Nuclear Mathematical and Computational Sciences: A Century in review, A Century Anew
Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003*)*

PARALLEL DISTRIBUTION OF TRACKING FOR 3D NEUTRON TRANSPORT CALCULATION

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

ABSTRACT

In this paper, we present the recent advances in parallel software development for solving neutron transport problems in context of the 3D characteristics method. The method of characteristics is based on the resolution of the differential transport equation following the tracking lines to collect the local angular flux components. The parallelization of this solver is based on distributing a group of tracks, generated by the ray tracing procedure, on several processors. Different distributing and load balancing techniques are presented. Due to the excessive number of the tracks in the demanding context of 3D large-scale calculations, two different strategies for data management for these large amounts of data are implemented. The first strategy uses the regular files and the second one uses the database. To access the data, we use either the regular sequential mode or the MPI-IO as API.

Key Words: Neutron transport, characteristics method, parallel IO.

1. INTRODUCTION

The method of characteristics (MOC) has been largely used to solve the transport equation because of its good performance for large problems regarding to the Collision Probability (CP) method. Unlike the CP method, the MOC does not generate huge matrices, and still provides solutions as accurate as the CP method. Assuming that there are no interactions between neutrons in the medium and the neutrons traveling take a straight trajectories, the characteristics method are based on the resolution of the differential transport equation following the tracking lines to collect the local angular flux components. The solution of the transport equation using MOC can then be split into:

- Geometry preparation and tracking generation;
- Integration of differential transport equation using the generated tracking lines and then computation of the scalar flux.

In the context of the 3D large-scale calculations, the number of tracking lines can become prohibitive. Therefore, the data management for these tracks is a real challenge: at same time, we want to reduce the CPU time and to avoid the problems with the I/O operations which is often the bottleneck for such applications.

In recent years, the parallel computers get faster and more accessible economically. The great advances have been made in the CPU and the communication performances. Accordingly, the scientists are increasingly use parallel computers to solve problems that require a large amount of computing effort. Most of these applications need also to perform I/O operations, such as reading data and writing the results. In general, the I/O speed is slower than the CPU and communication speed. In order for parallel computers to be fully usable in solving large-scale problems, the I/O performance must be scalable and balanced with respect to the CPU and communication performance of the system. The studies have shown that an inappropriate Application Program Interface (API) is often the cause of poor I/O performance in applications [1].

Like in many scientific applications, the nuclear engineering applications, especially the neutron transport calculation for large domains, generate and use a large amount of data. The static transport equation must be solved to obtain the flux from a source Q :

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

where $\vec{r}$ is a spatial point in the domain D, $\hat{\Omega}$ is the solid angle and E is the energy.

In this paper, we will present the developments made to optimize the data storage of the tracks in context of the 3D characteristics formulation. To reduce the number of these tracks in the domain, Track Merging Technique (TMT) was implemented. Using this technique, two tracking lines crossing the same regions in the same order are merged together. The TMT is generally very efficient because more than half of tracking lines are merged together without loosing accuracy. It can also be easily implemented inside the ray tracing routine that computes the tracks. In order to access the tracking data stored in the files, we need to do the I/O operations. We will also present here the developments made to perform such operations. An MPI-IO implementation has been used as an API, so the processors belonging to the same communicator can access the data in the same file collectively. Another alternative and complementary method is to use a database to store and access the information about the tracking lines. Once the tracking lines are generated and stored, we have to answer to the question: how to distribute these tracking lines among different processors on parallel machines without loosing a performance? We will explain the load balancing strategies, including a global round-robin distribution of tracks where we forecast the calculation load implied by each track length and also a macro-band decomposition where tracks crossing the same regions are grouped together.

The paper is organized as follows: In section 2, we represent the brief review of the 3D MOC principles, the tracking generation procedure, and the resolution of the transport equation on one tracking line. Section 3 is dedicated to the strategies we use to distribute and to manage the data associated with the tracks; track merging and load balancing techniques will be explained. Section 4 shows numerical tests performed with the CANDU supercells. In the last section, we conclude by discussion on prospective future work in order to obtain a loosely coupled large-scale distributed transport solver in coming years.

