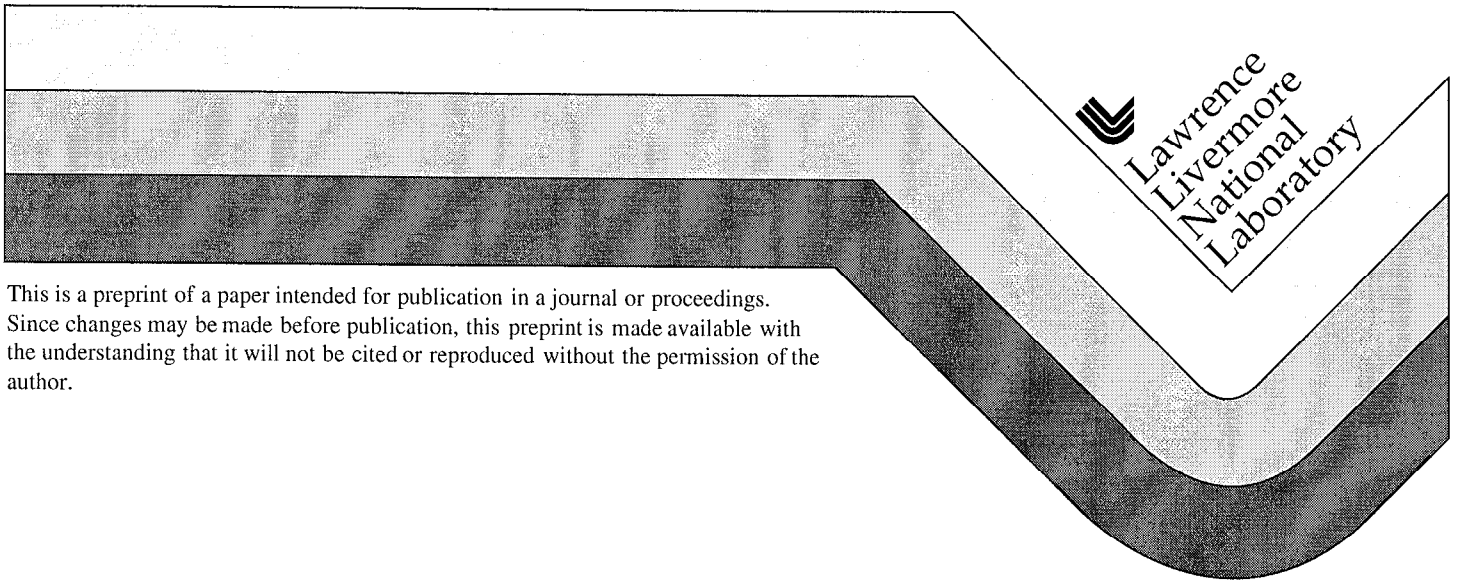
UCRL-JC-133787
PREPRINT

Radiation Transport Calculations on Unstructured Grids using a Spatially Decomposed and Threaded Algorithm

P.F. Nowak
M.K. Nemanic

This paper was prepared for submittal to the
Mathematics and Computation, Reactor Physics
and Environmental Analysis in Nuclear Applications
Madrid, Spain
September 27-30, 1999

April 12, 1999



Radiation Transport Calculations on Unstructured Grids using a Spatially Decomposed and Threaded Algorithm

Paul F. Nowak and Michael K. Nemanic
Lawrence Livermore National Laboratory
P.O. Box 808, L-22
Livermore, California 94551
pnowak@llnl.gov ; nemanic@llnl.gov

Abstract

We consider the solution of time-dependent, energy-dependent, discrete ordinates, and nonlinear radiative transfer problems on three-dimensional unstructured spatial grids. We discuss the solution of this class of transport problems, using the code TETON, on large distributed-memory multinode computers having multiple processors per "node" (e.g. the IBM-SP). We discuss the use of both spatial decomposition using message passing between "nodes" and a threading algorithm in angle on each "node". We present timing studies to show how this algorithm scales to hundreds and thousands of processors. We also present an energy group "batching" algorithm that greatly enhances cache performance. Our conclusion, after considering cache performance, storage limitations and dependencies inherent in the physics, is that a model that uses a combination of message-passing and threading is superior to one that uses message-passing alone. We present numerical evidence to support our conclusion.

1 Introduction

This work is motivated by the need to obtain accurate transport solutions to three-dimensional radiative transfer problems (i.e. the transport of thermal photons). The class of simulations we consider are those in which radiation flows through a moving background media, where the spatial grid moves with the fluid. This Lagrangian hydrodynamics approach can produce highly distorted spatial meshes as the calculation proceeds, thus requiring an unstructured grid capability for the transport method. This class of problems is also characterized by tens of thousands of unknowns per zone and upwards of millions of zones, thus requiring large parallel computing platforms (thousands of processors) to accommodate both memory and solution time requirements. A computing platform capable of solving such large problems is the IBM-SP machine at Lawrence Livermore National Laboratory (LLNL). The machine consists of three MPPs, each with 488 compute "nodes." Each node is an SMP of four CPUs and from 1.5 to 2.5GB of shared memory.

