LA-UR- 96-4099

CONF-961249--2

Title: DANTSYS/MPI - A SYSTEM FOR 3-D DETERMINISTIC TRANSPORT ON PARALLEL ARCHITECTURES

Author(s): R.S. Baker
R.E. Alcouffe

Submitted to: OECD/NEA Meeting, 2 - 3 December 1996, Paris, France

DISCLAIMER

Los Alamos
NATIONAL LABORATORY

Form No. 836 R5
ST 2629 10/91

DANTSYS/MPI - A SYSTEM FOR 3-D DETERMINISTIC TRANSPORT ON PARALLEL ARCHITECTURES

Randal S. Baker (rsb@lanl.gov) and Raymond E. Alcouffe (rea@lanl.gov)
Transport Methods Group, MS B226
Los Alamos National Laboratory
Los Alamos, NM 87545

Introduction

Since 1994, we have been using a data parallel form of our deterministic transport code DANTSYS [1] to perform time-independent fixed source and eigenvalue calculations on the CM-200's at Los Alamos National Laboratory (LANL). Parallelization of the transport sweep is obtained by using a 2-D spatial decomposition which retains the ability to invert the source iteration equation in a single iteration (i.e., the diagonal plane sweep). We have now implemented a message passing version of DANTSYS, referred to as DANTSYS/MPI, on the Cray T3D installed at Los Alamos in 1995. By taking advantage of the SPMD (Single Program, Multiple Data) architecture of the Cray T3D, as well as its low latency communications network, we have managed to achieve grind times (time to solve a single cell in phase space) of less than 10 nanoseconds on the 512 PE (Processing Element) T3D, as opposed to typical grind times of 150-200 nanoseconds on a 2048 PE CM-200, or 300-400 nanoseconds on a single PE of a Cray Y-MP. In addition, we have also parallelized the Diffusion Synthetic Accelerator (DSA) equations which are used to accelerate the convergence of the transport equation. DANTSYS/MPI currently runs on traditional Cray PVP's and the Cray T3D, and it's computational kernel (Sweep3D) has been ported to and tested on an array of SGI SMP 's (Symmetric Memory Processors), a network of IBM 590 workstations, an IBM SP2, and the Intel TFLOPs machine at Sandia National Laboratory. This paper describes the implementation of DANTSYS/MPI on the Cray T3D, and presents a simple performance model which accurately predicts the grind time as a function of the number of PE's and problem size, or scalability. This paper also describes the parallel implementation and performance of the elliptic solver used in DANTSYS/MPI for solving the synthetic acceleration equations.

The Diagonal Plane Sweep

In [2], we discuss our implementation of the diagonal plane sweep. To summarize, the transport operator on the left hand side of the first-order form of the source equation, i.e.,

$$\vec{\Omega} \bullet \nabla \Psi(\vec{r}, E, \vec{\Omega}) + \sigma_T \Psi(\vec{r}, E, \vec{\Omega}) = S(\vec{r}, E, \vec{\Omega}) \tag{1}$$

may be viewed as a lower diagonal matrix. Thus, given a known source on the right (the result of inner

and outer iterations over scattering and fission sources), the source iteration equation may be solved for the angular flux exactly in one iteration by performing an ordered sweep for each S_N direction. While this sweep is an inherently sequential operation, there actually exists a 2-D diagonal plane of cells which can be solved simultaneously, i.e., in parallel, when sweeping directions in a given octant (see Fig. 1). We map this diagonal plane onto a 2-D processor mesh, thus achieving 2-D parallelism for 3-D calculations.

