CONF-971005--9
LA-UR--97-955

PARALLEL 3-D S_N PERFORMANCE FOR DANTSYS/MPI ON THE CRAY T3D

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

Abstract

A data parallel version of the 3-D transport solver in DANTSYS has been in use on the SIMD CM-200's at LANL since 1994. This version typically obtains grind times of 150-200 nanoseconds on a 2048 PE CM-200. We have now implemented a new message passing parallel version of DANTSYS, referred to as DANTSYS/MPI, on the 512 PE Cray T3D at Los Alamos. By taking advantage of the SPMD architecture of the Cray T3D, as well as its low latency communications network, we have managed to achieve grind times of less than 10 nanoseconds on real problems. DANTSYS/MPI is fully accelerated using DSA on both the inner and outer iterations. This paper describes the implementation of DANTSYS/MPI on the Cray T3D, and presents two simple performance models for the transport sweep which accurately predict the grind time as a function of the number of PE's and problem size, or scalability.

I. 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 ordered sweep).[2,3] 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.[4] 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. This paper describes the implementation of DANTSYS/MPI on the Cray T3D and presents two simple performance models for the transport sweep which accurately predict 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.

In Ref. 3, we discuss our implementation of the ordered 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 solve 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%.[3] 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 "fake", 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.

II. IMPLEMENTATION OF THE ORDERED SWEEP ALGORITHM IN DANTSYS/MPI

Due to the SIMD nature of the CM-200, the ordered 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 ordered sweep by taking advantage of its SPMD architecture.[4] 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

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

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

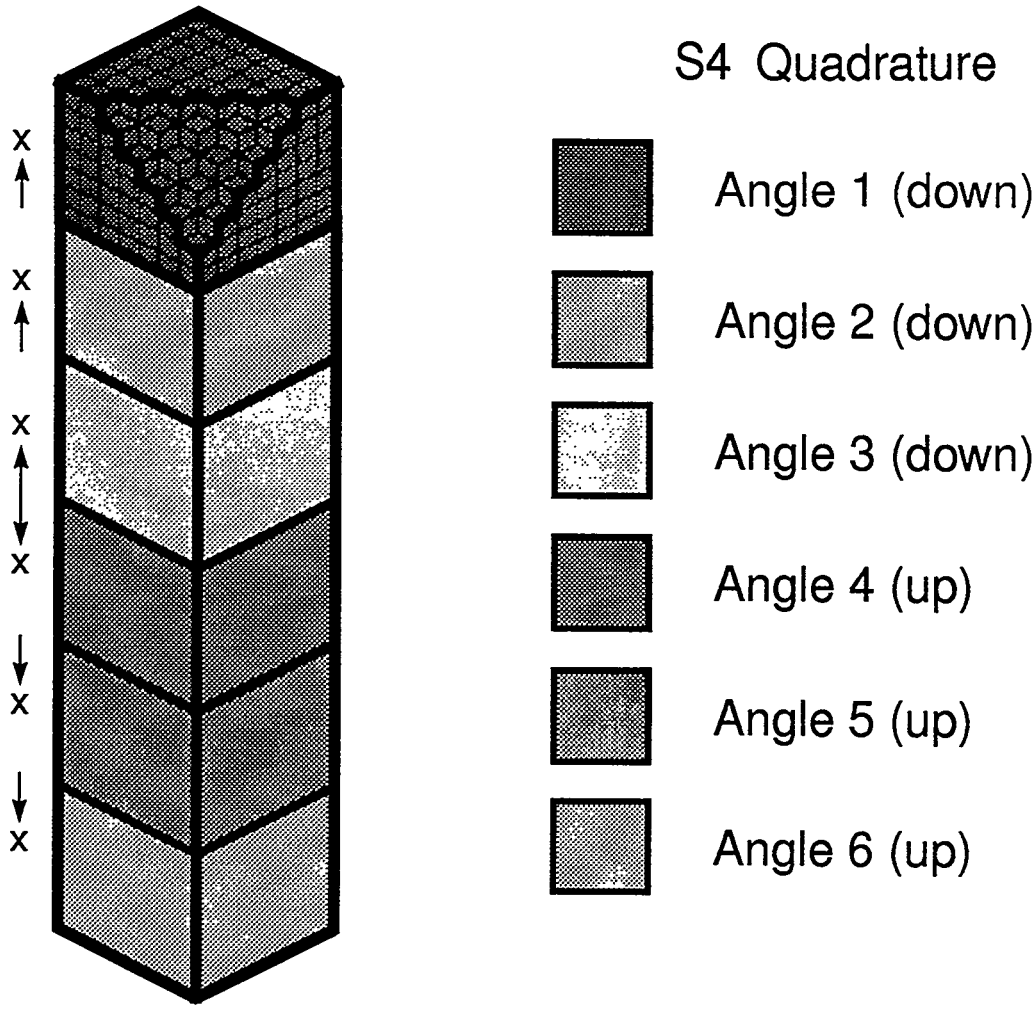

Figure 1. The Ordered Sweep

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 the loss in parallel efficiency due to load imbalancing. For a 128x128x128 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.

