Parallelization of the Three-Dimensional Transport Equation for Boron Neutron Capture Therapy

Eric E. Aubanel
Faculty of Computer Science
University of New Brunswick
Fredericton, New Brunswick
E3B 5A3 Canada
aubanel@unb.ca

Faysal El Khettabi
Laboratory for Threat Material Detection
University of New Brunswick
Fredericton, New Brunswick
E3B 5A3 Canada
faysalek@unb.ca

Abstract

We propose an asynchronous parallel algorithm for the linear Boltzmann transport equation in three dimensions, for Boron Neutron Capture Therapy (BNCT) radiation therapy planning, which can be implemented efficiently on shared memory parallel computers. The three-dimensional multigroup discrete ordinates transport equation is cast into a set of coupled two-dimensional equations, and discretization is accomplished on a grid of arbitrarily-shaped prismatic cells. This allows the 3D multigroup discrete ordinates transport equation to be solved using spatial parallel techniques originally developed in a recent work for the 2D discrete ordinates transport equation by using domain decomposition techniques on an unstructured triangular mesh. We use the Single Program Multiple Data approach to formulate a shared memory parallel numerical implementation with OpenMP. Our results demonstrate that the parallel version of the 3D deterministic algorithm yields good parallel efficiency.

Key words: *neutron transport equation, Boron Neutron Capture Therapy, parallel algorithms, OpenMP*

1. Introduction

The Boltzmann transport equation is the most comprehensive mathematical model, and also the most complicated and computationally intensive, for performing calculations in the radiation engineering field and in many areas of science and engineering [2]. For instance, the planning of the radiation treatment of a tumor begins with the creation of a three-dimensional image of the tumor and surrounding healthy tissue, using techniques such as computed tomography or MRI. The treatment planning occurs after the imaging is completed and involves substantial computations using the Boltzmann transport equation to study the radiation penetration in a portion of a patient's body and to derive the dose calculations as a result of excitation and ionization events in tumor and surrounding healthy tissue [1].

The current practice in radiation therapy planning is to use the Monte Carlo method for these calculations [4]. Deterministic methods offer the promise that high order approximation of the solution over the volume of an element could provide high accuracy without the statistical error of Monte Carlo methods, but they are difficult to apply in complex geometries [5, 6, 7, 8]. These methods must be implemented on parallel computers, since clinical use requires turnaround times in minutes.

Discretization of the Boltzmann equation is typically done using the discrete ordinates method for the angular variable, characteristic or discontinous finite element methods for the space variable, and the multigroup method for the energy variable [4]. Several studies on parallelization have been performed for deterministic transport calculations by applying angular and spatial domain decomposition [11]-[15]. Since the transport sweeps over discrete ordinate directions are independent of each other, parallelization over these discrete ordinate directions is trivial. This approach may not be ideal, since the number of the discrete ordinate directions is much less than the number of finite elements, and may be less than the number of processors available. An asynchronous iteration scheme based on parallelization of energy groups was proposed ten years ago [16], but to our knowledge this approach has not been adopted, primarily for reasons of efficiency [14]. An additional rationale for spatial parallelization is that the quality of the mesh is the most critical factor determining the accuracy of the results, which can lead to domains with millions of elements. Partitioning of the discrete ordinate directions or energies requires duplication of the entire mesh on each processor, which may not be possible and is cer-

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)

tainly not efficient. Spatial partitioning can lead to a better memory access pattern and even superlinear speedups [9]. Until recently, parallelization based on spatial domain decomposition has been restricted to rectangular meshes. In the past few years there have been several applications to unstructured meshes: Nowak and Nemanick [14] used a Hybrid MPI/OpenMP implementation of a method using Jacobi iteration, Plimpton *et al.* [15] developed an asynchronous message passing algorithm, and the present authors extended the ideas of Yavuz and Larson [11] to unstructured triangular meshes together with a shared memory SPMD implementation using OpenMP [9]. In the present work, we extend the latter work to three dimensions with multiple energy groups.

The organization of this paper is as follows. In Section 2, we review the multigroup neutron discrete ordinates transport equation for Boron Neutron Capture Therapy (BNCT) and formulate our spatial numerical solution for the three-dimensional transport equation. In Section 3, we discuss our asynchronous parallel algorithm and present results of an implementation with OpenMP in Section 4. Finally, we close with a summary and some conclusions in section 5.