Unlike 3-D spatial decompositions which require iterations to solving the source iteration equation, our 2-D spatial decomposition retains the ability to invert this equation in a single sweep, an important advantage in neutronics calculations where the mesh cells are often optically thin. However, as can also be seen from Fig. 1, the diagonal plane does not completely fill all the processor mesh when the diagonal plane is near the corners, resulting in a loss of Parallel Computational Efficiency (PCE), where PCE is defined to be the amount of useful work performed in a single sweep divided by the amount of total work. For a large cubic mesh, the PCE can be shown to be only 33%. Note that this does not mean that the code is only 33% parallel, but that 67% of the cells solved (in parallel) are actually dummy, or "ghost", cells. However, the PCE can be raised by reordering the data so that the sweep for the next discrete direction (Method 1, Successive in Angle) (see Fig. 1) or octant (Method 2, Simultaneous in Angle) is initiated by a processor as soon as it completes the previous angle/octant, while the downstream processors continue to work on the previous angle/octant. Using these variants, the PCE for large cubic meshes improves to 86% for Method 1 (S_6) and 50% for Method 2. Although Method 1 has a higher PCE, Method 2 proved to be more computationally efficient on the SIMD (Single Instruction Multiple Data) CM-200 due to the overhead costs of gather/scatter and shift operations on this machine [2]. Thus, Method 2 was implemented in our production code DANTSYS (THREEDANT) on the CM-200 in 1994.

Implementation of the Diagonal Plane Sweep on a T3D

Due to the SIMD nature of the CM-200, the diagonal plane sweep was performed over the entire $I \times J \times K$ sized problem mesh. However, on the Cray T3D, we can improve the PCE of the diagonal plane sweep by taking advantage of its SPMD architecture [3]. Let $N_J \times N_K$ represent the 2-D processor mesh onto which the $J \times K$ problem mesh is mapped such that the largest J and K mesh dimensions on any PE are $J_C = [J/N_J]$ and $K_C=[K/N_K]$, where $[x]$ is the ceiling function equal to the smallest integer larger than or equal to x. Let I_C represent the number of I-planes to be solved by a PE before communicating the resulting edge fluxes to the downstream PE's. Then, while we continue to use the diagonal plane sweeping algorithm over the space $I/I_C \times N_J \times N_K$, we use a simple sweep along rows to solve the balance equation within a PE for all cells in a "chunk" of size $I_C \times J_C \times K_C$. We refer to this method as Method 3 (Simultaneous in Angle + Block Sequential). Method 3 results in a PCE of

$$PCE = \frac{I \times J \times K \times M \times 2}{(N_J + N_K - 2 + I/I_C \times M \times 2)(I_C \times J_C \times K_C \times N_J \times N_K)} \qquad (2)$$

where M is the number of angles per octant. The numerator in Eq. (2) is the number of phase space cells solved in one sweep of the mesh for one pair of octants (quadrant), while the first term in the denominator is the number of steps to sweep through the diagonal plane space for all angles in an quadrant, and the second term is the number of phase space cells solved at each step. The first term accounts for the loss in parallel efficiency due to the diagonal plane sweep over the PE space, and the second term accounts for

