DEVELOPMENT AND PARALLELIZATION OF THE THREE-DIMENSIONAL CHARACTERISTICS SOLVER MCI OF DRAGON

M. Dahmani, G. J. Wu*, R. Roy and J. Koclas
Nuclear Engineering Institute
École Polytechnique de Montréal
P.O.Box 6079, Station Centre-Ville
Montreal, H3C 3A7 CANADA
{Mohamed.Dahmani , Robert.Roy, Jean.Koclas} @polymtl.ca

ABSTRACT

In this paper, recent advances in parallel software development for solving neutron transport problems are presented. The method of characteristics is based on the resolution of the differential transport equation following the tracking lines in order to collect the local angular flux components. Due to the excessive number of tracks in the demanding context of 3D large-scale calculations, reliable acceleration techniques are developed in order to obtain a faster iterative solver, especially for problems with high-scattering ratio. The load balancing strategies include a global round-robin distribution of tracks, an approach where we forecast the calculation load implied by each track length and a macro-band decomposition where tracks crossing the same regions are grouped together. The performance of the PVM and MPI implementations in the characteristics solver is analyzed for realistic applications.

1. INTRODUCTION

DRAGON[1] is a lattice cell code which uses the Collision Probability (CP) method for solving the neutron transport equation in arbitrary geometrical domain. A new solver (called MCI) based on the method of characteristics (MOC) was developed as an alternative technique for 3D domains. For large problems, the MOC technique is very promising because it does not generate huge CP matrices, and still provides solutions as accurate as the CP method for general 3D geometries. However, the MCI iterative solver can converge slowly, especially in the presence of high-scattering media. In order to minimize the number of iterations, a Self-Collision Rebalancing (SCR) method was designed as an acceleration method for the MCI sequential solver[2]; this method uses first-collision probability to recover from the local self-scattering effect. Another new technique, the Track Merging Technique (TMT), was also designed to reduce the number of tracking lines. Using this technique, two tracking lines crossing the same regions in the same order are merged together. It can be easily implemented inside the ray tracing routine that computes the tracks. The TMT is generally very efficient because more than half of tracking lines can be merged together without loosing any accuracy.

* G.J.Wu can now be reached at E-mail address: GuangJunWu@netscape.net

Based on these ideas, a sequential module MCI was developed in DRAGON to treat supercell 3D problems with isotropic boundary conditions. This technology was found accurate with respect to several standard transport problems specific to CANDU reactors, where reactivity devices are perpendicular to the fuel channels [2]. Using this sequential module, most of regular supercells have been solved in approximately the same number of outer iterations as in the CP method, even with the high-scattering ratio due to the heavy water moderator.

Nevertheless, it is still far from possible to solve the transport equation over the complex geometry of a whole power reactor core without various approximations. However, the exponential growth in capacity and power of computer resources enables the nuclear engineering community to improve their computational models and to explore the limits of older models. Nowadays, the three-dimensional effects for various critical configurations can be simulated using new parallel algorithms. Here, we will present one of the most flexible and well-organized algorithms known to perform such 3D state-of-the-art computations in a distributed environment.

The simplified static transport equation for obtaining a flux ϕ from a source Q is given by:

$$\hat{\Omega}.\vec{\nabla}\Phi(\vec{r},\hat{\Omega},E)+\Sigma_t(\vec{r},E)\Phi(\vec{r},\hat{\Omega},E)=Q(\vec{r},\hat{\Omega},E)\,, \tag{1}$$

where $\vec{r}$ is a spatial point in the domain D, $\hat{\Omega}$ is a solid angle and E is the energy. The parallelization based on CP method obtained after distributing the energy variable in multi-group solver, as well as the usual spatial domain decomposition methods, have already been developed a few years ago [3], and the spectrum of applications for similar techniques has been extended to other related fields such as radiography [4]. The aim of this paper is to explain the newest approaches based on the characteristics formulation of the transport equation (1) and to introduce the new parallel computational techniques implemented to extend the MCI sequential solver.