2. Neutron transport equation

The neutron transport equation is a special case of the Boltzmann equation, whereby the highly unlikely collisions among neutrons are ignored, thus rendering the resulting integro-differential equation linear [2]. The multigroup discrete ordinates approximation for G groups and M discrete ordinate directions can be written as [4]:

$$\Omega_m \cdot \bigtriangledown \psi_m^g(\mathbf{r}) + \sigma^g(\mathbf{r})\psi_m^g(\mathbf{r}) = \sum_{g'=g}^{G} \sigma^{s,g' \to g}(\mathbf{r})\phi^{g'}(\mathbf{r}) + Q_{es,m}^g(\mathbf{r}), \quad m = 1, ..., M, \quad g = 1, ..., G, \quad (1)$$

where $\mathbf{r} = (x, y, z)$, $\Omega_m = (\mu_m, \nu_m, \xi_m)$ is the unit vector along the discrete ordinate direction m having the corresponding weight ω_m, Q_{es} is an external source of neutrons, ψ is the angular neutron flux, σ^g and $\sigma^{s,g' \to g}$ are group constants(cross-sections) and ϕ^g is the scalar flux at energy group g:

$$\phi^g(\mathbf{r}) = \sum_{m=1}^{M} \omega_m \psi_m^g(\mathbf{r}). \quad (2)$$

To model neutron transport for BNCT we place the source of neutrons only in the highest energy group. The dose deposited to the material can be computed from the scalar flux ϕ^g.

In this work we use the same multigroup method as in [3], where the energy partition into G groups was defined by:

$$\{\mathbf{E}_0, \frac{\mathbf{E}_0}{\alpha_{max}}, \frac{\mathbf{E}_0}{\alpha_{max}^2}, ..., \frac{\mathbf{E}_0}{\alpha_{max}^G}\}, \quad (3)$$

where α_{max} is a positive constant less than one which is related to the atomic mass number of the material and $\mathbf{E}_0$ is a given energy. Equation 1 can be solved by using back substitution, beginning with the highest energy of the energy partition $\mathbf{E}_0/\alpha_{max}^G$.

For the purpose of spatial discretization we assume that the domain $\mathbf{D}$ has the form

$$\mathbf{D} = \mathbf{D}_{x,y} \times (\mathbf{a}, \mathbf{b}),$$

where $\mathbf{D}_{x,y}$ is a two dimensional polygonal domain. Let $\mathbf{a} = z_0 < z_1 < z_2 < ... < z_N = \mathbf{b}$ be a subdivision of $(\mathbf{a}, \mathbf{b})$ and $\Delta z_i = z_{i+1} - z_i$. After an integration on the interval (z_i, z_{i+1}) and specification of boundary conditions, equation 1 can then be written for positive ξ_m (indices and constants can be adjusted for negative ξ_m) as:

$$\mu_m \frac{\partial \psi_{i+\frac{1}{2},m}^g}{\partial x}(x,y) + \nu_m \frac{\partial \psi_{i+\frac{1}{2},m}^g}{\partial y}(x,y) + (\sigma_{i+\frac{1}{2}}^g(x,y) + \frac{\xi_m}{\mathbf{c}\Delta z_i})\psi_{i+\frac{1}{2},m}^g(x,y) = \sigma_{i+\frac{1}{2}}^{s,g\to g}(x,y)\phi_{i+\frac{1}{2}}^g(x,y) + Q_{i+\frac{1}{2},m}^g(x,y) + \frac{\xi_m}{\mathbf{c}\Delta z_i}\psi_m^g(x,y,z_i), \quad (4)$$

$$\psi_{i+\frac{1}{2},m}^g(x,y) = \psi_{in,i+\frac{1}{2},m}^g(x,y), \quad \mu_m n_x(x,y) + \nu_m n_y(x,y) < 0, i = 0, 1, ..., N_z - 1,$$

the notation $f_{i+\frac{1}{2}}$ refers to

$$f_{i+\frac{1}{2}}(x,y) = \frac{1}{\Delta z_i}\int_{z_i}^{z_{i+1}} f(x,y,z)dz, \quad (5)$$

and we have supplement the system by assuming for the flux ψ_m^g the following relation,

$$\psi_{i+\frac{1}{2},m}^g(x,y) \approx \mathbf{c}\psi_m^g(x,y,z_{i+1}) + (1-\mathbf{c})\psi_m^g(x,y,z_i), \quad (6)$$

where the quadrature formula coefficient $\mathbf{c}$ is a real in $(0,1)$, the function ψ_{in} is a given function on the boundary of $\mathbf{D}$, and $n(x,y)$ is the outer unit normal on $\partial\mathbf{D}_{xy}$ at (x,y). Equation 4 can be solved by the source iteration scheme, in which one iterates on the scattering source ϕ^g [9]. Note that the term Q in equation 4, for non fission multigroup computation can be split into two terms, an external source Q_{es} and the source due to scattering from higher energies