2. THE 3D CHARACTERISTICS FORMULATION

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:

$$V_j\Phi_j=\int_{V_j} d^3r\int_{4\pi} d^2\Omega\Phi(\vec{r},\hat{\Omega}) \tag{2}$$

The main idea of the method is to solve the differential form of the Boltzmann equation following the tracking lines (also called "characteristics"). To involve the tracking lines, we change the reference using the local coordinates, equation (1) becomes:

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

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 $\chi_j(\vec{T},t)$ 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.

2.1. Tracking Generation Procedure

The d^4T element is decomposed into a solid angle element $d^2\Omega$ and 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 select direction $\hat{\Omega}$ for the starting point $\vec{p}$. The quadrature used in our code **DRAGON** [2] is the Equal Weight Quadrature EQ_N. The directions in EQ_N are choosing to be axially invariant.

Let $\hat{\Omega}$ be one of the values of angular quadrature, the set of the planar quadrature to the direction $\hat{\Omega}$ is a set of the equidistant and parallel lines to $\hat{\Omega}$: this is the characteristics lines or tracking lines. [3]

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. We obtain then K segments $\left[r_{k-1},r_k\right]$ and we define length of segment L_k as:

$$L_K=(\vec{r}_k-\vec{r}_{k-1}).\hat{\Omega},\quad k=1,2,....,K \tag{4}$$

and N_k is the region where the segment k is located.

For the opposite direction $\hat{\Omega}'=-\hat{\Omega}$, we use the same lines but in the opposite way:

$$\vec{r}_k'=\vec{r}_{K-k},\qquad k=1,2,...,K$$

$$L_k = L_{K+1-k}, \quad k=1,2,...,K .$$

This will allow us to decrease the number of tracking sweep.
The procedure is to define the geometry, to split the domain into regions, and generate the lines according to techniques explained above. To solve the differential transport equation, we have to associate to each region its nuclear data (cross sections).

2.2. The 3D characteristics Solver MCI

2.2.1. Solution on a single track

For a given track line $\vec{T}$, we define the crossing point $\vec{r}_0, \vec{r}_1, ..., \vec{r}_K$. For each segment of line, we must calculate the integrated angular flux:

$$L_k \bar{\phi}_k (\vec{T}) = \int_0^{L_k} dt \Phi(\vec{r}_{k+1} + t\hat{\Omega}, \hat{\Omega}) \tag{5}$$

The angular flux $\Phi(\vec{r}_{k+1} + t\hat{\Omega}, \hat{\Omega})$ is obtained by solving the differential transport equation on each segment k :

$$(\frac{d}{ds} + \Sigma_{N_k}) \Phi(\vec{r}_{k-1} + s\hat{\Omega}, \hat{\Omega}) = \frac{Q_{N_k}}{4\pi}; \; s \in [0, L_k] \tag{6}$$

Assuming isotropic input current at the external boundary and isotropic sources, a recursion is based on the following simple attenuation equation:

$$\phi_k = \phi_{k-1} e^{-\tau_k} + \frac{q_k}{\sigma_k}(1 - e^{-\tau_k}), \tag{7}$$

where local sources are $q_k = \frac{Q_{N_k}}{4\pi}$, local total cross section are $\sigma_k = \Sigma_{N_k}$ and total optical path 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 the inverse direction $-\hat{\Omega}$ using the same line; however, the input currents are not the same at both ends.

At this stage, the required local data is a collection of segment lengths L_k and numbers N_k for each region encountered along the line. The nuclear data needed are the local total cross section $\sigma_k = \Sigma_{N_k}$ and the scattering cross section from one group to it self $(\Sigma_s^{g \leftarrow g})_{N_k}$. No scattering matrices are necessary; this approach is important to preserve certain linearity in calculation and to preserve independency between the calculations performed on different tracks. This represents the atomic level for our application from which we can construct the overall solution for each region, each angle, and each energy group.

2.2.2. The MCI solver

Using equation (3), we get the integrated flux:

$$V_j \Phi_j = \int_\Gamma d^4T \sum_k \delta_{jN_k} L_k \overline{\phi}_k (\vec{T}) \tag{8}$$

where δ is the Kronecker symbol, and where k runs over all integers. All the characteristics lines are accepted, but only the contributions of segments crossing the same region j are added together.
To preserve the exact volumes, the characteristics lines have to be normalized. This is achieved by evaluating for each angle a numerical approximation V_j for volumes using segment lengths. We define then the angular volume for each direction $\hat{\Omega}_i$:

$$V_j(\hat{\Omega}_i) = \sum_n p_{i,n} \sum_k \delta_{jN_k} L_k$$

integrating over all angles, we obtain :

$$V_j = \frac{1}{4\pi} \sum_i \omega_i V_j(\hat{\Omega}_i)$$

where the $p_{i,n}$ is the weight associated with the line $\vec{T}_{i,n}$ having a point $\vec{p}_{i,n}$ center of the mesh as origin in the perpendicular plane $\pi_{\hat{\Omega}_i}$; and where ω_i is the weight corresponding to the direction $\hat{\Omega}_i$ in the quadrature. The normalization is done by multiplying L_k by the factor $\frac{V_j}{V_j(\hat{\Omega}_i)}$. The scalar flux in region j can be recovered by a reductive addition:

$$\Phi_j = -\frac{1}{4\pi \Sigma_j V_j} \sum_T \omega_T \sum_k \delta_{jN_k} \Delta\phi_k + \frac{Q_j}{\Sigma_j}, \tag{9}$$

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

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 number of lines is obviously much larger than the number of processors available. Therefore, the lines are grouped and each processor takes control of a group. At each iteration step, multicasting 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 a consistent unknown ordering. The scalar flux is then reconstructed on every single processor before the iteration procedure starts (multicasting communication process). In our MCI parallel solver, we use the message-passing interface MPI to communicate between the processes [4]. To reduce the number of iterations the self-collision rebalancing acceleration technique is used [5].

3. LOAD BALANCING AND DISTRIBUTION OF THE TRACKING

Since the time spent in calculation on one track depends linearly on the number of segments [3], the number of tracking distributed on each processor will be the major criteria to build our load balancing algorithm.

3.1. Track Merging Technique

In process of generating the tracking lines for a large domain, two successive lines 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 the other has $L-\varepsilon$. It was shown that the two tracks can be merged together with $O(\varepsilon^2)$ -order of error on the local angular flux. The resulting line takes the averaged segments length and additional weights of the original lines as its properties. Two levels of merging are applicable:

- **TMT-1**: one line can be merged to the next one as the ray tracing is performed, thus this equivalent to merging lines on a unique dimension;
- **TMT-2**: lines are stored and the reduction is done on both dimension of the perpendicular plane.

3.2. Load balancing strategies

3.2.1. Statistical load balancing

- Distribution on a beam (**SPLT** option): in this option, we subdivide the number of track on packs of $\frac{N_T}{N_p}$ tracks generated by EXCELT module. The first processor will work on the first $\frac{N_T}{N_p}$ generated, the second processor takes the next $\frac{N_T}{N_p}$ and so on. Figure 1 shows the distribution of the tracks using this option considering one direction with its associated perpendicular plane for four processors.
  For the domains that contain a lot of small regions, if we don't choose the right number of processors, one of the track groups may have a large number of segments than other ones.

- Round-robin distribution (**STRD** option): the tracks are distributed using constant intervals. However, the first processor take the tracks number $1, 1+N_p, 1+2N_p, ...$ the second processor $2, 2+N_p, 2+2N_p, ...$ In general, the processor i take the tracks $i+1 \mod N_p$.
  Using this option, it's statistically unexpected to have two groups of tracks having great difference in the average number of segments. We represent this option of distribution of tracks over four processors in Figure 1.

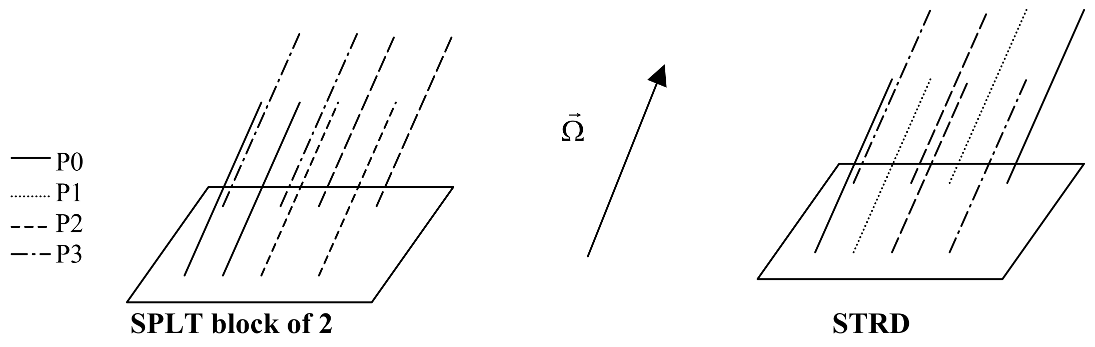

Figure 1. Distribution of the tracks on 4 processors for SPLT and STRD options.