This paper differs from previous works in that, 1) it discusses the combination of message passing and threading algorithms, 2) it discusses the special challenges that result from unstructured spatial grids and 3) it presents a highly efficient algorithm for handling multiple energy groups that greatly improves cache and thread efficiency. The remainder of the paper is organized as follows. In the next section we describe the radiative transfer problem and its solution in the context of the physical and algorithmic characteristics that influence parallelism. The algorithms described are those in the transport code TETON at LLNL, which is written in FORTRAN 90 and was originally optimized for the Cray vector architecture. In Section 3, we describe how problems are decomposed in space

and distributed over nodes using a message-passing algorithm. In Section 4, we describe the solution of the resulting problem on a "node," where the node has one or more processors sharing a common memory. This section discusses both threading and a new "batching" algorithm for multiple energy groups. In Section 5, we compare various approaches applied to very large problems and show scaling results. We conclude by discussing work in progress and suggest future research areas.

2 Problem Description

2.1 Physical Problem

The equations for thermal radiative transfer describe the transport, absorption, scattering and emission of photons within a physical material. As photons propagate, they are both absorbed and emitted by the material thereby changing the properties of the material. These changes in the material properties subsequently alter the rate and the energy spectrum of emission, as well as the photon mean-free-path. These physical processes are modeled with the Boltzmann transport equation for photons

$$\left(\frac{1}{c}\frac{\partial}{\partial t}+\Omega\cdot\nabla+\sigma(\mathbf{r},E,T(t),\rho(t))\right)I(\mathbf{r},\Omega,E,t)=\sigma_a(\mathbf{r},E,T(t),\rho(t))B(\mathbf{r},E,T(t))+$$
$$Q_R(\mathbf{r},t)+\int_0^\infty dE'\int_{4\pi}d\Omega'\sigma_s(\mathbf{r},E'\to E,\Omega'\to\Omega,T(t),\rho(t))I(\mathbf{r},\Omega',E',t), \tag{1a}$$

and an energy balance equation for the material

$$c_v(\mathbf{r},T(t),\rho(t))\frac{\partial T(\mathbf{r},t)}{\partial t}=\int_0^\infty dE\sigma_a(\mathbf{r},E,T(t),\rho(t))\int_{4\pi}d\Omega\Big(I(\mathbf{r},\Omega,E,t)-B(\mathbf{r},E,T(t))\Big)+Q_M(\mathbf{r},t), \tag{1b}$$

together with the initial and boundary conditions

$$\begin{aligned} I(\mathbf{r},\Omega,E,0)&=I_i(\mathbf{r},\Omega,E),\\ T(\mathbf{r},0)&=T_i(\mathbf{r}),\\ I(\mathbf{r},\Omega,E,t)&=I_b(\mathbf{r},\Omega,E,t),\quad \mathbf{r}\in\partial D,\quad \Omega\cdot\mathbf{n}<0. \end{aligned} \tag{1c}$$

The following definitions apply:

$I(\mathbf{r},\Omega,E,t)$	=	radiation intensity,
$B(\mathbf{r},E,T)$	=	Planck function,
$T(\mathbf{r},t)$	=	material temperature,
σ,σ_s,σ_a	=	total, scattering, absorption opacity,
ρ	=	material density,
c_v	=	specific heat,
c	=	speed of light in a vacuum,
Q_R	=	external radiation source,
Q_M	=	external matter source.

The fundamental unknowns are the radiation intensity I and the material temperature T in a seven dimensional phase space (three in position **r**, two in direction Ω, one in energy E, and one in time t). These equations exhibit strong nonlinear coupling through the emission terms and the material opacity. The material opacity itself is a very strong function of temperature, photon energy and atomic number and can range over ten orders of magnitude in a single calculation. The end result is that fine resolution in space, energy and direction is required to adequately represent the photon distribution in time. A thorough discussion of the above equations and the physics they represent may be found in [Pomraning, 1973].

2.2 Numerical Algorithm

This paper discusses the deterministic solution of the radiative transfer problem described by Eq. 1, in which the differential equations are discretized in space, energy, direction and time and solved as a large system of coupled equations. The discretization techniques used in TETON are standard or have been discussed elsewhere: upstream corner-balance in space [Adams, 1997], multigroup in energy, discrete ordinates in angle and fully implicit in time. (An excellent reference is [Lewis, 1984].) The system is solved using the grey transport acceleration method (GTA) [Larsen, 1988], except that an S_2 quadrature is always used for acceleration. The resulting one group and S_2 problem is solved using transport synthetic acceleration (TSA) [Ramone, 1997].