Since our ordered sweep algorithm is completely compatible with the standard (inner and outer iterations) method of solving the transport equation, implementation into an existing production code such as DANTSYS is straightforward. Thus, DANTSYS/MPI retains the traditional DANTSYS structure of 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[5] RTFLUX file upon completion, which may then be used by the edit module for post-processing. Parallel I/O on the T3D 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. In addition, for portability, we have also developed an alternate library (SPEIO) where a single PE performs all I/O operations and uses message passing to send/receive data to other PE's as required.

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 have also been added for portability to other platforms. Since DANTSYS/MPI is written in standard Fortran 90 and uses MPI, portability is not an issue. DANTSYS/MPI has been tested on Cray PVP's and Sun workstations (serial only) and the CrayT3D, IBM SP2, and the LANL cluster of SGI SMP's known as Blue Mountain (serial/parallel).

III. PERFORMANCE MODELS FOR THE ORDERED SWEEP ALGORITHM

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. One such commonly used model is Amdahl's Law, which can be expressed as

$$S_p = \frac{1}{(1 - f_p) + \frac{f_p}{p} + \frac{T_c}{T_s}} \tag{3}$$

where S_p is the parallel speedup, T_s is the solution time for one PE, T_c is the parallel communication time, f_p is the parallel fraction of the code, and p is the number of PE's. Note that Amdahl's "Law" is actually not a fundamental statement of parallel performance, but rather a simple performance model based upon a code structure consisting of only parallel and serial fractions plus, in this form, a constant communications time. While the assumption of a constant communications time is reasonable for the ordered sweep algorithm since the communications scale linearly with the number of PE's, this is not necessarily true for other parallel algorithms. However, the effects of the PCE are not included in the above model. Furthermore, Amdahl's Law requires T_s in order to estimate speedups, but it is generally not feasible to run the large calculations one does on hundreds of PE's on only a single PE due to time and memory constraints.

Therefore, we have constructed a more realistic performance model for the ordered sweep algorithm in DANT-

SYS/MPI. In this model, we model the total parallel solution time T_p as

$$T_p = (ITS \times I \times \{J/N_J\} \times \{K/N_K\} \times M \times 8 \times T_{src}) + 4 \times (N_J + N_K + [I/I_C] \times M \times 2 - 2) \times (I_C \times J_C \times K_C \times T_{swp} + I_C \times (J_C + K_C) \times (T_{bw} + T_{bnd}) + T_{lat} + T_{sync} + T_{dp}) \times ITS + T_{ser} \quad (4)$$

where $\{x\}$ is the floor function equal to the smallest integer less than or equal to x, T_{src} is the time per cell required to compute the source moments, angular source, and flux moments, T_{swp} is the time per cell spent in solving the difference equations, T_{bw} is the communications time (bandwidth) per cell face of a chunk, T_{bnd} is the time spent in calculating boundary data per cell face, T_{lat} is the communications latency, T_{sync} is the average time per each step of the ordered sweep spent in synchronization, T_{dp} is the serial overhead per each step of the ordered sweep, and T_{ser} is the serial overhead outside of the sweep. Note that T_{src} will be a function of the problem dependent S_N and P_N orders and T_{swp} will, in general, be a function of the selected spatial differencing method and the amount of (problem dependent) negative flux fixup, if any, for diamond differencing with set-to-zero fixup. T_{ser} will also vary from problem to problem, depending upon the number of inner and outer iterations, and other factors. However, T_{bw}, T_{bnd}, T_{lat}, T_{sync}, and T_{dp} are assumed to be problem independent. As defined here, the total solution time T_p does not include the time required for input or edit processing.