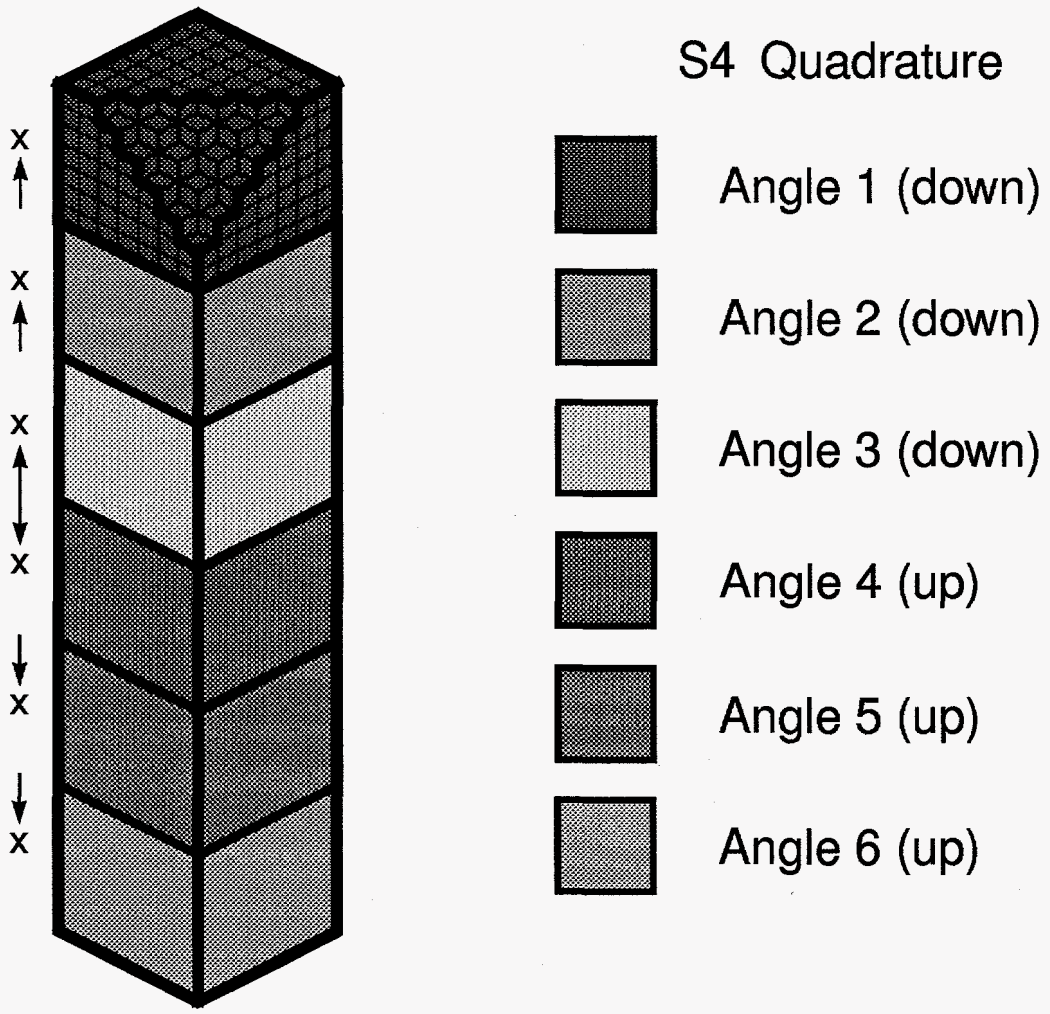

Figure 1. The Diagonal Plane Sweep

the loss in parallel efficiency due to load imbalancing. For a 128 x 128 x 128 mesh with an S_6 triangular (M = 6) quadrature on 64 PE's and I_C = 4, Eq. (2) results in a PCE of 96.5%. While I_C = 1 provides the largest possible PCE, we have found that, when communications latencies are considered (see below), the overall computation time is generally minimized with I_C = 4 on the T3D.

A Performance Model for the Diagonal Plane Sweep

Performance models are a necessity on parallel platforms for truly understanding actual versus ideal (linear) speedup, and estimating performance for varying problem sizes and machine configurations. We have constructed such a model for the diagonal plane sweep algorithm in DANTSYS/MPI.

We model the total time per iteration as

$$\begin{aligned} T_{it} = {} & 4 \times (N_J + N_K - 2 + I/I_C \times M \times 2) \\ & (I_C \times J_C \times K_C \times T_{comp} + I_C \times (J_C + K_C) \times (T_{bw} + T_{bnd}) \\ & + T_{lat} + T_{sync} + T_{dp}) \\ & + T_{ser} \end{aligned} \tag{3}$$