Recently, domain decomposition techniques have been applied to large-scale parallel transport calculations using the method of characteristics. The spatial decomposition of multi-assembly problems where neutron paths are directly linked together when crossing assemblies exhibit limited speed up (5.3 on 8 processors) on shared memory Sun Entreprise4000 because of tight coupling between the assemblies [5]. The angular decomposition technique, where directions are distributed among a set of processors, has also been tested [6]. Speed up obtained using this angular decomposition is of the order of 3.7 for 4 processors, and 6.8 for 8 processors. These techniques were restricted to two-dimensional problems, but they give the trend that we are looking for when we apply our characteristics method to three-dimensional geometries.

The paper is organized as follows. In section 2, we present a brief review of MOC principles, and the particular 3D implementation of these principles in our solver. Our iterative MOC scheme is also described with the related acceleration techniques that can be applied in the 3D context. In section 3, we present different options for the implementation of a general MCI parallel solver. Section 4 is composed of numerical tests, most of these appearing in CANDU supercells, and the parallel performance for various parallel options is studied. In the last section, we conclude by a discussion on prospective future work in order to obtain a large-scale distributed transport solver in the coming years.

2. THE 3D CHARACTERISTICS SOLVER MCI

2.1 MOC PRINCIPLES AND SPECIFIC 3D IMPLEMENTATION

The main idea of the method is to solve the differential form of the Boltzmann equation following the tracking lines (also called "characteristics"). Assuming a finite domain V split into homogeneous regions, each having a volume V_j, the average (one-group) flux Φ_j is given by:

$$V_j \Phi_j = \int_{V_j} d^3 r \int_{4\pi} d^2\Omega \Phi(\vec{r},\hat{\Omega}) = \int_{\Gamma} d^4 T \int_{-\infty}^{+\infty} dt\, \chi_{V_j}(\vec{T},t)\Phi(\vec{p}+t\hat{\Omega},\hat{\Omega}) \qquad (2)$$

A characteristics line $\vec{T}$ is determined by its orientation $\hat{\Omega}$ along with a reference starting point $\vec{p}$ for the line. The variable t refers to the local coordinates on the tracking line and the function $\chi_{V_j}(\vec{T},t)$ is defined as 1 if the point $\vec{p}+t\hat{\Omega}$ on the line $\vec{T}$ in the region V_j, and 0 otherwise. Assume that a line $\vec{T}$ crosses K regions before reaching the external boundary, and define the crossing points $r_k = \vec{p} + t_k\hat{\Omega}$ ordered in the neutron traveling direction. The d^4T element is then decomposed into a solid angle element $d^2\Omega$ a corresponding plane element d^2p. The Γ domain is covered by a quadrature set of solid angles and by scanning the plane $\pi_{\hat{\Omega}}$ perpendicular to the selected direction $\hat{\Omega}$ for the starting point $\vec{p}$. The differential transport equation for angular flux that has to be solved on each segment k is now:

$$\left(\frac{d}{ds} + \Sigma_{N_k}\right)\Phi(\vec{r}_{k-1} + s\hat{\Omega},\hat{\Omega}) = \frac{Q_{N_k}}{4\pi};\ \mathrm{s} \in [0, \mathrm{L_k}] \qquad (3)$$

For the chosen line $T(\hat{\Omega},\vec{p})$, the collection of segment lengths L_k and region numbers N_k for each region encountered along the line must be calculated (and can eventually be recorded in sequential binary files). Assuming isotropic input currents at the external boundary and isotropic sources, a recursive solution can be found from the following simple attenuation equation:

$$\phi_k = \phi_{k-1} e^{-\tau_k} + \frac{q_k}{\sigma_k}(1 - e^{-\tau_k}), \qquad (4)$$