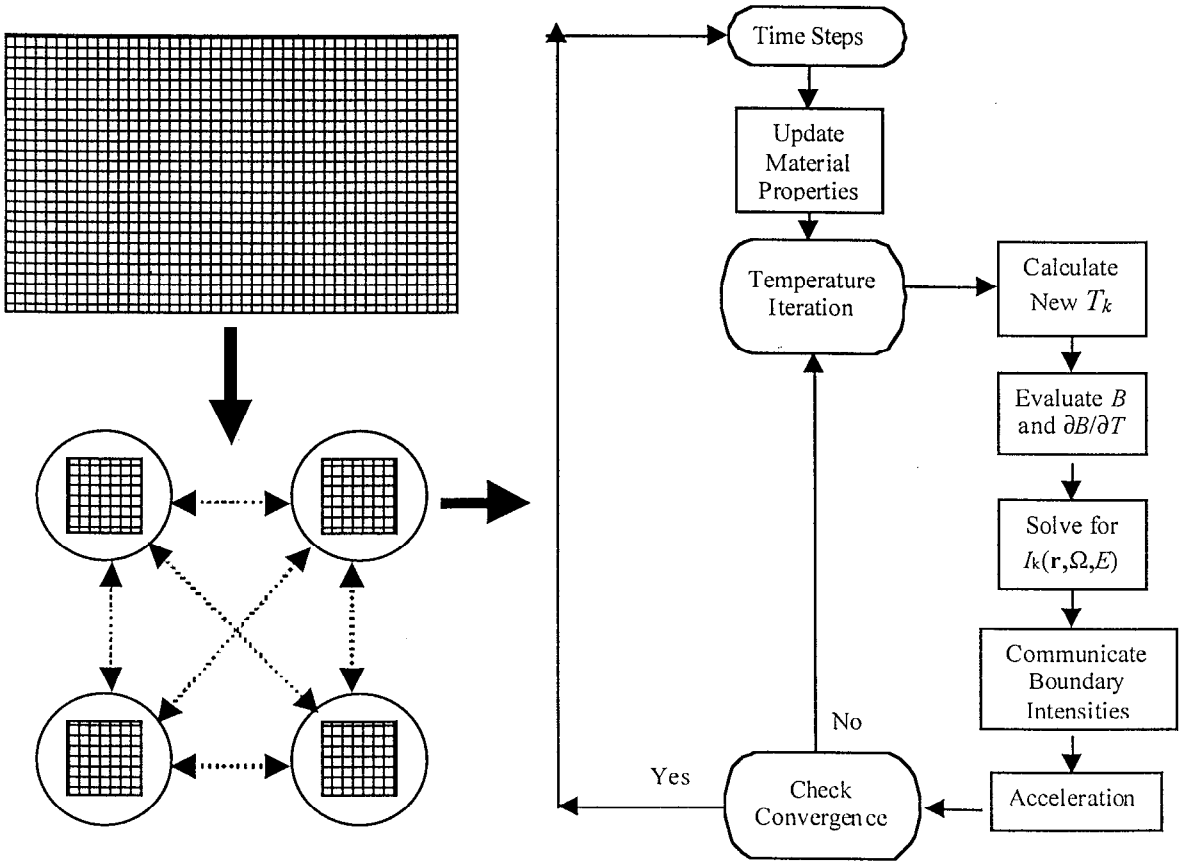


Figure 1. Flowchart of the radiative transfer iteration on a node.

A brief discussion of our solution procedure follows. We use a fully implicit time discretization with the exception that the opacities and specific heats are evaluated at the beginning-of-timestep temperatures. To obtain an implicit Planckian, we use a Newton-Raphson iteration (with iteration index k) based on the following first order expansion

$$B(T_{k+1}) \approx B(T_k) + (T_{k+1} - T_k)\left.\frac{\partial B}{\partial T}\right|_{T_k} . \tag{2}$$

Substitution of Eq. (2) into the discretized form of Eq. (1a) and Eq. (1b) leads to a linear steady-state multigroup transport problem to be solved for each Newton-Raphson iteration on each time step. The flowchart in Fig. 1 displays our overall iteration strategy for each time step across each spatial subdomain. Starting with a beginning of time step temperature and material properties, we evaluate the emission source and then solve for the intensity in each direction and energy group over the entire spatial domain. The boundary intensities are then communicated: a domain sends it's exiting intensities to and receives incoming intensities from all neighboring domains. This is followed by a "local" acceleration step (i.e. there is no internode communication except to test convergence of the TSA iteration), an update of the material temperature and finally a global convergence test on the temperature and the integrated intensity. If convergence is satisfied, we advance to the next time step. Otherwise, we continue the temperature iteration loop.

3 Distribution of the Problem to Nodes

The class of problems described in this paper requires a six dimensional calculation (three in space, two in angle and one in energy) at each time step. Thus it is possible to decompose a problem in space, angle and energy, either separately or in combination. In order to guide us in determining the best decomposition strategy, we consider the following criterion:

- It must optimize both the high-order problem (multigroup, S_N) and the low-order problem (one group, S_2) used for iterative acceleration.
- It must scale to problems with billions of unknowns using thousands of processors.
- It must handle multigroup physics constraints (e.g. group to group scattering).
- It should try to preserve single processor convergence rates.
- The communication costs should be low and not increase substantially with the number of domains.
- It should minimize total computation time and storage.
- The resulting computation on a single node should be as cache efficient as possible.