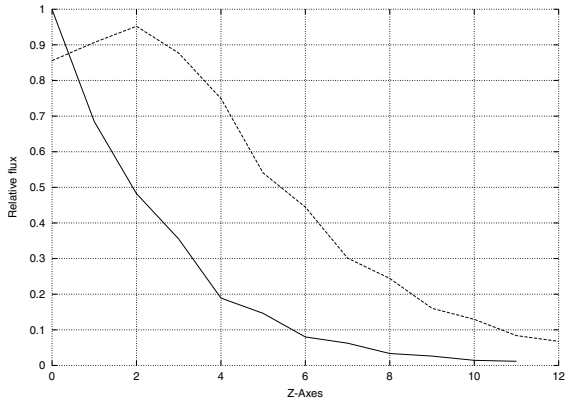


Figure 1. Relative epithermal (solid line) and relative thermal (dashed line) total neutron scalar flux along z axis with a source along z axis

to lower energies:

$$Q^g_{i+\frac{1}{2},m}(x,y) = Q^g_{es,i+\frac{1}{2},m}(x,y) + \sum_{g'=g+1}^{G} \sigma^{s,g'\to g}(x,y)\phi^{g'}_{i+\frac{1}{2}}(x,y), \qquad (7)$$

where $\phi^{g'}_{i+\frac{1}{2}}(x,y)$ is the scalar flux at energy g'.

2.1. BNCT Model Problem

As epithermal neutron beams can come only from an exterior source such as nuclear reactors or charged particle accelerators, the given epithermal neutron source function is on the surface of the domain. The spatial domain is a cylindrical phantom-like head that models the brain only. We triangulate $\mathbf{D}_{x,y}$ by means of a uniform set of x triangles and $x^{1/2}$ subdivisions of $(\mathbf{a},\mathbf{b})$. An S_8 angular quadrature approximation, which spans 40 angles in the plane, is used for the angular variable. We use the step discontinuous finite element method for the spatial flux evaluations with 0.5 as value for quadrature formula coefficient $\mathbf{c}$.

As we assume that the most abundant elements in the head are Hydrogen, Oxygen, Carbon and Nitrogen, the value of α_{max} is that of Oxygen (0.78). This gives 36 energy groups (see equation 3) to simulate the distribution of neutrons between the epithermal (10 kev) and thermal (1 ev) energies. Epithermal neutrons easily penetrate tissue and in so doing slow down and become thermal neutrons, as shown in figure 1 for results from the numerical solution of equation 4.

Execution time was more than 13 hours on a 400 MHz MIPS R12000 processor using a mesh of 15,580 triangles for the $\mathbf{D}_{x,y}$ domain. Reducing this to under a few minutes to make it feasible for use in BNCT treatment planning can only be done with implementation on a parallel computer. We present in the following section a parallel algorithm and implementation.

3. Parallel Source Iteration

We introduce an asynchronous parallel algorithm and an implementation on a shared memory parallel computer.

The source iteration method to resolve the three-dimensional multigroup discrete ordinates transport equation requires transport sweeps across the z-axis and in the domain $\mathbf{D}_{x,y}$ for each discrete angular direction $(\mu_m, \nu_m, \xi_m = \pm\sqrt{1-\mu_m^2-\nu_m^2})$ (see equation 4).

The sweep along the z-axis depends on the sign of ξ_m. When ξ_m is negative, the sweep is from the top to the bottom otherwise the sweep is from the bottom to the top. If ξ_m is positive (and if not, the system indices and constants are adjusted accordingly), the inflow flux $\psi^g_m(x,y,z_{i+1})$ for slice z_{i+1} is computed by using

$$\psi^g_m(x,y,z_{i+1}) = \frac{1}{c}\psi^g_{i+\frac{1}{2},m}(x,y) - \frac{(1-c)}{c}\psi^g_m(x,y,z_i), \qquad (8)$$

where the flux $\psi^g_{i+\frac{1}{2},m}$ between the slice z_i and slice z_{i+1} is calculated by solving equation 4. This requires transport sweeps over the domain $\mathbf{D}_{x,y}$ in the M discrete ordinate directions.

A triangulation $\mathcal{T}_h$ is established over the domain $\mathbf{D}_{x,y}$, i.e, the set $\mathbf{D}_{x,y}$ is subdivided into a finite number of triangles T of dimension h. For each direction (μ_m, ν_m), we partition the mesh into layers $S^m_0, S^m_1, ..., S^m_{end}$:

$$S^m_0 \equiv \{T \in \mathcal{T}_h : \partial^m_{\text{in}} T \subset \partial^m_{\text{in}} \mathbf{D}_{x,y}\}, \qquad (9)$$

$$S^m_{i+1} \equiv \{T \in \mathcal{T}_h : \partial^m_{\text{in}} T \subset \partial^m_{\text{in}} (\mathbf{D}_{x,y} - \bigcup_{j\le i} S^m_j)\}, \quad i = 0,1,...,end, \qquad (10)$$

where $\partial^m_{in}\mathbf{D}_{x,y}$ is the inflow boundary of $\mathbf{D}_{x,y}$ and $\partial^m_{in}T$ is the inflow boundary of the triangle T.