where local sources are given by $q_k = Q_{N_k}/4\pi$, the local total cross sections are $\sigma_k = \Sigma_{N_k}$ and total optical paths when crossing the region N_k are $\tau_k = \Sigma_{N_k} \times L_k$. Reciprocity relations allow us to solve concurrently for direction $\hat{\Omega}$ and for its reverse direction $-\hat{\Omega}$ using the same line; however, the input currents are not the same at both ends. The scalar flux in region *j* can be recovered by a reductive operation over all lines:

$$\Phi_j = -\frac{1}{4\pi\Sigma_j V_j}\sum_T \omega_T \sum_k \delta_{jN_k}\Delta\phi_k + \frac{Q_j}{\Sigma_j}, \qquad (5)$$

where δ is the Kronecker symbol and $\Delta\phi_k = \phi_k - \phi_{k-1}$ is the local flux difference.

Equations (2-5) are the basis of all characteristics solvers that will be used here under the following multi-group iterative scheme:

1. Guess new fission sources and incoming currents (outer loop).
2. Guess scattering sources (inner loop).
3. Compute the flux map for each energy group:
   (a) compute local solutions for each characteristics line;
   (b) apply local acceleration techniques;
   (c) perform a reductive sum of all contributions to the flux moments;
   (d) apply global inner acceleration techniques.
4. If flux map are not converged, go to 2.
5. Apply global outer acceleration techniques.
6. Compute critical factors for next neutron generation.
7. If not converged, go to 1.

This iterative scheme exhibits good performance only if consistent acceleration techniques are used. We will now present the current state of work on these acceleration techniques pertinent to the MCI module in DRAGON.

2.2 SELF-COLLISION REBALANCING TECHNIQUE

As we can see from the iterative scheme described above, we apply at Step 3.(b) any local acceleration techniques to reduce the number of inner iterations. Several acceleration techniques are developed and used for MOC [7]. In our characteristics solver, we use the Self-Collision Rebalancing (SCR) technique [8]. This technique is based on the equivalence between the collision probabilities method and the method of characteristics. The SCR uses the self-collision probabilities (first-flight collision probabilities from one region to itself) in order to rebalance the energy distribution of the scalar flux for each region separately.

2.3 TRACK MERGING TECHNIQUE

In the process of generating tracking lines for a large domain, two successive lines can cross exactly the same regions with the same direction. Segment lengths can be slightly different: for a reference length L, one track may have length $L+\varepsilon$ while another has $L-\varepsilon$. It was shown that the two tracks could be merged together with $O(\varepsilon^2)$-order of error on the local angular flux. The resulting line takes the averaged segment lengths and additional weights of the original lines as its properties [8].

Another step in attempting to group together characteristics lines having the same behavior is the macro-band grouping [7]. In this concept, we identify tracks by the region numbers that they are crossing. This classification encompasses two different attributes: the spatial regions that are crossed plus the set of directions that allow this crossing sequence. Once the finite set of these attributes has been identified, it is possible to define an almost perfectly vectorized process for each macro-band. This approach could be useful in the context of SMP processors, providing another (fine-grain) level of parallelism.

3. PARALLELIZATION IN THE MCI SOLVER

For cases with a large number of regions, even when using both TMT and SCR techniques, the iterative solver based on characteristics method can become very expensive. In order to solve such a large-scale problems with minimum amount of time, we use a parallel approach in the MCI solver.

Parallelization of transport solvers is generally based on domain decomposition; however, this can include spatial, angular or energy decomposition. In characteristics methods context, the basic computation unit is focused on repeating the same sequence of calculations on each tracking line. Obviously, this is the finest granularity level of operations that can be performed without any communication. The idea is to distribute the tracking lines on several different processors. The number of lines is much larger than the number of processors available. Therefore, the lines are grouped and each processor takes control of a group. Several strategies are used to group the lines in order to load balance the parallel calculation. In the following, we assume that N_T represents the number of tracks and that N_P is the number of processors.