Given the above criterion, spatial domain decomposition is the method of choice. Energy decomposition fails because the acceleration step cannot be decomposed and because it requires that each node hold the entire mesh, thus severely limiting the problem size. Decomposition in angle suffers these same limitations, and also requires frequent communication points.

To perform a spatial decomposition, the original mesh is partitioned using the readily available program METIS [Karypis, 1997] and the individual sub-meshes are distributed among the available nodes via message passing from a master node. Each node solves a full radiative transfer problem on its portion of the mesh and, except for communication points, functions as if it were solving the

entire problem. The required synchronization points are to monitor convergence, calculate a new time step and to exchange fluxes on subdomain boundaries. This exchange is carried out using a non-blocking MPI send and receive pipeline [Gropp, 1994], and occurs after each complete intensity solve (Fig. 1).

4 Solution of the Problem on a Node

We now focus on solving the transport problem on a single node defined as a group of processors sharing a common memory. We consider two distinct, but intimately connected, aspects: using multiple threads and optimizing cache performance.

4.1 Threading in Angle

Our criterion for developing a shared-memory (SMP) algorithm using threads is similar to the ones we chose for deciding how to spatially distribute the problem. In the case of threads our primary concern is developing an algorithm that automatically handles both the high-order problem (multigroup, S_N) as well as the low-order acceleration step (one group, S_2). Threading over angles satisfies this constraint, while threading over blocks of energy groups does not.

Threading is accomplished using a single OpenMP [OpenMP, 1997] directive ("parallel do") on the loop over angles. This loop encompasses the source calculation, the sweep through the spatial grid and, as we will see below, the loop over energy groups. Both the solution for the intensities and the acceleration calculation use this angle loop. The goal is to put as much work as possible in the threaded loop and our experience has shown that this is critically important. (An initial attempt at loop level directives provided no more than a speedup of 1.5 out of 4 processors.)

OpenMP has several advantages for our application. Our implementation is extremely "light-weight" in that threads are created for the life of the run, have low startup and synchronization costs, and are easy to use. In a typical problem, ~96% of the total work in the calculation is in the threaded loop and the overhead associated with the threading is less than 2% of the total run time. OpenMP is also flexible so that we were able to experiment with different thread scheduling schemes and lock/signal/barrier functions that are part of the OpenMP directive set. A further advantage of OpenMP is that it is portable across our platforms (DEC and IBM).

Our "single-directive" implementation sounds straightforward, but in reality the code had to undergo an evolving reorganization process to transform the angle loop so that each angle could be done independently by a thread. Thread-private arrays were pre-allocated before entering the parallel region of the angle loop. This was done to limit the amount of data passed on the individual thread stacks, because we were concerned about stack overflow when running large problems. On exiting the parallel region, the thread private array data is combined into the common global arrays.

4.2 Efficient Use of Cache

Due to the fact that unstructured meshes have arbitrary connectivity between grid elements, there is no simple relationship mapping from one subcell to its neighbors. In order to allow for completely arbitrary cell shapes and topologies, it is necessary to compute an ordered list of subcells for each discrete ordinate direction. As a result, indirect memory references are used throughout the lowest level of the algorithm, where the grid is "swept" for each direction. Indirect memory references are

essentially random walks through memory and can result in a cache miss for every memory reference and almost no reuse of the data that resides in cache. Several ways were used to improve this situation and are shown in Fig. 2. Figure 2 displays the CPU time, the percent of "useful" cycles and the number of cache misses for a benchmark problem. At several places, we annotate algorithmic improvements and the resulting improvement in performance. In general, the improvements described below reduced the number of data cache misses and thus resulted in shorter run times and a larger fraction of "useful" cycles.