With this partition of $\mathcal{T}_h$, the approximate solution may be obtained in an explicit manner, first in S^m_0, then in S^m_1, etc. The updated exiting fluxes in S^m_i are transferred to the neighboring layer S^m_{i+1} for use as updated incident fluxes.

Within each spatial cell, the approximate solution can only be computed from the incident flux on the inflow boundary and the source of the cell. The updated exiting fluxes are transferred to neighboring spatial cells for use as updated incident fluxes. Thus, the calculation for any

given spatial cell depends on the calculations in all "upwind" cells. This approach represents an inherently sequential procedure which can only be parallelized at the cost of slower convergence of the solution. In order to overcome this obstacle, we extend our previous work on the two-dimensional neutron transport problem [9]. The domain $\mathbf{D}_{x,y}$ is decomposed into P subdomains $\mathbf{D}^p_{x,y}$ using the multilevel implementation of Recursive Spectral Bisection (RSB) [10]. Mathematically $\mathbf{D}$ can be written as:

$$\mathbf{D} = \bigcup_p \mathbf{D}^p_{x,y} \times (\mathbf{a}, \mathbf{b}). \tag{11}$$

This results in a decomposition of $\mathbf{D}$ into columns oriented in the z direction, each of which is assigned to a processor.

During each transport sweep across $\mathbf{D}_{x,y} \times (\mathbf{a}, \mathbf{b})$, each processor may require incident fluxes from other processors, which may or may not have been calculated yet. Let $\oplus$ show the need for incident boundary fluxes and let $\otimes$ show the availability of new outgoing fluxes from a neighboring subdomain. Thus, if $\otimes \equiv \oplus$, then we use the new available information; otherwise, we use the older estimates. Thus, introducing an iteration superscript, the parallel source iteration expression of equation 4 can be written, when ξ_m is positive, as :

$$\begin{aligned}
&\mu_m \frac{\partial \psi^{g,(l+1)}_{i+\frac{1}{2},m,p}}{\partial x}(x,y) + \nu_m \frac{\partial \psi^{g,(l+1)}_{i+\frac{1}{2},m,p}}{\partial y}(x,y) \\
&\quad + (\sigma^g_{i+\frac{1}{2},p}(x,y) + \frac{\xi_m}{\mathbf{c}\Delta z_i})\psi^{g,(l+1)}_{i+\frac{1}{2},m,p}(x,y) \\
&\quad = \sigma^{s,g\to g}_{i+\frac{1}{2},p}(x,y)\phi^{g,(l)}_{i+\frac{1}{2},p}(x,y) + \\
&\quad Q^g_{i+\frac{1}{2},m,p}(x,y) + \frac{\xi_m}{\mathbf{c}\Delta z_i}\psi^g_{m,p}(x,y,z_i),
\end{aligned} \tag{12}$$

$$\psi^{g,(l+1)}_{i+\frac{1}{2},m,p}(x,y) = \begin{cases} \psi^g_{in,i+\frac{1}{2},m}(x,y) & \text{if } (x,y) \in \Gamma_p, \\ & \mu_m n_x(x,y) + \nu_m n_y(x,y) < 0 \\ \psi^{g,(l+1)}_{i+\frac{1}{2},m,p'}(x,y) & \text{if } (x,y) \in \Gamma_{p,p'}, \otimes \equiv \oplus, \\ \psi^{g,(l)}_{i+\frac{1}{2},m,p'}(x,y) & \text{if } (x,y) \in \Gamma_{p,p'}, \otimes \neq \oplus, \end{cases} \tag{13}$$

where $n(x,y)$ is the outer unit normal on the part of the boundary of $\mathbf{D}^p_{x,y}$, $\Gamma_p = \partial \mathbf{D}^p_{x,y}$, that coincides with the outer boundary of $\mathbf{D}_{x,y}$, and $\Gamma_{p,p'}$ is the interface between subdomains $\mathbf{D}^p_{x,y}$ and $\mathbf{D}^{p'}_{x,y}$.

At the beginning of a transport sweep, each processor p, $1 \le p \le P$, has estimates for both the incident interface fluxes and the scattering source within the subdomain $\mathbf{D}^p_{x,y} \times (\mathbf{a}, \mathbf{b})$. Processor p executes its task sequentially over the discrete ordinate directions $1 \le m \le M$. For each direction it processes each slice z_i, $1 \le i \le N_z$, sequentially by updating the inflow flux $\psi^g_{m,p}(x,y,z_{i+1})$ at slice $i+1$ using equation 8.