where T_{comp} is the average computation time (source moments, angular source, balance equation, and flux moments) per cell, T_{bnd} is the time spent in calculating boundary data, T_{sync} is the average time per diagonal plane step spent in synchronization, T_{lat} is the communications latency, and T_{bw} is the communications time (bandwidth). T_{ser} is the serial overhead per iteration outside of the sweep, while T_{dp} is the serial overhead per each step of the diagonal plane sweep. Note that this model is an approximation in the sense that T_{comp} should be broken out into at least two separate components, one (computation time for balance equation) which is multiplied by the number of diagonal plane steps to sweep through the mesh times the maximum number of spatial cells assigned to a processor, and one (computation time for source moments, angular source, and flux moments) which is a multiplied by only the maximum number of spatial cells assigned to a processor. For simplicity, we use one combined parameter here. Since the time spent in solving the balance equation comprises the majority of the total solution time, and since $I/I_C \times M \times 2 > N_J + N_K - 2$ for the modeled problems, this is a reasonable approximation. Note also that T_{comp} will be a function of problem dependent S_N and P_N orders, while the others will not. T_{comp} will also be a function of the spatial differencing method and the amount of (problem dependent) negative flux fixup, if any, for Diamond Differencing with Set-to-Zero Fixup.

Values for the above parameters were determined by instrumenting the computational kernel Sweep3D of DANTSYS/MPI. Sweep3D is a small test code which performs the diagonal plane sweep plus the inner iteration. Typically over 95% of the (transport) time in DANTSYS/MPI is spent in these two areas, so Sweep3D is a representative model for DANTSYS/MPI. The values determined from Sweep3D were T_{bnd} = 0.31 *u*secs/cell face, T_{bw} = 0.153 *u*secs/cell face, T_{lat} = 11.6 *u*secs, T_{sync} = 11.7 *u*secs, T_{dp} = 12.5 *u*secs, and T_{ser} = 2,575 *u*secs.

As an aside, we used the CRI performance analysis tool Apprentice to examine the 50 x 50 x 50 mesh problem on a single PE. Using the FLOP count from the Apprentice, the single PE performance of Sweep3D for this problem was 15.0 MFLOPs. The peak performance of the DEC Alpha PE used in the

T3D is 150 MFLOPs. However, the Alpha PE used in the T3D has only an 8 KByte Direct Mapped cache, and the transport algorithm in Sweep3D performs only 1.4 FLOPs/Load. Thus, memory bandwidth/latency, not PE speed, is the limiting factor. This is typical of most physics codes with large data sets run on cache-based microprocessors.

Application to Assembly 6

Assembly 6 is a critical assembly consisting of a highly enriched uranium spherical core surrounded by a beryllium-oxide reflector. This assembly is experimentally known to have an eigenvalue of unity. Because of mirror symmetry, only one-eighth of the assembly was actually modeled by the mesh. All calculations were performed using an S_6 product quadrature set ($M = 9$) in conjunction with multigroup ENDF 12-group P_2 prompt neutron cross sections on a 94 x 94 x 94 spatial mesh. The diagonal plane sweep algorithm was implemented in conjunction with diamond differencing in space. No fixup was needed in this problem. The value of the k eigenvalue calculated by DANTSYS/MPI is 0.9965 with a convergence criteria of 0.001.

We first examine the validity of our model. Assuming the values of T_{bnd}, T_{bw}, T_{lat}, T_{sync}, T_{dp}, and T_{ser} determined from Sweep3D are still valid, and examining Eq. (3), we see that Eq. (3) predicts a linear relationship between the total cell computation time and the number of cells solved per PE, with slope T_{comp} and a Y-intercept of zero. Figure 2 shows the result of this prediction when DANTSYS/MPI is used to calculate Assembly 6 for 64 to 512 PE's. We see that the relationship is indeed linear, and that our model is valid when used to predict performance for typical problem sizes on up to hundreds of PE's.