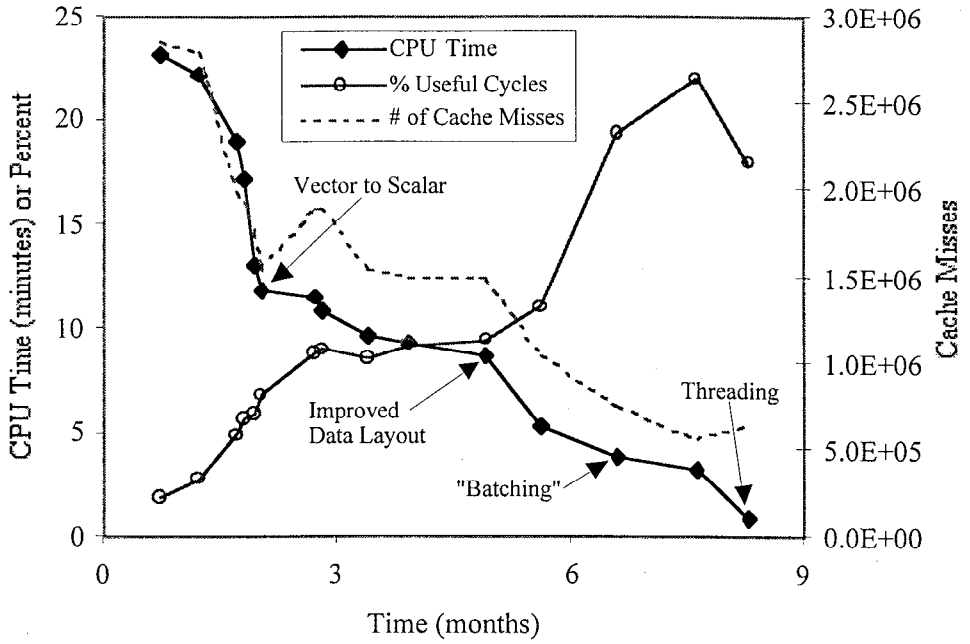


Figure 2. Improvements in serial run time and cache efficiency due to various effects.

The most dramatic improvement resulted from the elimination of "working" arrays in favor of scalars and the fusion of loops. DCPI [Anderson, 1997] was the principal tool used to improve cache performance because it returns timings on stalls and cache misses for each line of source code. (DCPI is a tool unique to the DEC.) With this "high resolution" look at cache efficiency, we were able to modify single lines in which array/array operations were replaced with array/scalar and scalar/scalar operations. This substitution greatly reduced the number of indirect memory references. CacheVu [APR, 1998] is another tool we used to guide us in improving data layout. It instruments an executable and tallies cache misses due to array "collisions." (i.e. when references in two or more arrays cause collisions over the same cache line).

Another large improvement was attained by storing the connectivity information such that one array reference can bring into cache all of the integer data that is required to calculate a single intensity (e.g. list of neighboring subcells, zone index, face index, etc.). The same grouping of data can be done for the real arrays (e.g. incident intensities and sources).

A third remedy is to "prefetch" the data that is required to calculate an intensity. Before entering the loop over subcells, one loads all of the data required to compute the first intensity. Once inside the loop, the intensity for subcell i would be computed while the data required for the intensity at i+1 would be loaded. In this fashion, stalls resulting from slow memory access are amortized over

concurrent calculation. More dramatic gains can be achieved if one maximizes the reuse of data that is already loaded, for example all of the geometry information. We now describe a way to do this.

4.3 Energy Group "Batching"

For the class of radiative transfer problems we are considering, the dominant physical processes are absorption and emission. Due to the fact that an absorbed photon is subsequently emitted in a spectrum over the entire photon energy range (i.e. a Planck function), the transfer matrix is full. As a result, there is really no advantage to calculating the photon groups in an ordered manner. One way to maximize data reuse is to collect or "batch" all photon energy groups that have the same set of angles and transport them together. In this fashion, we can maximize the reuse of data that is in cache and greatly improve the algorithm efficiency. Evidence of this is shown in Fig. 3a and Fig. 3b.

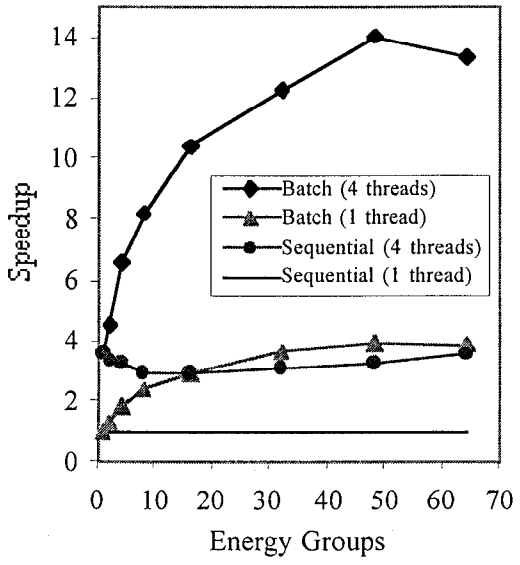


Figure 3a. Speedup vs. groups.

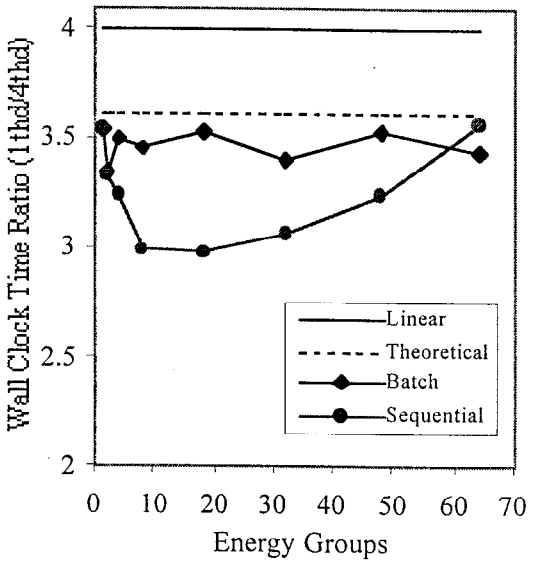


Figure 3b. Thread efficiency vs. groups.