The values for T_{bw}, T_{bnd}, T_{lat}, T_{sync}, and T_{dp} were determined by instrumenting the computational kernel Sweep3D. Sweep3D is a small test code which includes the computational kernel of DANTSYS/MPI which performs the ordered sweep plus the inner iteration. Typically over 95% of the (transport) time in DANTSYS/MPI is spent in these two areas. A simple one-group, 64x64x64 spatial mesh, S_6P_1 problem was run over PE sizes ranging from 8 to 512 PE's on the Cray T3D. The values thus determined from Sweep3D were T_{bw} = 0.146 +/- 0.003 µsecs/cell face, T_{bnd} = 0.24 +/- 0.02 µsecs/cell face, T_{lat} = 20.0 +/- 0.4 µsecs, and T_{sync} = 14.8 +/- 1.6 µsecs. In addition, a similar problem, but with varying J and K mesh sizes so that the chunk size varied from 8 to 3136, was run on a single PE. From these runs, we obtained T_{dp} = 28.5 +/- 3.1 µsecs and, for diamond differencing with no fixup, T_{swp} = 0.802 +/- 0.003 µsecs. Note that T_{bnd} corresponds to a communications bandwidth of 33.3 MBytes/sec, while the theoretical maximum for a Cray T3D is 100 MBytes/sec when using two communication wires per PE.[6]

As an aside, we used the CRI performance analysis tool Apprentice to examine a S_6P_1, 50x50x50 spatial 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.

IV. PERFORMANCE BENCHMARKS

In order to assess the validity of our models, we examined a simple, one-group benchmark problem where we varied the mesh size, scattering order, and number of PE's. The basic problem consisted of a homogeneous cube with reflecting boundary conditions on the left, bottom, and front faces, and a uniform source. The dimensions of the cube were varied so that a constant mesh spacing of 0.1 cm was maintained. The cross sections were σ_t = 1.0 cm^{-1}, σ_s = 0.5 cm^{-1}, σ_1 = 0.2 cm^{-1}, and σ_2 = 0.05 cm^{-1}. A convergence criteria of 10^{-5} was used for all problems. The spatial differencing method was diamond differencing, where no fixup was required given the optically thin

cells. This problem was run using mesh sizes of 48x48x48, 64x64x64, 128x128x128, and 256x256x256, with a S_6 triangular quadrature set ($M = 6$) and a P_1 scattering order, and also on a 40x40x40 spatial mesh with a P_2 scattering order and an S_6 product quadrature set ($M = 9$).

In Tables 1 through 5 below we list, for each PE size, the total solution time, the total speedup factor S_p (measured from the first PE size with sufficient memory to run the given problem), the speedup ratio from the previous PE size [where 2.00 corresponds to an ideal linear speedup, and speedup ratios of greater than 2.00 (i.e., "super-linear speedup") are almost certainly due to cache effects], the grind time (calculated by dividing the total solution time by the number of inner iterations, spatial cells, and number of angles), as well as the grind times predicted by our models, and, finally, the PCE. In order to lend some significance to our grind times, we note that the DANTSYS grind time for the 64x64x64 mesh problem on a single processor of a Cray Y-MP is 316.6 nsecs, and the associated hardware performance is 131.2 MFLOPs. Table 6 lists the number of inner iterations required for each problem, as well as the resulting integral particle balance. In addition, Table 6 also contains the parameters used for the models, as described below.

Table 1: 40x40x40 Mesh, S_6P_2

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Amdahl's Law GT (nsecs)	Model GT (nsecs)	PCE
1	194.04	1.00		2807.3	2807.3	2802.4	1.0000
2	97.81	1.98	1.98	1415.1	1417.8	1425.1	0.9945
4	50.32	3.86	1.94	728.0	718.0	724.7	0.9890
8	26.92	7.21	1.87	389.5	369.4	374.5	0.9783
16	13.36	14.52	2.01	193.3	195.1	197.7	0.9677
32	7.90	24.56	1.69	114.3	107.9	109.5	0.9474
64	4.03	48.15	1.96	58.3	64.3	64.6	0.9278
128	2.94	66.00	1.37	42.5	42.5	41.0	0.7426
256	2.19	88.60	1.34	31.7	31.6	30.4	0.5952
512	1.87	103.76	1.17	27.1	26.2	27.1	0.4148

Table 2: 48x48x48 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Amdahl's Law GT (nsecs)	Model GT (nsecs)	PCE
1	157.94	1.00		1859.5	1859.5	1864.6	1.0000
2	82.49	1.91	1.91	971.2	938.1	950.2	0.9931
4	39.55	3.99	2.09	465.7	475.9	483.2	0.9863
8	20.77	7.60	1.90	244.5	244.8	249.9	0.9730
16	10.74	14.71	1.93	126.5	129.3	131.5	0.9600
32	5.99	26.37	1.79	70.5	71.5	72.5	0.9351
64	3.56	44.37	1.68	41.9	42.6	42.2	0.9114

Table 2: 48x48x48 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Amdahl's Law GT (nsecs)	Model GT (nsecs)	PCE
128	2.39	66.08	1.49	28.1	28.1	27.4	0.8675
256	1.69	93.46	1.41	19.9	20.9	19.7	0.8276
512	1.53	103.23	1.10	18.0	17.3	18.0	0.5684

