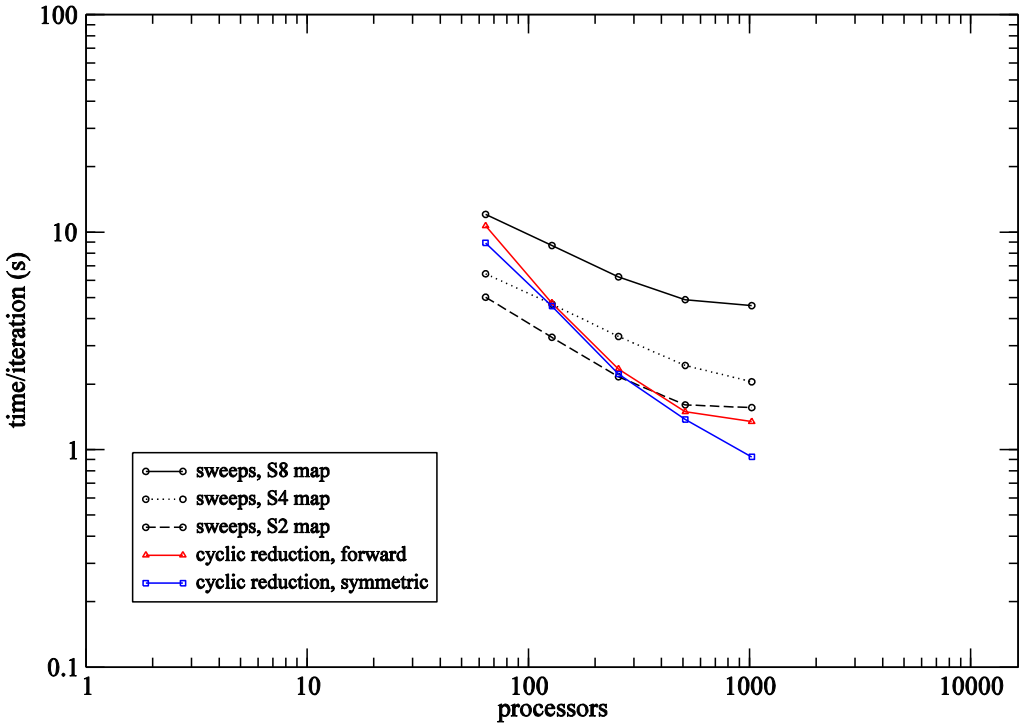

Figure 8: Thunderbird scaling, 1024x1024 quad4 mesh.

6. CONCLUSIONS

The parallel performance of sweep-based transport solvers is inherently limited by available concurrency. It was the purpose of this study to examine the potential of alternative solution strategies based on prefix approaches that have greater concurrency. One prefix strategy, that of cyclic reduction, was developed and analyzed in this study; its theoretical scaling performance compares favorably with the sweeps-based approach. Actual computer scaling studies generally validate our theoretical predictions. Therefore we conclude that massively parallel transport methods based on these approaches demonstrate great potential for favorable performance on future systems in comparison to current sweeps-based approaches.

The above conclusion is contingent on several factors, however. Our performance models indicate that favorable scaling requires that future parallel systems have a small diameter, e.g. $\log(P)$ rather than $P^{1/2}$. Mesh-based architectures are predicted to display the same asymptotic scaling degradation with cyclic reduction solvers as all systems do with sweeps-based solvers. Even if future systems have a small diameter it is not clear whether network contention will be problematic; our models ignore this effect. In some of our timing studies we did observe degradation in the performance of the cyclic reduction solver at very large processor counts; it is unclear whether this is an intrinsic effect at this level of parallelism or whether future systems will delay this degradation to much larger levels of parallelism.

REFERENCES

1. M.L. Adams, N. Amato, P. Nelson, and L. Rauchwerger, "Efficient Massively-Parallel Implementation of Modern Deterministic Transport Calculations", Report to the Department of Energy, Texas A&M University, College Station, TX (2002).
2. S.D. Pautz, "An Algorithm for Parallel S_n Sweeps on Unstructured Meshes," *Nucl. Sci. Eng.*, **140**, 111-136 (2002).
3. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA (1999).
4. S. Oliveira, "Parallel multigrid methods for transport equations: The anisotropic case", *Parallel Computing*, **22**, pp. 513-537 (1996).
5. R.D. Jarvis and P. Nelson, "Time-Optimal Parallel Ray-Tracing Sweeps in Plane-Parallel Discrete Ordinates: Implementation and Testing on a Hypercube", unpublished article, Texas A&M University, College Station, TX (1997).
6. S. Pautz, B. Bohnhoff, C. Drumm, and W. Fan, "Parallel Discrete Ordinates Methods in the SCEPTRE Project," *Proceedings of International Conference on Mathematics, Computational Methods and Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009.