At the end of a transport sweep, new estimates have been computed for exiting fluxes from $\mathbf{D}^p_{x,y} \times (\mathbf{a}, \mathbf{b})$. The updated exiting fluxes are available to neighboring processors p' for use as updated incident fluxes. The processors terminate the task of iteration l by updating the scalar flux for use in iteration $l+1$ by:

$$\phi^{g,(l)}_{i+\frac{1}{2},p}(x,y) = \sum_{m'=1}^{M} \omega_{m'} \psi^{g,(l)}_{i+\frac{1}{2},m',p}(x,y). \tag{14}$$

We first solve equation 12 beginning with the highest energy group G of the energy partition (see equation 3). Then the same equation is solved for the other energies in descending order, taking into account scattering from higher energies to low energies (source term $Q^g_{i+\frac{1}{2},m,p}$ - see equation 7).

3.1. Implementation

Implementation of the numerical solution of equation 12 requires an asynchronous communication pattern. Fluxes must be communicated between processors owning adjacent subdomains during each sweep. Alternatively, only fluxes from a previous iteration could be used (leaving out the $\otimes \equiv \oplus$ boundary condition in equation 13); we call this the synchronous algorithm, since fluxes would only have to be communicated between processors once at the end of each iteration. Problems with asynchronous communication are best implemented on shared memory computers [17], because each processor can access a global address space without the participation of other processors. Therefore we have implemented our algorithm using OpenMP, an industry-wide standard for threads-based shared memory parallelization.

The computational kernel of the code, which contains the source iteration, was placed in one OpenMP parallel region and parallelized using the Single Program Multiple Data (SPMD) approach. That is, we did not use the typical approach taken with OpenMP of parallelizing individual loops. Variables in the parallel region were chosen to be private by default, with the notable exception of the flux ψ and various constants, which were set to be shared.

As discussed above, each processor performs transport sweeps over its subdomain sequentially for each discrete ordinate angle. It is desirable to order the transport sweeps in each subdomain so as to maximize the availability of updated incident fluxes from adjacent subdomains. We have shown that this can result in an improvement in the rate of convergence of the asynchronous algorithm [9]. For each subdomain $\mathbf{D}^p_{x,y}$ and for each direction m, we find the first

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)

layer that intersects $\mathbf{D}^p_{x,y}$(see equations 9,10):

$$n_m = \min_{\mathrm{i}}\{i, S^m_i \cap \mathbf{D}^p_{x,y} \neq \emptyset\}. \quad (15)$$

We then order the directions for transport sweeps in the subdomain $\mathbf{D}^p_{x,y}$ as $m_1, m_2, ..., m_i, m_{i+1}, ..., m_M$, such that $n_{m_i} \leq n_{m_{i+1}}$. This means that while we sweep in a specified direction in processor p, we sweep in a direction in a neighboring processor $p^{'}$ so that both processors are able to use the new outgoing interface boundary fluxes from their neighbors as soon as possible.

This means, for instance, that sweeping in subdomains on the outer boundary of the domain will be done first for directions that come from outside. When incident fluxes are required from another subdomain they are obtained (updated or not) transparently from global memory.

3.2. Convergence

Since the parallel algorithm breaks the dependency of the transport sweeps, convergence will be slower than the serial algorithm. We showed previously for neutron transport in two dimensions that this problem can be mitigated somewhat by the use of our asynchronous algorithm (see equations 12-13), resulting in convergence in a little more than half the number of iterations as the synchronous algorithm. There still remains the fact that in many cases, old boundary subdomain fluxes will still be used ($\otimes \neq \oplus$). This occurs for approximately 40% of triangles that lie on $(x, y) \in \Gamma_{p,p'}$, for the two dimensional problem studied in [9]. As a result, convergence of the asynchronous parallel algorithm is still slower than for the sequential algorithm. Importantly, the number of iterations did not increase rapidly with the number of processors used, reaching a plateau of twice the number of iterations compared to the serial algorithm.

The convergence of the parallel algorithm is sensitive to the error introduced by using old boundary subdomain fluxes, particularly during the first iteration, when there are no old fluxes available. These old fluxes would normally be set to an initial guess solution, for example to zero, before the first iteration. This situation would be expected to be aggravated in our approach to the three-dimensional problem, since the error resulting from the domain decomposition would propagate in the z direction through the $\psi^g_{m,p}(x, y, z_i)$ term in equation 12.

However, it is possible to increase the convergence of our multigroup parallel algorithm. Since the problem is solved by using back substitution computation from higher to lower energies, one can use the calculated flux $\psi^{g+1}_{i+\frac{1}{2},m}$ at energy $g+1$ as an initial guess solution $\psi^{g,(0)}_{i+\frac{1}{2},m}$ to compute the flux at energy g:

$$\psi^{g,(0)}_{i+\frac{1}{2},m}(x, y) = \psi^{g+1}_{i+\frac{1}{2},m}(x, y) \quad if \quad g < G \quad (16)$$

For ψ continuous over the energy interval $(\mathbf{E}_0, \mathbf{E}_0/\alpha^G_{max})$, the error introduced by the domain decomposition can be reduced by this procedure.