Table 3: 64x64x64 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Model GT (nsecs)	PCE
1					1854.0	1.0000
2					939.0	0.9948
4	95.79	1.00		475.8	474.7	0.9897
8	48.02	1.99	1.99	238.5	242.7	0.9796
16	24.63	3.89	1.95	122.3	125.4	0.9697
32	13.40	7.15	1.84	66.6	66.8	0.9505
64	7.25	13.21	1.85	36.0	36.9	0.9320
128	4.44	21.57	1.63	22.1	22.1	0.8972
256	2.89	33.15	1.54	14.4	14.5	0.8649
512	2.20	43.54	1.31	10.9	10.9	0.8067

Table 4: 128x128x128 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Model GT (nsecs)	PCE
1					1918.1	1.0000
2					963.9	0.9974
4					483.9	0.9948
8					243.9	0.9897
16					123.2	0.9846
32	108.10	1.00		63.2	62.9	0.9746
64	54.05	2.00	2.00	31.6	32.5	0.9648
128	29.52	3.66	1.83	17.3	17.3	0.9458

Table 4: 128x128x128 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Model GT (nsecs)	PCE
256	16.64	6.50	1.77	9.72	9.48	0.9275
512	9.66	11.19	1.72	5.64	5.64	0.8930

Table 5: 256x256x256 Mesh, S_6P_1

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Model GT (nsecs)	PCE
1					1948.0	1.0000
2					976.1	0.9987
4					488.9	0.9974
8					245.2	0.9948
16					123.1	0.9922
32					62.1	0.9871
64					31.4	0.9821
128					16.0	0.9722
256	113.02	1.00		8.26	8.26	0.9624
512	60.05	1.88	1.88	4.39	4.39	0.9435

Table 6: Problem and Model Parameters

Problem Size	No. Inners	Particle Bal.	$f_p>$	T_c (secs)	f_p	T_{src} (μsecs)	T_{ser} (secs)
40^3	15	3.05-6	0.992	0.2799	0.99406	1.981 (0.009)	0.4780
48^3	16	2.22-6	0.992	0.2624	0.99429	1.050 (0.011)	0.1989
64^3	16	3.97-6	0.996			1.043 (0.008)	0.2880
128^3	17	4.06-6	0.9991			1.112 (0.022)	0.5406
256^3	17	5.66-6	0.9997			1.144 (-)	1.227

From Eq. (3), we can estimate the minimum parallel fraction of the code f_p by looking at the ratio of S_p at any two values of p, which we denote p_1 and p_2, and assuming that T_C is zero, or

$$f_p > \frac{S_2/S_1 - 1}{S_2/S_1 \times (1 - 1/p_2) - (1 - 1/p_1)} \tag{5}$$

The numbers in column 3 of Table 6 are calculated by choosing 512 PE's for p_2, and p_1 at the smallest number of PE's the problem will fit on. Obviously, DANTSYS/MPI is a highly parallel code and, as the granularity increases with increasing mesh size, so does the parallel fraction of the code and the maximum possible speedup.

The communications time T_C is measured by timing the routines that transfer the angular fluxes to the adjoining PE's for the 128 PE test cases. Given T_C, we can exactly determine f_p from Eq. (3) for those problems which fit on a single PE (which provides T_s). With f_p, we can then use Amdahl's Law to predict the grind times for all PE's using the results from only two points (1 and 128 PE's). Despite the fact that Amdahl's Law does not include the effects of PCE, we see that it still does an excellent job of modeling the grind time. As expected, the exact f_p's calculated from Eq. (3) are greater than those predicted by Eq. (5).

None of the benchmark problems in Tables 1 through 5 required fixup. Thus, we can assume that the value of T_{swp} determined from Sweep3D is still valid, as are T_{bw}, T_{bnd}, T_{lat}, T_{sync}, and T_{dp}. If we rearrange Eq. (4), we see that it predicts a linear relationship between the source computation time and the number of cells solved per PE, with slope T_{src} and Y-intercept T_{ser}, given that all the other parameters are known. Therefore, by performing a linear least squares fit to the data, we determine the T_{src}'s listed in Table 6 for each problem, along with the standard deviation listed below in parentheses. In theory, the least squares fit also gives us T_{ser}, but in practice the resulting error in this quantity is too large for us to use it. Instead, once we have T_{src}, we estimate T_{ser} by subtracting off all the other remaining components for the 512 PE data from the total solution time T_p and assume that the remaining time is T_{ser}. The least squares fit for the 40x40x40 spatial mesh problem is shown below at Figure 1, where the circles connected by dashes correspond to the data and the solid line is the least squares fit.