In Fig. 3a, we compare our new "batching" algorithm to the standard "sequential" algorithm (i.e. one group at a time), as a function of the number of energy groups, and using one and four threads. We calculate speedups as compared to the sequential algorithm with one thread. Looking at the "batching" algorithm with one thread, we see that it is roughly four times as fast as the sequential algorithm with one thread, and is faster than the sequential algorithm with four threads when solving for more than 16 energy groups. For the "batching" algorithm with four threads, a speedup of roughly fourteen (compared to reference case) is obtained for 48 energy groups. The speedup curve "rolls over" at 64 groups because the computational data set is getting large enough to increase the number of cache misses (see Fig. 6b for four threads).

In Fig. 3b, we compare the speedup, the ratio of the single thread and four thread time, of the batched and sequential algorithms to the linear and theoretical speedups. The theoretical speedup takes into account the fact that only 96% of the total work is threaded while the remainder is serial.

The "batching" algorithm consistently performs close to the theoretical limit over the range of energy groups tested.

5 Numerical Results

We now present results that show how our combined algorithm scales for large problems. Comparisons are made to the results using MPI alone. In Fig. 5, we show run time as a function of the number of processors for four fixed size problems. The problems range in size from a single group, S_4 calculation on 3.2×10^5 zones ($\sim 6.1 \times 10^7$ unknowns) to an eight group, S_6 calculation on 1.024×10^6 zones ($\sim 3.1 \times 10^9$ unknowns). All four runs show good scaling, having a parallel efficiency of ~90% for the largest processor count compared to the run time per CPU for the smallest processor count. (The problems require too much memory to be run on fewer processors than the minimum shown in Fig. 5.)

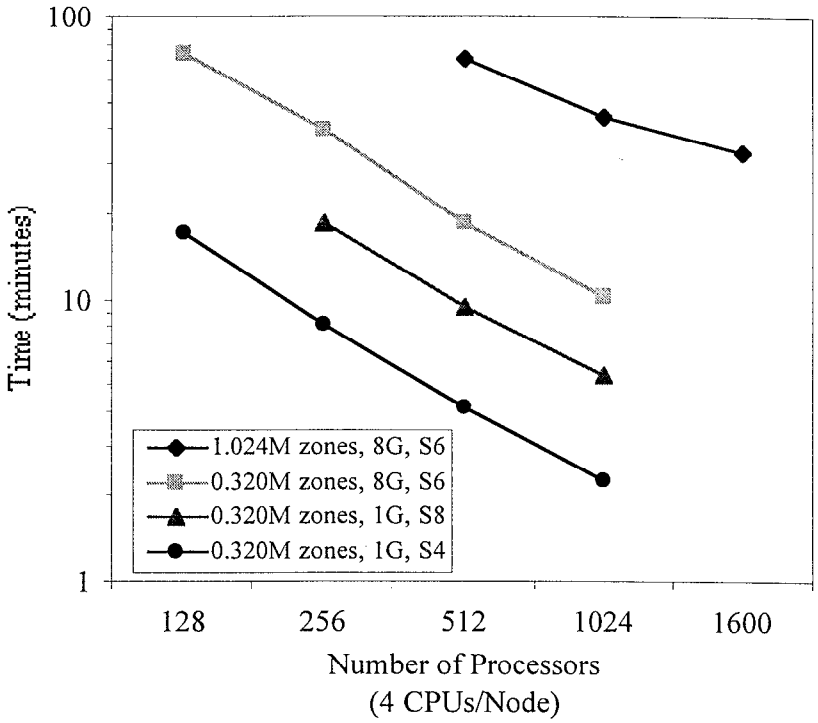
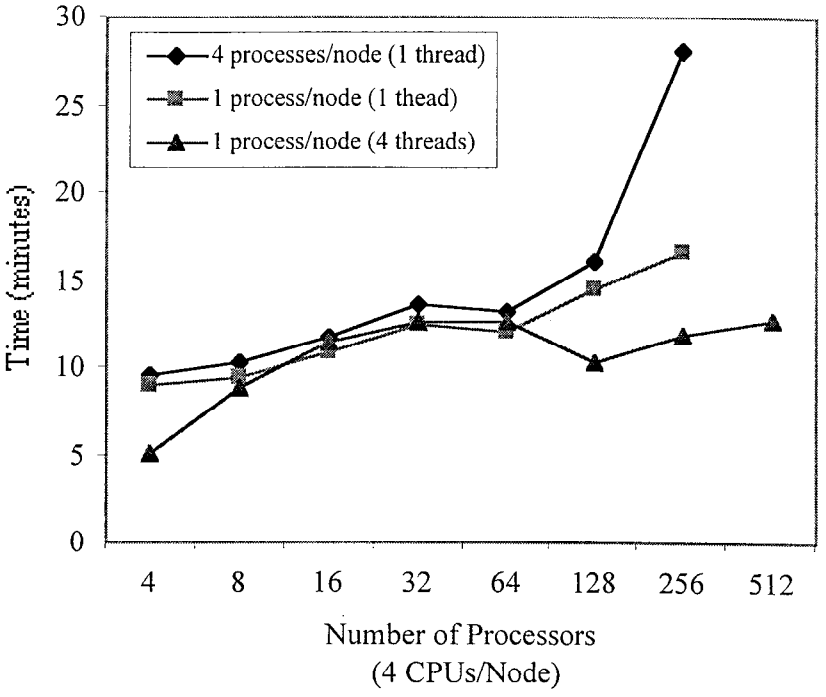


Figure 4. Scaling results for a constant problem size using MPI and threads.