4. Results

Our algorithm was implemented in Fortran 90, and results were obtained on an SGI Origin 3800 with 64 MIPS R12000 400 MHz processors and 32 GB memory. The Origin 3800 is a distributed shared memory machine, with a "first-touch" data-placement policy. This means that pages are allocated to memory close to the processor that runs the code. Therefore, the shared array containing the flux $\psi^g_{i+\frac{1}{2},m,p}(x, y)$ was initialized in parallel to maximize local memory references. All the processors initiate their work simultaneously after one processor initializes the input data for all. At the end of every iteration a global L_2 relative error norm is calculated for the scalar fluxes:

$$\epsilon = \frac{\|\phi^{(l+1)}_{i+\frac{1}{2}} - \phi^{(l)}_{i+\frac{1}{2}}\|_2}{\|\phi^{(l)}_{i+\frac{1}{2}}\|_2}. \quad (17)$$

This is the only serial part of our algorithm, in addition to the initialization described above and any writing of fluxes to disk. If the convergence criterion $\epsilon < 10^{-7}$, then the program proceeds to the next lower energy, after having calculated new source terms.

All tests are carry out on the BNCT model. We considered two different neutron sources impinging on a cylinder of 10 cm radius and 20 cm height: (i) incident at a normal to the center of the top of the cylinder with a circular cross-sectional radius of 2 cm; (ii) incident on the center of the side of the cylinder, with a square cross section of 4 cm wide and 2 cm high.

We also considered two ways of initializing the flux for energies lower than E_0/α^G_{max}, as discussed in the previous section: using zero as initial guess solution or reusing fluxes from the previous highest energy computation.

4.1. Convergence Rate

The sequential source iteration program required 5 iterations per energy group to converge, for a total of 180 iterations over all energies. The number of iterations increased with the number of processors for the parallel source iteration program, but reached a plateau after 8 or 16 processors. The sum of the iterations required to converge over all energy groups is shown in figure 2 as a function of the number

0-7695-1926-1/03/$17.00 (C) 2003 IEEE
Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)

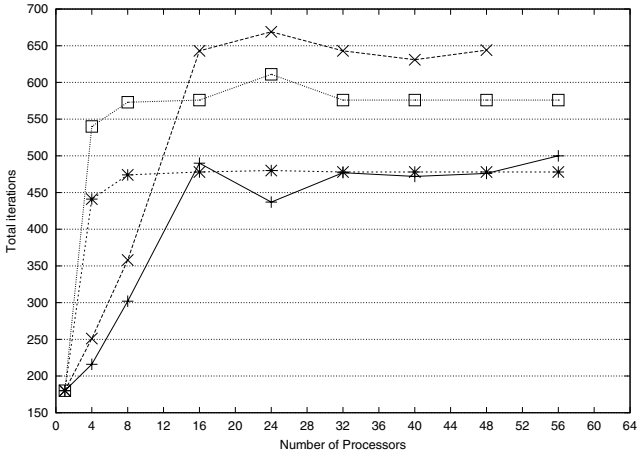
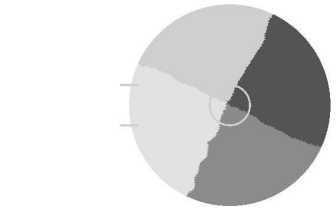
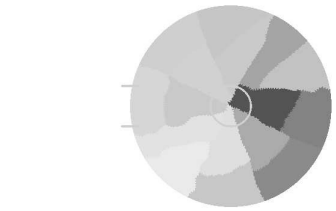


Figure 2. Total number of iterations (over all energies) for source on the side with flux reuse (+) and without (×) flux reuse between energy groups, and for source on top with flux reuse (∗) and without (□) flux reuse.



(a)



(b)

Figure 3. Domain decomposition for (a) four and (b) sixteen processors, showing the two locations of the neutron source)

of processors, for both types of neutron sources, and for both types of flux reuse.

Consider first the effect of reusing fluxes from a higher energy for the initial guess. When fluxes are not reused, the initial guess is set to zero for every energy, and the number of iterations required to converge increases more rapidly and reaches a higher plateau than when they are reused. Consider in detail the case when the source is on the top of the cylinder, and 32 processors are used. If fluxes are not reused, convergence is achieved in 16 iterations for each energy. If they are reused, the convergence occurs in 16 iterations for the highest energy, then in 19 iterations for the second highest, followed by 14 iterations for the third highest and in 13 iterations for the rest. This pattern is typical, and can be explained as follows. Since the external source occurs only at the highest energy (group G), the resulting fluxes are quite different than for the other energies (see figure 1), therefore the flux for group G is not a good initial guess for the flux of group $G-1$, and is worse than simply initializing the flux to zero. The fluxes for all energies other than group G are more similar, since there is no external source, therefore reusing them results in a better initial guess when inter-subdomain fluxes are required and consequently leads to a decrease in the number of iterations required to converge. All further discussion will refer to calculations where the fluxes are reused, unless otherwise specified.