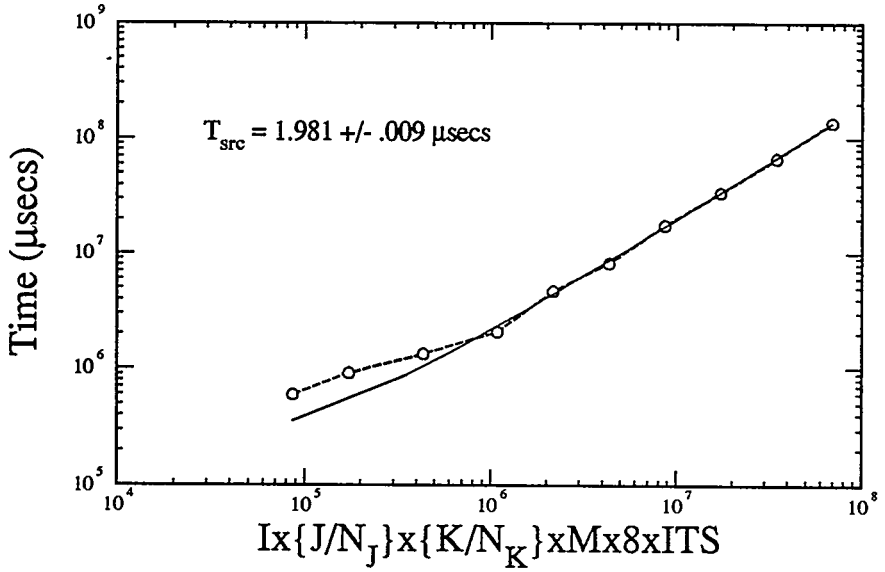

Figure 1. Least Squares Fit for the 40x40x40 Spatial Mesh.

The parameters for T_{src} and T_{ser} listed in Table 6, along with those for T_{bw}, T_{bnd}, T_{lat}, T_{sync}, and T_{dp} from Sweep3D described above, are used to calculate the model grind times in tables 1 through 5. We see excellent agreement with both those predicted by Amdahl's Law (where available) and the actual grind times. Unlike Amdahl's Law, our model does not require that a calculation fit on a single PE in order to determine the model parameters, but instead needs only two points at any PE size.Thus, we can use our model to predict total speedups for applications too large to run on a single PE. Figure 2 below shows the actual grind times for the 40x40x40 calculation for 1to 512 PE's, the grind times predicted by our model, and the ideal grind time, assuming a linear speedup from 1 to 512 PE's.

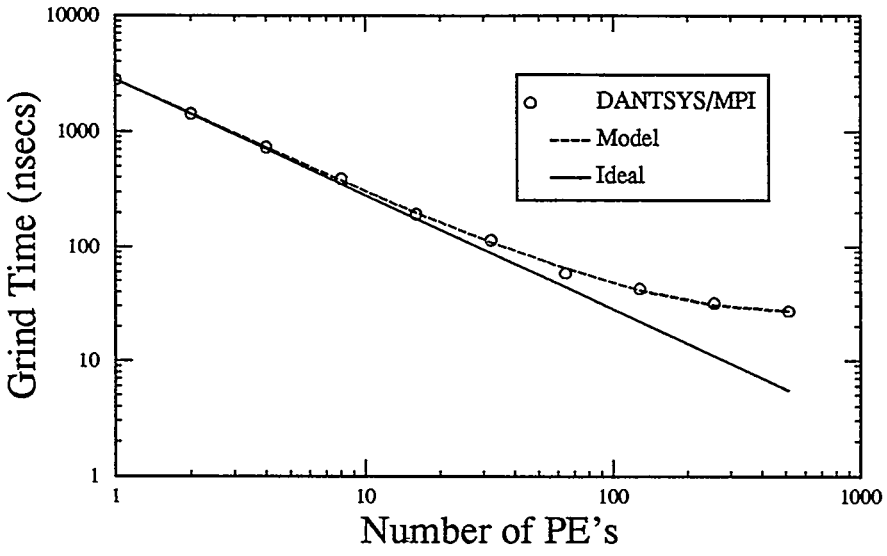

Figure 2. Grind Times for the 40x40x40 Spatial Mesh

In comparing the single PE grind times (both actual and predicted) for the S_6P_1 problems (Tables 2-5), we see that they fall in the narrow range of 1,859 to 1,948 nsecs, regardless of the mesh size. Since these calculations are essentially identical except for the spatial mesh size, we would expect the grind times to also be essentially identical, although the total solution times increase with increasing mesh size. This points out the importance of presenting both total solution time and grind times. Furthermore, it provides an additional validation of our model, since the grind time predicted by our model for the 256x256x256 mesh on a single PE, and the data in Table 6 used by our model (T_{src}) in making that prediction, are in good agreement with similar calculations actually performed on a single PE (Table 2). Thus, the total speedup of ~ 440 predicted for the 256x256x256 mesh problem on 512 PE's is reasonable. Of course, since the smaller meshes have a smaller granularity, the agreement in grind times does not hold true as the number of PE's is increased, as can also be seen from the tables.