3.1 STATISTICAL LOAD BALANCING

These options are statistical in the way that they rely on the uniform distribution of number segments when grouping tracks. Three of these options have been developed:

- **ANGL**: all tracks of the same direction are grouped together; the number of directions must be equal to the number of processors;
- **SPLT**: subsets of N_T / N_P tracks respect the order in which these would be sequentially generated;
- **STRD**: each tracks is given in a round-robin fashion to each processor, so that track i is on processor $i \bmod p$.

Different tests were performed with these options, and it was shown that the **STRD** option is the most efficient in statistically preserving the load balance [9].

3.2 DETERMINISTIC LOAD BALANCING

Only one option was encoded using a deterministic approach based on the calculation load of each single track:

- **MCRB**: the number of segments of each track is taken into account as a weighting factor for distributing tracks; each processor receives an equivalent total weight based on the calculation load.

This option represents the most uniform load balancing. The tests done using this option show speedups nearly linear and similar to the statistical **STRD** option [9]. The **MCRB** option is still under investigation and, in the context of distributed grid computing, it could help to synchronize the reduction operations.

3.3 PERFORMANCE OF MESSAGE-PASSING ALGORITHMS

After using one of the options described above, the total number of tracks N_T is distributed on the processors. At each iteration step, a multicasting communication process is used to recover the partial contribution to the angular fluxes; each processor has its own copy of the flux and the source arrays with the same consistent ordering for unknowns (regions and energies). The scalar flux is then reconstructed on every single processor before the iteration procedure starts (multicasting communication process). The original MCI parallel solver was designed and coded in such way it runs on an arbitrary number of processors using the parallel virtual machine (PVM) environment [10]. Recently, an MPI version of the MCI parallel solver was also implemented to improve the message passing performance.

3.3.1 PVM IMPLEMENTATION

The angular flux is scattered by one given processor to all others processors belonging to the virtual machine. A process begins to construct their subpart of the flux and then scatters its results to all other processes using a **pvmfmcast** subroutine call. At this level, all processes are synchronized. After that, every process will receive (**pvmfrecv**) the results from the others and accumulate their contributions (loop on other processors). The **pvmfrecv** subroutine is called $N_P - 1$ times for each process, this non-collective operation can increase the communication time and can therefore downgrade the parallel performance.

3.3.2 MPI IMPLEMENTATION

Recently, the MPI version was implemented in the MCI solver to take advantage of the growth of the MPI library regarding to the collective communication subroutines and of its extended portability [11]. The angular flux is then scattered to all processors by using the **Mpi_Bcast** subroutine that performs the send and receive in one synchronized step. When constructing the flux, we have used the **Mpi_Allreduce** to perform the collective communication between the process, making the reduction sum on all contributions at the same time. The reduction sum is then stored in the buffer and a copy is scattered to all processes (see Fig. 1). In the following, we will compare both implementations.