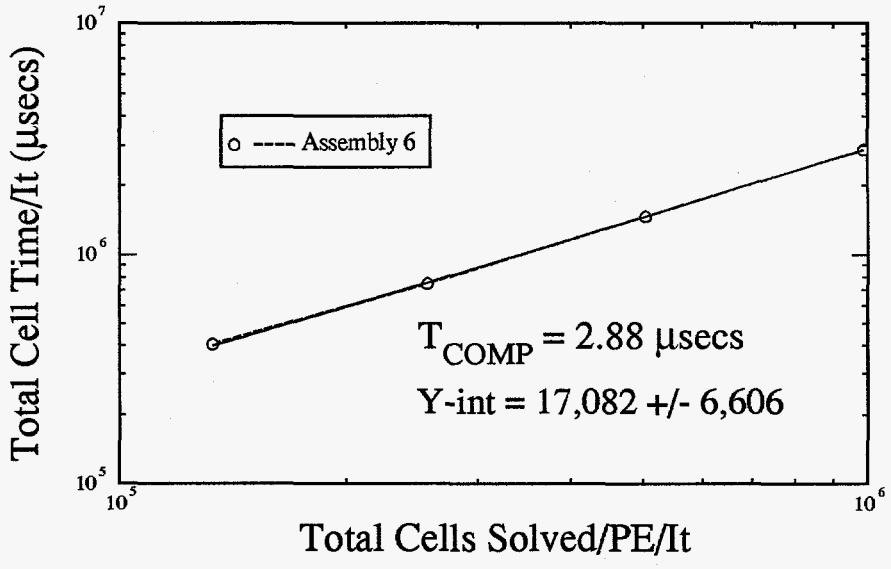

Figure 2. Examination of Model Validity for Assembly 6

Figure 3 shows the actual grind times for the Assembly 6 calculation for 64 to 512 PE's, the grind times

predicted by our model (using T_{comp} = 2.88 *u*secs) for 1 to 512 PE's, and the ideal grind time, assuming a linear speedup from 1 to 512 PE's, where the grind time for 1 PE is determined from our model. As can be seen, the model predicts the actual grind times quite closely, given the parameter T_{comp}, and the total speedup obtained for this realistic application is 335 on 512 PE's. The grind time for this application on a single Cray YMP PE is ~400 nsecs, so the performance actually achieved on 512 T3D PE's is equivalent to that of 47 YMP PE's.