It is also interesting to compare the values of T_{src} in Table 6 between the P_1 and P_2 (40x40x40 mesh) problems. Since, in 3-D, P_1 calculations have four moments and P_2 calculations nine, we would expect, for simple one-group problems, that T_{src} would be approximately twice as great for the 40x40x40 mesh as for the other meshes. This is indeed the case. This increase in T_{src} for the P_2 problem is entirely responsible for the increase in the single PE grind time. Finally, we should also mention that, as the mesh size increases in Table 6, there is very little increase in the number of iterations required to solve the problem since, for the ordered sweep algorithm, we retain the ability to

invert the source iteration equation in a single sweep.

V. APPLICATION TO ASSEMBLY 6

Performance results for DANTSYS/MPI on a more realistic problem are presented below in Tables 7 and 8. Assembly 6 is a critical assembly consisting of a highly enriched uranium spherical core surrounded by a beryllium-oxide reflector.[3] 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 94x94x94 spatial mesh. Because of mirror symmetry, only one-eighth of the assembly was actually modeled by the mesh. The ordered sweep algorithm was implemented in conjunction with diamond differencing in space. No fixup was needed in this problem due to the optically thin meshes. This assembly is experimentally known to have an eigenvalue of unity, while the value of the k eigenvalue calculated by DANTSYS/MPI was 0.9969 with a convergence criteria of 10^{-4}.

Table 7: Assembly 6 Performance Results

p	Soln. Time (secs)	S_p	Speedup Ratio	Grind Time (nsecs)	Model GT (nsecs)	PCE
1					3411.8	1.0000
2					1721.7	0.9769
4					864.1	0.9747
8					430.6	0.9500
16					215.6	0.9259
32	3,766.5	1.00		107.5	107.0	0.9176
64	1,876.4	2.01	2.01	53.5	54.0	0.9093
128	1,002.1	3.76	1.87	28.6	27.6	0.8933
256	541.7	6.95	1.85	15.5	15.1	0.8778
512	311.1	12.11	1.74	8.88	8.88	0.8484

Table 8: Assembly 6 Problem and Model Parameters

Problem	No. Outers	No. Inners	Particle Bal.	$f_p>$	T_{src} (μsecs)	T_{ser} (secs)
94^3	16	586	3.65-5	0.9993	2.603 (.030)	56.763

The least squares fit for Assembly 6 is shown below in Fig. 3. The value for T_{src} of 2.603 μsecs is considerably larger than that of 1.981 μsecs for the one-group S_6P_2 case, but Assembly 6 is a 12-group problem including fission. Since we do not perform any decomposition over energy, this additional work is entirely local on each PE and requires no additional communications. A smaller version of Assembly 6 with a 14x14x14 spatial mesh was run on a single PE, where the actual grind time was 3,741.0 μsecs. Given the effects of problem size on performance due to

cache and pipelines, this value is in reasonable agreement with our model prediction of 3,411.8 μsecs.

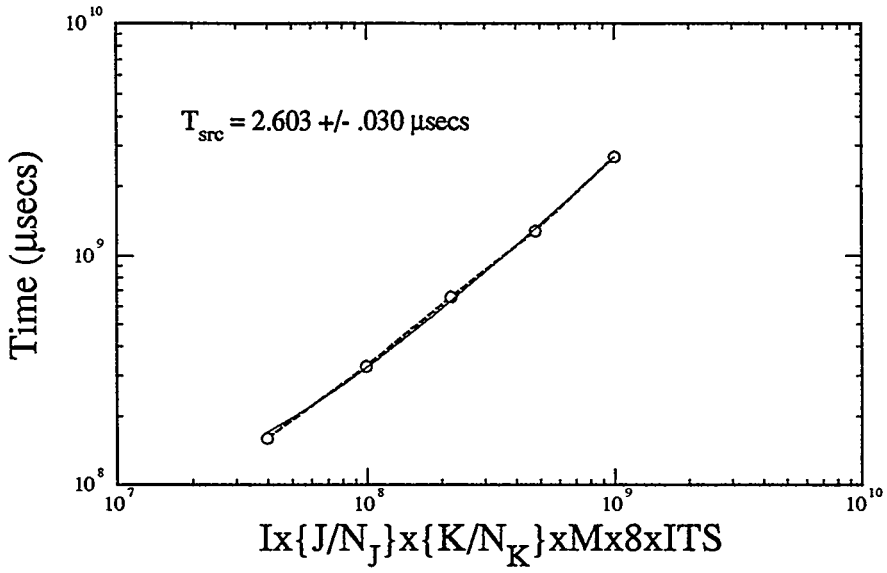

Figure 3. Assembly 6 Least Squares Fit.