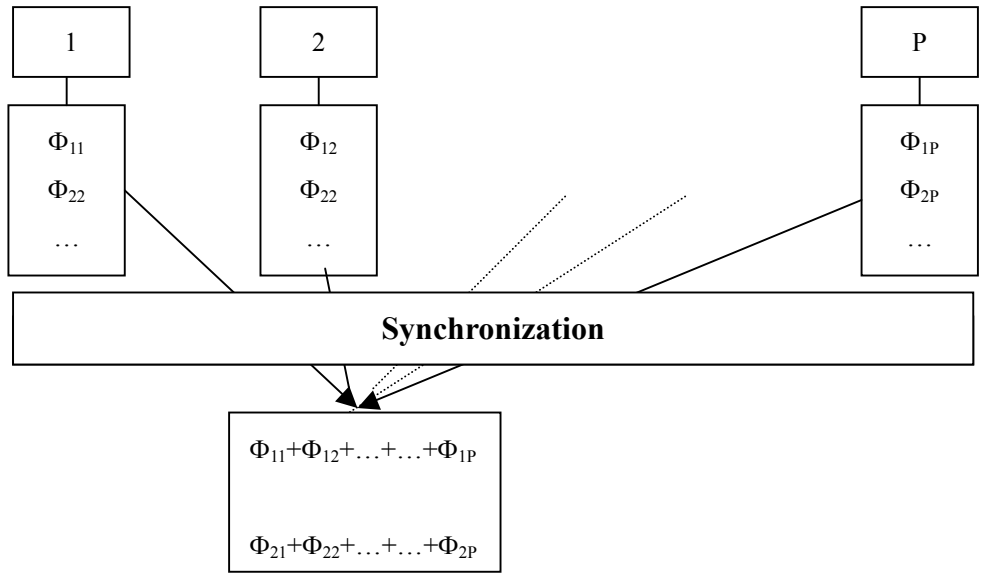

Figure 1. Collective communication by reduction sum in MPI implementation.

4. NUMERICAL RESULTS

Tests were done for CANDU6 absorber rod supercell calculations in 3D geometry [2]. Two horizontal fuel bundles and one vertical rod compose the domain. Tests were performed on two Beowulf clusters located at the Center for Research in Computation and its Applications (CERCA) [12]:

- **Beowulf 1** is a cluster of 16 nodes equipped with AMD Athlon 1.4 GHz processor and 1 GB memory connected by a Fast Ethernet switched network;
- **Beowulf 2** is a small SMP composed of 8 QuadXeon multiprocessors, each of these 8 nodes has 4 processors sharing a 4GB memory. Nodes are connected with a 1 Gb/s Myrinet switch.

All the tests were done using the STRD option with SCR acceleration and with three different levels of track merging:

- **No TMT**: no track merging at all;
- **TMT-1**: one line can be merged to the next one as the ray tracing is performed, thus this is equivalent to merging lines on a unique dimension;
- **TMT-2**: more than the previous option, lines are sorted and merging is done on both dimensions of the perpendicular plane.

In Table I, we vary the density of tracks (DT) (tracks/cm^2) and show the CPU time spent by the PVM implementation of the MCI solver for different number of processors on Beowulf 1. The CPU time decreases with the number of processors, but this decreases is little lower for the TMT-1 option and even lower for TMT-2. This is due to the insufficient number of lines of TMT-2, so the communication time increases faster than computation time as we can see in Figure 2 which represents the speedup variation with the number of processors. In Table II, we show the speedup variation of the PVM implementation on both clusters. The speedup on the SMP cluster is almost linear and the closest to the ideal speedup because the communications are faster. In Figure 3, we compare the respective speedup curves of the MPI and PVM implementations with and without merging lines. The MPI speedup curve is more linear than the PVM one for both cases; this is due to the increased performance of the message-passing MPI collective subroutines. With the PVM implementation, we begin to lose performance after 8 processors when the communication time becomes important; this shows that the MPI implementation is more scalable.

CONCLUSIONS

We have presented here the recent developments done in the characteristics solver MCI. To obtain a fast iterative solver, self-collision rebalancing and track merging techniques were used. Results have shown that this characteristics solver can offer fast and accurate solutions when implemented with efficient parallel computation algorithms. The decrease in the computational time strongly depends on the number of tracks in the domain. When this number is significantly larger than the number of regions, the computing time is dominant and the parallelism is efficient.

Next step would be to introduce modular ray tracking schemes based on pattern recognition for the macro-bands. Calculation along characteristics is so simple that mobile agents could be defined to compute on very limited view of the whole-core system, allowing these solvers to evolve on grid computers. In order to further improve the parallel performance and to allow for the solver to become more portable to the SMP architectures clusters, I/O operations could also be performed on the tracking file using MPI-2.

ACKNOWLEDGEMENTS

This work has been carried out partly with the help of a grant from the Natural Science and Engineering Research Council of Canada. M. Dahmani would like to thank the Beowulf administrators (S. Fourmanoit, P. Hamelin and M. Béland) at CERCA for their help.