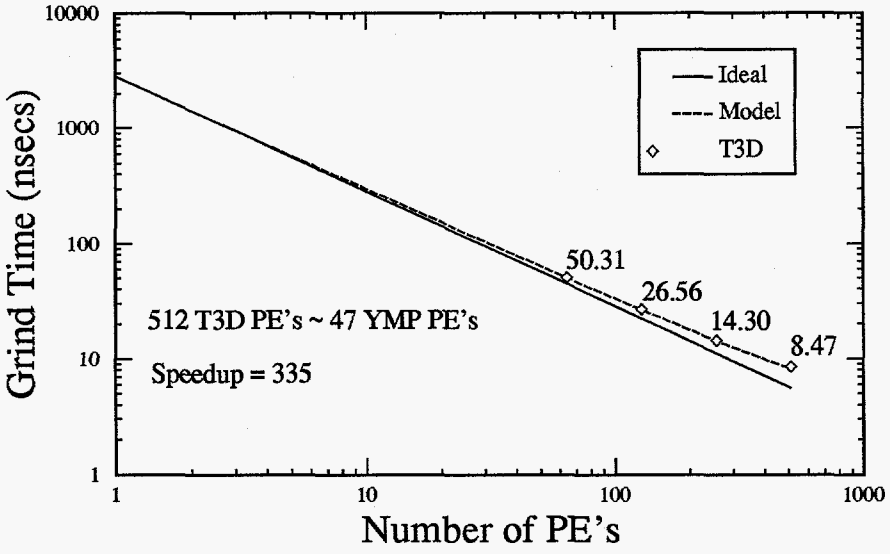

Figure 3. Assembly 6 Grind Times

Extrapolation to the Intel TFLOPs Machine

Sandia National Laboratory is in the process of acquiring a 4096 PE MPP from Intel, where the PE's consist of 2 PentiumPro CPU's with 128 MBytes of memory. Using our model, we can predict the performance of DANTSYS/MPI on this machine with an appropriate adjustment of parameters. Unlike the Cray T3D, where DANTSYS/MPI uses a synchronous data passing communication library (SHMEM), we will use a asynchronous message passing library (MPI) on the Intel machine, so T_{sync} is set to zero. Given the larger cache and faster clock on the Intel machine, we estimate that a single PE performance of 20 MFLOPs (versus 15 on the T3D) should be achievable, so T_{bnd}, T_{dp}, and T_{ser} are set to 3/4ths of the T3D values. Based upon preliminary performance estimates of the communications network, the parameters for T_{lat} and T_{bw} are set to 25 *u*secs and 0.0765 *u*secs, respectively.

The problem we model is an S_6 (M = 6), P_1 calculation on a 256 x 256 x 256 mesh. On the T3D, T_{comp} for this problem is 1.78 *u*secs, so we use 1.335 *u*secs for the Intel machine. Figure 4 presents the results of our model for 1 to 4096 PE's. In addition, Fig. 4 also presents some very preliminary test results from actual Sweep3D calculations on up to 700 PE's of the TFLOPs machine. These initial results are in good agreement with our model predictions. Thus, we have confidence that our predictions of actual speedups

of over 2,300, and performance equivalent to that of over 500 YMP PE's, are achievable.

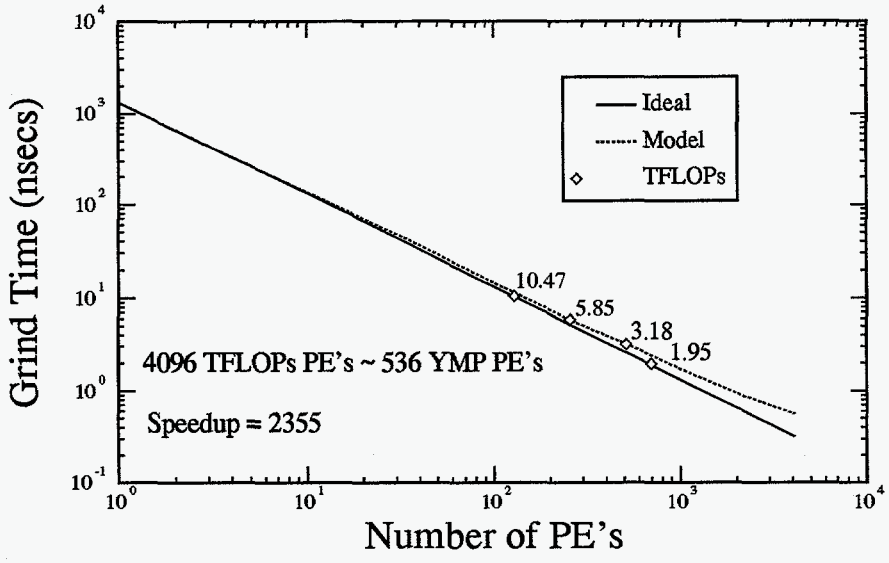

Figure 4. Predicted Performance on the Intel TFLOPs Machine

Iteration Acceleration

Many transport calculations require some means of accelerating convergence in order to be practical. DANTSYS uses Diffusion Synthetic Acceleration (DSA) to accelerate convergence of both the inner and outer iterations. DSA has proven extremely valuable in accelerating convergence for calculations containing highly scattering and/or fissile regions.

DSA works by using a lower-order diffusion-like equation to accelerate the solution of the higher-order transport operator. Typically, one diffusion acceleration step is performed for every transport step. Thus, an efficient means of solving an elliptic equation is essential in reducing the overall computation time. However, in two or three dimensions, the solution of elliptic equations is a non-trivial problem. The multigrid method in conjunction with pre-conditioning by line inversions has proven most successful in the past when used with DANTSYS, but an efficient multigrid scheme has yet to be developed for parallel architectures. Thus we have also implemented a Conjugate Gradient (CG) diffusion solver in DANTSYS/MPI because CG schemes are readily parallelized. Since each PE in DANTSYS/MPI contains all cells in the I-dimension for a given J and K, we precondition by line inversions in the I direction, but use Red/Black iterations over the cells in the J and K dimensions. Although not as effective a pre-conditioner as line inversions in all three dimensions, this scheme is compatible with our 2-D spatial decomposition. Furthermore, this scheme represents an extrinsic decomposition in the sense that the results do not vary as the number of PE's is varied.