Consider next the effect of changing the geometric location of the neutron source. The number of iterations rises much more rapidly for the source on the top than for the source on the side. The reason for this can be seen by looking at how the source intersects with the domain decomposition. Figure 3 shows the domain decomposition for 4 and 16 processors, together with the location of the source. For the source on the side, neutrons travel initially in only one subdomain, but for the 16-processor decomposition, they reach other subdomains sooner. Since the flux goes to zero rapidly with distance (see figure 1), it stays mostly in one subdomain for the 4-processor decomposition, whereas it is spread over several subdomains for the 16-processor decomposition. Therefore convergence occurs in only 6 iterations in the former case, but in 13 iterations (for most energies) in the latter case. Since the source on the top immediately couples multiple subdomains for all decompositions, the number of iterations rises rapidly starting at 4 processors, with typically 13 iterations required to converge per energy group.

4.2. Parallel Speedup

The speedups obtained by our asynchronous parallel algorithm are shown in figure 4. Timings were averaged over all iterations and energies. The speedup per iteration was calculated as $S_p = t_s/t_p$, where t_s and t_p are the execution times per iteration of the sequential and parallel (using p processors) programs. Note that this speedup is superlinear, that is its slope is greater than one, for fewer than 16 processors, and slightly superlinear for 16 processors or more.

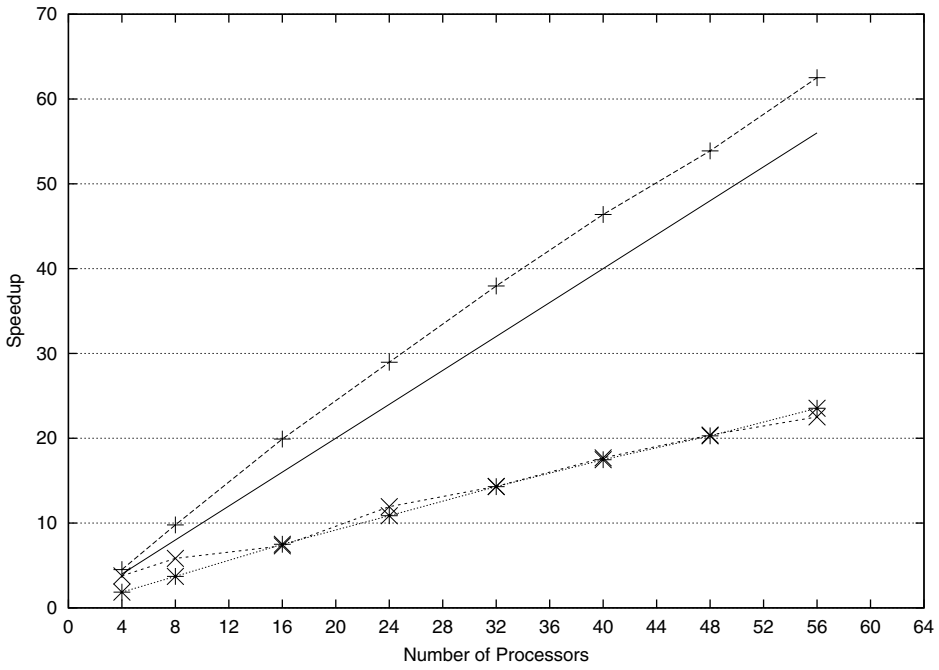


Figure 4. Parallel speedups per iteration ($+$) and total speedups for source on side ($\times$) and on top ($*$)

One possible explanation is that the shared array containing the flux requires 1.5 GB storage, which is too large to fit in a single 2-processor node of the Origin 3800 (1 GB). Therefore the serial program must perform a number of memory accesses to a remote node; if the data is layed out in such a way that the memory accesses are localized as much as possible, then the parallel program may have fewer remote memory accesses. However, upon investigation with SGI's Perfex performance monitor, it became clear that the cause lay in the better cache reuse of the parallel programs. This is evident from the L2 cache hit rate (fraction of data accesses satisfied from the L2 cache), which was 0.934 for the serial program, 0.965 for four processors, and 0.976 for greater than four processors. This is an effect common to many memory-bound SPMD parallel programs, where explicit allocation of data with processors leads to better cache reuse.

The overall speedup, given by $(N_{iter,s}/N_{iter,p})(t_s/t_p)$, where $N_{iter,s}$ and $N_{iter,p}$ are the number of iterations required to converge for the serial and parallel calculations respectively, is also shown in figure 4, for both neutron sources. The decrease in the convergence rate of the parallel program offsets to some extent the speedups obtained per iteration. However, the overall speedup still increases with the number of processors, and the execution time is reduced to 40 minutes with 48 processors.