REFERENCES

1. G. Marleau, A. Hebert and R. Roy, "A User's Guide for DRAGON", Technical Report **IGE-174 Rev. 3**., École Polytechnique de Montréal (1997).
   A copy of this report is available at **http://www2.polymtl.ca/nucl**
2. G. J. Wu and R. Roy, "A New Characteristics Algorithm for 3D Transport Calculation", to appear in *Annals of Nuclear Energy* (2002).
3. A. Qaddouri, R. Roy, M. Mayrand, B. Goulard, "Collision Probability Calculation and Multigroup Flux Solvers Using PVM", *Nucl. Sci. Eng*., **123**, 392 (1996).
4. F. Inanc, B. Vasiliu, D. Turner, "Parallel Implementation of the Integral Transport Equation-Based Radiography Simulation Code", *Nucl. Sci. Eng.,* **137**, 173 (2001).
5. S. Kosaka and E. Saji, "The Characteristics Transport Calculation for a Multi-Assembly System using Neutron Path linking Technique", *Proc. Mathematics and computation ANS M&C 99*, September (1999).
6. G. S. Lee, N. Z. Cho, S. G. Hong, "Acceleration and Parallelization of the Method of Characteristics for Lattice and Whole-Core Heterogeneous Calculation", *Proc. of PHYSOR 2000*, May (2000).
7. G. J. Wu, "Développement d'une méthode des caractéristiques tridimensionnelle et application aux calculs de supercellules d'un réacteur CANDU", *Ph.D. dissertation*, Ecole Polytechnique de Montréal (2001).
8. G. J. Wu and R. Roy, "Self Collision Rebalancing Technique for the MCI Characteristics Solver", *20th Annual Conference of Canadian Nuclear Society*, Toronto (2000).
9. G. J. Wu and R. Roy, "Parallelization of Characteristics solver for 3D neutron transport", *Lecture Notes in Computer Sciences*, **2131**, 344 (2002)
10. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, "PVM3 User's Guide and reference Manual", Report **ORNL/TM-12187**, Oak Ridge National Laboratory (1994).
11. W. Groop, E. Lusk, A. Skjellum, "Using MPI: Portable Parallel Programming with the Message-Passing Interface", MIT Press, Cambridge (1994).
12. For informations see: http://www.cerca.umontreal.ca

Table I. CPU time for the PVM implementation of MCI vs. the number of processors on Beowulf 1 when varying the tracking density and the track merging.

		Number of processors					
DT	Track merging	1	2	4	8	12	16
		MCI CPU time (s)					
2.5	No TMT	70.96	37.38	19.68	11.81	9.93	9.4
2.5	TMT-1	36.28	19.77	10.90	7.13	6.55	6.51
2.5	TMT-2	10.39	6.14	4.36	3.9	4.34	4.5
4	No TMT	114.49	58.92	31.11	17.53	13.66	12.3
4	TMT-1	54.14	28.31	15.67	9.88	8.43	8.28
4	TMT-2	12.29	7.27	4.76	4.25	5.24	5.12
10	No TMT	254.43	132.54	65.95	35.3	25.19	21.52
10	TMT-1	86.61	45.53	23.62	13.43	10.63	9.86
10	TMT-2	13.78	8.16	5.38	4.46	4.91	5.4

Table II. Speedup variations on both clusters for the PVM implementation of MCI (results are for No TMT or TMT-1 and a fixed density DT = 4).