Figure 4 shows the actual grind times for the Assembly 6 calculation for 32 to 512 PE's, the grind times predicted by our model, 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, and the estimated total speedup obtained for this realistic application is ~380 on 512 PE's. More importantly, 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 45 Y-MP PE's.

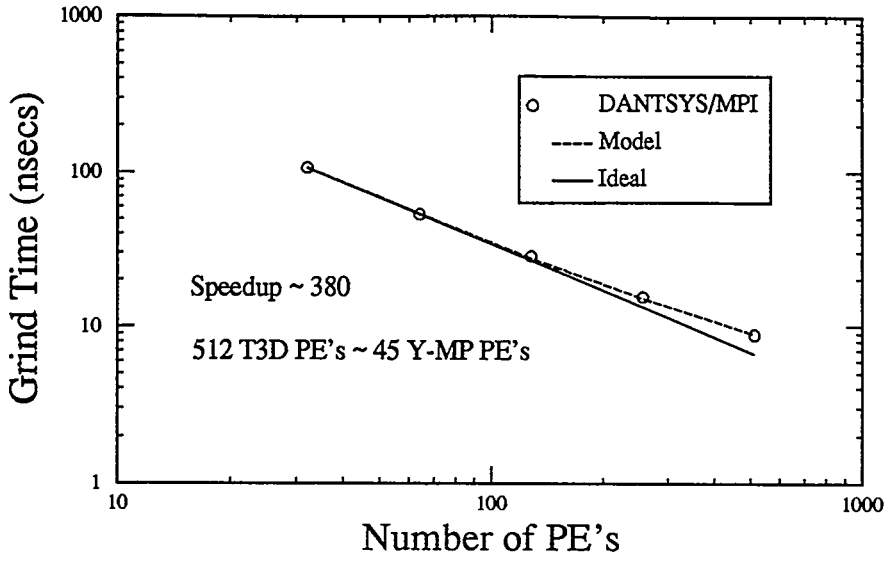

Figure 4. Assembly 6 Grind Times

VI. 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 preconditioned 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.

To examine the performance of the diffusion solver on the T3D, we have chosen the small LWR problem of Takeda[7] except we have varied the spatial mesh from 50x50x50 to 200x200x200 in each of the coordinate dimensions. This is a two energy group, S_8P_0 k_{eff} calculation and is being converged to 5.0e-5 in the eigenvalue and the pointwise scalar fluxes. To first address the effectiveness of our diffusion solution algorithm on parallel architectures, we have solved this problem as a pure diffusion problem using DANTSYS/MPI. For each of the spatial mesh

variations, we have done the diffusion calculation using a number of processors from 16 to 512. In Table 9 we present the results in terms of total CPU time to solve the problem. We also include for reference results from a single processor Y-MP run using both the same diffusion solver as that used on the T3D (*I*-line preconditioned CG) and also using a multigrid solver. The dashes in the table mean that the calculation would not fit in the available memory. These results show that we are able to solve large diffusion problems on the T3D and that our algorithm does scale reasonable well with a speedup ratio of about 1.80, though it does tail off as the number of processors increase for a constant problem size. We also note in passing that although for the T3D we can run up to 42 times faster than the Y-MP, the impact of the effectiveness of the method used to invert the diffusion operator is such that much of the gain is lost. Thus in the best of all worlds, we would very much like a parallel multigrid solver. The table results are presented in graphical form in Figure 5.

Table 9: CPU Time for a Diffusion Calculation as a Function of the No. of PE's

mesh size	Number of PE's on the T3D						Y-MP	
	16	32	64	128	256	512	CG	MG
50x50x50	22.0	12.6	7.3	4.9	3.2	-	49.0	8.3
100x100x100	257.2	143.9	75.9	44.4	26.4	16.5	693.8	52.8
200x200x200	-	-	584.2	324.4	184.3	97.0	-	-