5. Conclusions

The formulation of the three-dimensional transport equation as coupled two-dimensional equations allowed us to extend previous work on the parallelization of the transport equation [9]. The proposed asynchronous parallel algorithm is shown to be viable, and we observe no significant degradation in the convergence rates as has been reported [11, 12, 13].

Implementation of our asynchronous parallel algorithm was done using OpenMP. Our results demonstrate that SPMD parallelization together with OpenMP can yield excellent speedup. This was achieved on the SGI Origin 3800, a cc-NUMA machine, by taking advantage of SGI's "first touch" data placement policy and parallel intialization of shared arrays. Our parallel implementation does require more code modification than the loop-level approach, which is usual for OpenMP programs, but it is still far easier than a message-passing implementation.

We achieved superlinear speedups per iteration due to cache effects. Convergence of the parallel program took more iterations than the serial program, but this increase reached a plateau of 2.5 times the number of serial iterations. This is comparable to the relative slowdown in the parallel convergence rate found in our previous results for the two-dimensional mono-energetic Boltzmann equation. The rate of convergence was found to be sensitive to the location of the source of neutrons. Convergence was im-

proved by reusing fluxes between calculations for each energy group, which reduced the error on the first iteration of the parallel programs.

Acknowledgments

Our work made use of the infrastructure and resources of MACI (Multimedia Advanced Computational Infrastructure) funded in part by the CFI (Canada Foundation for Innovation) and the University of Alberta.

References

[1] IAEA, "Curent Status of Neutron Capture Therapy". IAEA-TECDOC-1223 (May 2001)

[2] C. Cercignani, "The Boltzmann equation and its applications", Springer-Verlag, New York (1988).

[3] M. S. Clowdsley, J. H. Heinbockel, H. Kaneko et.al, "A Comparison of the Multigroup and collocation Methods for Solving the Low Energy Neutron Boltzmann Equation", *Canadian Journal of Physics*, **78** pp. 45-56 (2000).

[4] E. E. Lewis and W. F. Miller, "Computational Methods of Neutron Transport", John Wiley and Sons, New York (1984), reprinted by the American Nuclear Society, La Grange Park (1993).

[5] P. Lesaint, P.A. Raviart, "On a Finite Element Method for Solving the Neutron Transport Equation", in *Mathematical Aspects of Finite Elements in Partial Differential Equations*, **89**, Academic Press, New York (1974).

[6] G.R. Richter, "An Optimal-Order Estimate for the Discontinuous Galerkin Method", *Math. Comput.*, **50**, 75 (1988).

[7] F.E. Khettabi, C. Lécot, "Characteristic Methods for Discretizing the two-Dimensional Transport Equation on an Unstructured Grid of Triangular Cells", *Proc. Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications*, **2**, pp. 975-984., American Nuclear Society, La Grange Park (1997).

[8] F. E. Khettabi, " Exponential characteristic Methods for Discretizing the Two-Dimensional Transport Equation on an Unstructured Grid of Triangular Cells", *International Conference on the physics of nuclear science and technology*, (1998)

[9] E. E. Aubanel and F. E. Khettabi, "Parallelization of Radiation Transport on Unstructured Triangular Grids with Spatial Decomposition and OpenMP", Workshop on Parallel and Distributed Scientific and Engineering Computing with Applications, in *proc. of the 16th Intl. Parallel and Distributed Processing Symposium*, IEEE (2002).

[10] Stephen T. Barnard and Horst D. Simon, "A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems", *Concurrency: Practice and Experience*, **6**, pp. 101-107 (1994).

[11] M. Yavuz and E. Larson,"Iterative Methods for Solving $X - Y$ Geometry S_n Problems on Parallel Architecture Computers", *Nucl. Sci. Eng.*, **111**, 46, (1992).

[12] S.P. Burns and M.A. Christon, "Spatial Domain-Based Parallelism in Large-Scale, Participating Media, Radiative Transport Applications", *Numerical Heat Transfer*, Part B, **31**, 401 (1997).

[13] J. Goncalves and P.J. Coelho, "Parallelization of the Discrete Ordinates Method", *Numerical Heat Transfer*, Part B, **32**, 151 (1997).

[14] P. Nowak and M.K. Nemanic, "Radiation Transport Calculations on Unstructured Grids using a Spatially Decomposed and Threaded Algorithm", *Proc. ANS Conf. on Math. and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications*, 379 (1999).

[15] S. Plimpton, B. Hendrickson, S. Burns, W. McLendon III, "Parallel Algorithms for Radiation Transport on Unstructured Grids", *Proc. SuperComputing 2000*, IEEE (2000).

[16] R. Hiromoto, B.R. Wienke, R.G. Brickner, "The Performance of asynchronous iteration schemes applied to the linearized Boltzmann transport equation", *Parallel Computing*, 18, pp. 241-268 (1992).

[17] S.M. Pancake, "Is Parallelism for You?", *Computational Science and Engineering* Vol. 3, No. 2, 18 (Summer, 1996).