		Number of processors					
Beowulf	Track merging	1	2	4	8	12	16
		Speedups					
1 (REF)	No TMT	1	1.94	3.68	6.53	8.68	9.60
2 (SMP)	No TMT	1	1.98	3.88	7.48	10.69	13.60
1 (REF)	TMT-1	1	1.89	3.45	5.47	6.42	6.37
2 (SMP)	TMT-1	1	1.96	3.78	6.99	9.38	12.28

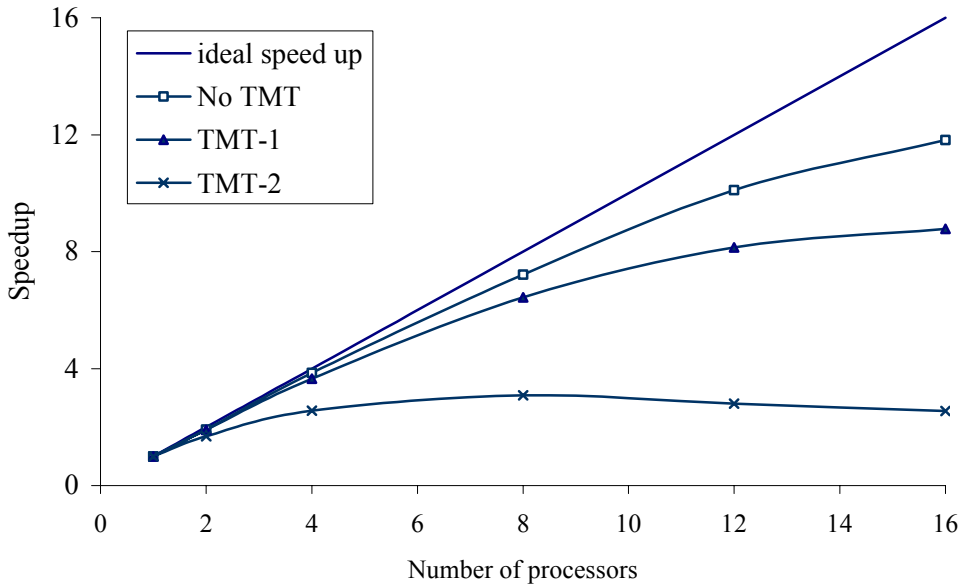

Figure 2. Speedup for different track merging options (PVM implementation)

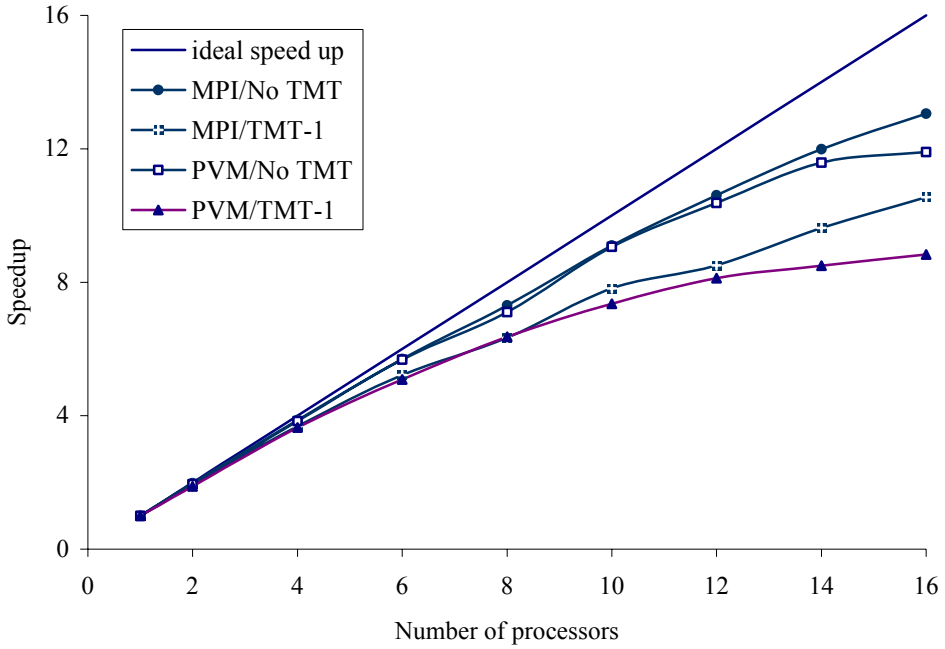

Figure 3. Speedups for PVM and MPI implementation with and without TMT (DT=10)