Figure 6 compares constant workload per processor runs made with MPI processes alone and with the combination of MPI and threads (4 threads per node). The fundamental size problem given to a single processor is an eight energy group and S_6 (48 angles) calculation on 2000 zones. As the total number of CPUs is increased, two effects are seen in the MPI only curves. The upper curve, representing 4 MPI processes per node, shows increased CPU time at 128 CPUs due to the "saturation" of the communication sub-system. There is a similar rise in the middle curve, representing 1 MPI process per node. At 256 CPUs, the top curve suffers from an additional cost due to the contention among the 4 MPI processes per node for the single memory bus. The effect of "stressed" communications and contention for memory seem to synergize in the upper curve at 256 CPUs. This is not apparent on the middle curve since it has only 1 MPI process accessing memory.

The first effect is not present at all in the bottom curve, representing 1 MPI process with 4 threads, until the 512 CPUs are used, but even then the "strain" is only marginally significant.

Figure 5. Scaling results for constant work per processor.



6 Discussion and Conclusions

This paper was motivated by the need to obtain time and memory efficient solutions to time-dependent, energy-dependent, nonlinear radiative transfer problems on three-dimensional unstructured spatial grids. Toward that end, we have presented a parallel algorithm that uses a combination of spatial decomposition using message passing between "nodes" and a threading algorithm over angle on each "node". To further improve efficiency of the calculation on a "node," we have described an energy group "batching" algorithm that yields a 4x speedup over the traditional sequential algorithm. We have presented results that demonstrate the scaling and modest communication cost of this approach. We have also shown that for this class of problems, that the combined algorithm is superior to one that uses MPI alone.

The combination of MPI and 4 threads (SMP) has several advantages over 4 MPI processes alone. Since there are fewer communicating processes with MPI-SMP, it can use larger, more efficient messages with fewer processes waiting at synchronization points. In addition, with fewer and thus larger (as measured in mean-free-paths) subdomains, the effect of subdomain size on convergence is diminished. Since the threads share the same "read only" arrays, the MPI-SMP code can use memory more efficiently to solve the same problem. Finally, fewer processes mean that fewer output files are produced for graphics and problem restart.

We also experimented and were marginally successful with two other threading implementations: compiler auto-parallelization of individual loops and Pthreads applied to subroutines. Auto-

parallelization resulted in high loop overhead because the threads were created on each entrance into a loop and destroyed on each exit. With great care, we were able to get a maximum of ~1.5x speed up in wall clock time when running on 4 CPUs. In the Pthreads implementation, individual subroutine calls within the angle loop were parallelized. Because only one parameter can be passed in a Pthread create call, the arguments in the original routine had to be packaged, passed and unwrapped by each thread - this was very tedious to setup and modify. In addition, Pthread code was extremely difficult to debug due to its cryptic and often incomplete error messages. We initialized the Pthreads so that they "lived" throughout a run, thus minimizing the thread overhead. After much work, we were able to demonstrate a 2x speed up in wall clock time using a Pthread modified TETON on 4 CPUs.

The results presented here suggest several areas for future study. The first is to develop an improvement to the Block-Jacobi iteration on interface intensities to aid in convergence of optically thin problems. In general, single domain convergence rates may not be preserved. As the size of the subdomains becomes small as measured in mean-free-paths, the rate of convergence will slow [Yavuz, 1992]. For our time-dependent calculations, however, photons stream at the speed of light for only the length of the time step. Even when the subdomain size is much less than one mean-free-path, there may not be a significant increase in iteration count if the time step size is small relative to the time it would take for a photon to stream across the subdomain. This is typically the case in order to model the physical processes accurately.