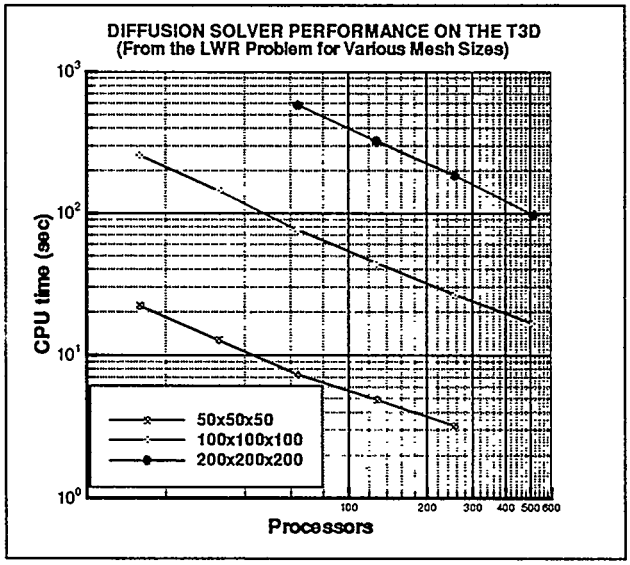

Figure 5. CPU Times as a Function of the Number of PE's for the LWR Problem.

To follow up the impact of the parallelization of the inversion of the diffusion operator upon a total diffusion accelerated transport problem. we have taken the 100x100x100 mesh version of the small LWR problem and run it on the T3D with a varying number of processors. We also include for reference the performance of this problem on the Y-MP. In Table 10 we present the results as CPU time in seconds broken down in the diffusion solve time, the transport solve time and the total time to do the calculation. From these results we can discern that the DSA time is a factor of two greater than the transport time and that they scale very well together. We note in passing that multigrid is about a factor of 5 times more efficient than the CG although its use requires more transport iterations and

thus more transport time. In Figure 6 we present these results in graphical form.

Table 10: CPU Time for the Components of a DSA Solution of LWR on the T3D

	Number Of PE's On The T3D						Y-MP	
	16	32	64	128	256	512	CG	MG
DSA time	348.2	195.4	102.7	60.1	36.0	22.3	928.2	198.2
Trans time	181.3	96.5	52.5	30.6	18.1	11.8	299.1	455.2
Total time	536.4	295.6	157.1	91.8	54.8	34.6	1237.5	666.0

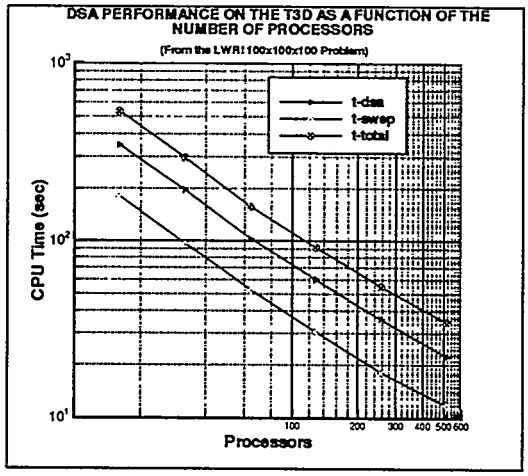

Figure 6. CPU Times for the Various DSA Solution Components From the LWR Problem.

VII. CONCLUSIONS

DANTSYS/MPI is a highly parallel code which performs effectively on realistic applications. As implemented in DANTSYS/MPI on the Cray T3D, the ordered sweep algorithm allows us to solve not only larger problems than possible on traditional super-computers, but also to solve them more quickly. Since the 2-D spatial decomposition of our ordered sweep algorithm inverts the source iteration exactly in a single sweep, its performance is insensitive to mesh size The ordered sweep algorithm allows to retain both the use of the standard outer and inner iteration strategy of production codes and also the use of standard accelerations schemes such as DSA. Since the ordered sweep algorithm provides only 2-D parallelism, it has a lower compute-to-communications ratio than other decomposition strategies. However, as demonstrated here, the ordered sweep algorithm can be expected to perform well on any machine architecture with a low-latency communications network. We have provided a performance model which accurately predicts the performance of DANTSYS/MPI on the Cray T3D, and can easily be used to predict the performance on other machine architectures as well. Because DANTSYS/MPI is written in standard Fortran 90 using MPI, it can easily be ported to other architectures. We will present results for other architectures in a future paper.

VIII. 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, J. M. McGhee, K. R. Koch, and J. E. Morel, "Two S_N Algorithms for the Massively Parallel CM-200 Computer", submitted to Nucl. Sci. Eng. for publication.

4. 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).

5. R. D. O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes, Version IV", Los Alamos Scientific Laboratory report LA-6941-MS (1977).

6. Cray Research, Inc., "Cray T3D Applications Programming", TR-T3DAPPL, Revision D, Release 1.0 (1994).

7. T. Takeda and H. Ikeda., "3-D Neutron Transport Benchmarks", OECD/NEA Committee on Reactor Physics Report, NEACRP-L-330 (1991).