Table 1 below presents the results of DANTSYS/MPI calculations (128 PE's) on a benchmark LWR prob-

lem with and without DSA. The LWR benchmark calculation consists of 2 groups, S_8P_0, on a 100 x 100 x 100 mesh. The convergence criteria is 0.001. PBAL is the integral particle balance, TSWEP is the time spent in performing the sweep of the transport equation, and TDSA the time for the diffusion equation and Chebyshev acceleration of the fission source. The total time includes both TSWEP and TDSA, plus time for calculating group sources, balance tables, etc. DSA reduces the number of iterations by over a factor of 36 in this eigenvalue calculation, but the overall run time is only reduced by a factor of six due to the cost of solving the diffusion equation. TSWEP does not decrease by a factor of 36 due to the extra work required to calculate the leakages at every cell face for the DSA equations. We plan to investigate other pre-conditioners that will hopefully prove more effective in solving the CG equations, and thus improve the performance of DSA in DANTSYS/MPI. Additionally, the investigation of effective parallel multigrid solvers is an active field.

Table 1: LWR1 Iteration Acceleration

Acceleration	k	PBAL	No. Outers	No. Inners	TSWEP	TDSA	Total
No DSA	0.9641	4.17e-3	36	295	311.2 secs	1.2 secs	320.4 secs
With DSA	0.9625	1.90e-5	5	8	14.4 secs	38.5 secs	53.7 secs

Code Structure

DANTSYS/MPI is divided into input (serial only), edit (serial only), and solver (serial/parallel modules). Since the input and edit modules are neither CPU or memory-intensive, there is no need to parallelize them. The input module prepares the cross sections, geometry information, and other data required by the solver module. This information is stored in link files on disk, which are then read in by the solver module to perform the actual calculation. The solver module memory requirements are driven by the storage requirements for the flux moments, since we do not need to store the angular fluxes for our 2-D spatial decomposition. In turn, the solver module writes a CCCC-standard RTFLUX file upon completion, which may then be used by the edit module for post-processing. Parallel I/O is performed by PSFLIB, a local CRI product which allows flexible yet efficient I/O to a single file from an arbitrary number of PE's.

DANTSYS/MPI was originally developed in Fortran 77, but is migrating to full use of the features of Fortran 90, especially automatic/allocatable arrays and array syntax. Communications on the T3D are handled by CRI's Shared Memory (SHMEM) data passing constructs, but message passing constructs based on MPI are currently being added for portability to other platforms. DANTSYS/MPI currently runs on Cray PVP's (serial only) and the CrayT3D (serial/parallel). With the addition of MPI, we expect to have a full parallel capability on all of these platforms plus the Intel TFLOPs machine, IBM SP2's, and clusters of SGI SMP's.

References

[1] R. E. Alcouffe et al., "DANTSYS: A Diffusion Accelerated Neutral Particle Transport Code System", Los Alamos National Laboratory Manual LA-12969-M (1995).

[2] K. R. Koch, R. S. Baker, and R. E. Alcouffe, "Solution of the First-Order Form of the Three-Dimensional Discrete Ordinates Equations on a Massively Parallel Machine", Trans. Am. Nucl. Sci., **65**, p. 198, Boston, MA (1992).

[3] R. S. Baker, C. Asano, and D. N. Shirley, "Implementation of the First-Order Form of the 3-D Discrete Ordinates Equations on a T3D", Trans. Am. Nucl. Soc., **73**, p. 170, San Francisco, CA (1995).