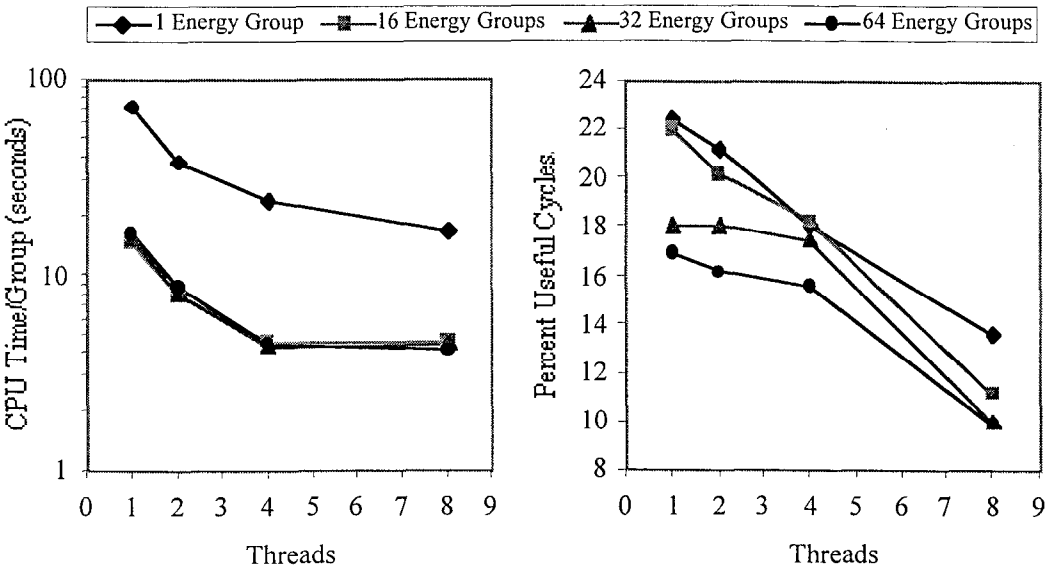


Figure 6a. CPU time vs. threads.

Figure 6b. Useful cycles vs. threads.

There may be situations when slow convergence rates are observed and several remedies exist to address the problem. One approach is the KBA algorithm [Koch, 1992] that directly inverts the transport operator in parallel by carefully ordering the sweep directions within each subdomain. This algorithm is currently limited to orthogonal spatial grids, but several researchers are looking to extend this method to arbitrary grids [Adams, 1998]. Another approach is to use Block-Jacobi iteration, and accelerate the interface fluxes by including a boundary residual source term and solving for additive corrections to the boundary intensities.

Secondly, we are concerned about the scaling of our threading approach to larger numbers of processors per node. The ability to efficiently use larger numbers of processors per node (8 or 16 in the near future) will depend on our ability to improve the data layout and thus prevent "data starvation." With four processors, we observe significant processor "stalls" in the computation because we cannot retrieve data from memory fast enough. With more processors contending for the same memory bus, this situation will worsen. This effect is clearly shown in Fig. 6 where we plot CPU time/energy group and the "useful" cycles [Anderson, 1997] as a function of the number of threads for four different runs. We observe that beyond four processors, we see little reduction in the CPU time/group from the additional processors. This is explained by a substantial reduction in the percent of useful cycles when more than four processors are used. The cycles are being used while waiting for data. Below, we discuss steps that we have already taken to improve data layout and thus use cache more effectively.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[Adams, 1997] Adams, M. L., Subcell Balance Methods for Radiative Transfer on Arbitrary Grids, *Transport Theory Statist. Phys.*, **26**, 385, 1997.

[Adams, 1998] Adams, M. L., Amato, N. M., Nelson, P., Rauchwerger, L., Efficient Massively Parallel Implementations of Modern Deterministic Transport Calculations, a project selected for funding under ASCI ASAP Level II, 1998.

[Anderson, 1997] Anderson, J., Berc, L. M., Dean, J., et. al., Continuous Profiling: Where have all the Cycles Gone?, *Proc. 16th Symposium on Operating System Principles*, 1, 1997. (see also http://www.research.digital.com /SRC/dcpi/new.html)

[APR, 1998] Applied Parallel Research, Inc., P.O. Box 1837, Roseville, CA 95677, 1998. (see also http://www.apri.com/CacheVu.PDF)

[Gropp, 1994] Gropp, W., Lusk, E., Skjellum, A., *Using MPI: Portable Parallel Programming with the Message Passing Interface*, MIT Press, 1994.

[Karypis, 1997] Karypis, G., Kumar, V., METIS User's Guide: A Software Package for Partitioning Unstructured Graphs, Partitining Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, 1997.

[Koch, 1992] Koch, K. R., Baker, R. S., Alcouffe, R. E., Solution of the First-Order Form of the Three-Dimensional Discrete Ordinates Equations on a Massively Parallel Machine, *Trans. Am. Nucl. Soc.*, **65**, 198, 1992.

[Larsen, 1988] Larsen, E. W., A Grey Transport Acceleration Method for Time-Dependent Radiative Transfer Problems, *J. Comput. Phys.*, **78**, 459, 1988.

[Lewis, 1984] Lewis, E. E., Miller, Jr., W. F., *Computational Methods of Neutron Transport*, Wiley & Sons, Inc., 1984.

[OpenMP, 1997] OpenMP Architecture Review Board, OpenMP FORTRAN Application Program Interface, 1997.

[Pomraning, 1973] Pomraning, G. C., *The Equations of Radiation Hydrodynamics*, Pergamon, Oxford, 1973.

[Ramone, 1997] Ramone, G. L., Adams, M. L., Nowak, P. F., Transport-synthetic Acceleration methods for Transport Iterations, *Nucl. Sci. Eng.*, **125**, 257, 1997.

[Yavuz, 1992] Yavuz, M., Larsen, E. W., Iterative Methods for Solving x-y Geometry S_N Problems on Parallel Architecture Computers, *Nucl. Sci. Eng.*, **112**, 32, 1992.