- Distribution on angles (**ANGL** option): all the tracks having the same direction are grouped together. In this option, the number of processors has to be equal to the number of angles. So, each processor takes the tracks with the same direction.
  The disadvantage of this option is the limitation in the number of processors, for example, if we have four processors and five directions, one of the processors will be doubly loaded. For two directions, two of the four processors will don't be used (see Figure 2. for this option).

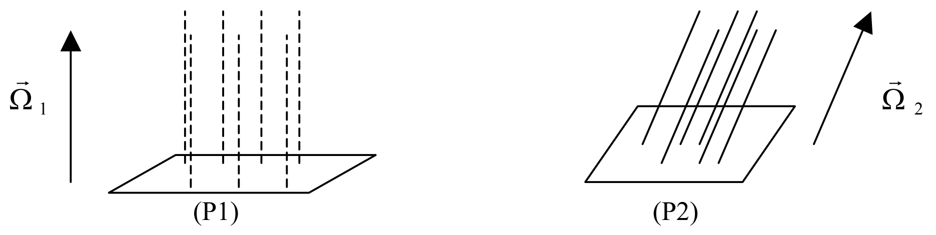

Figure 2. Distribution of the tracks with 2 directions and 2 processors for ANGL options.

3.2.2. Deterministic load balancing

The only one option was coded using a deterministic approach based on the calculation load of each single track is the macro-band (**MCRB** option). The macro-band can be defined as combination of a section of a geometric volume V_M and an angle $\hat{\Omega}_M$ that satisfied the following condition: all tracks $\vec{T}$ in $\hat{\Omega}$ direction are considered either they don't cross the volume V_M or they cross always the same regions in same order no matter where they traverse the macro-band.
In this option, the number 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 is still under investigation.

Once the one of these options is chose, two approaches for distributing the groups of tracks over the processors can be considered. The first approach is that one processor (master) distributes the tracks on other processors (slaves); the second approach is that every single processor generates all tracks by calling the EXCELT module and then takes its own group of tracks allocated after calling the TCHTRK module. In our calculation, we don't use the first approach because when the master processor is working to generate the tracks, the slaves remain in wait status. In addition to that, the size of the tracking file is generally very large so that the time of communication is non negligible. With the second approach we can avoid to have the processors in wait and also the communications.

3.3. Performance of the I/O operations

To generate tracks, to merge them, and to distribute them over processors, we need to do the I/O operations to access and to store the data (read/write). The speed and performance of the I/O operations depend essentially on File System (FS) and the Application Program Interface (API). Usually the tracks are stored in sequential binary file created by the EXCELT module. The sequential access in file with conventional APIs is very expensive when the size of the file is large as in our applications. In parallel calculation, this is also often the bottleneck. To eliminate such encountered problems and to improve a performance, we introduce two strategies to manage the data:

- using the files to store the data via a FS. To access the data, we use the sequential mode and MPI-IO as API;
- using a database to store the data; a high level interface integrating MPI-IO is used to access the data.

In the following, we will describe both of these strategies.

3.3.1. MPI-IO implementation

To avoid the concurrency access problems that we faced especially when we run our applications on SMP parallel machine, and to improve, in general, the performance for the read/write operations on files, we have implemented the MPI-IO for a file I/O. MPI-IO implementation, ROMIO, a part of the MPI-2 standard [6], is an interface designed specifically for portable high-performance parallel I/O. MPI-IO allows the users to specify a noncontiguous access pattern and read or write all the data with a single I/O function call. It allows also performing collective I/O requests of group of processes.
In order to achieve a good performance, we use the collective read/write functions. We also use the explicit collective pointer with offsets to locate the data directly in file, this allow us to reduce the access time.

3.3.2. Database implementation

For large problems, the size of the tracking file can reaches hundreds of Megabytes to Gigabytes. Thus, storing and managing this large amount of data is very difficult and it stresses the computer resources. We use the database to store the data as an alternative. A high level interface integrating MPI-IO is used to store the data in and to retrieve the data from the database. A database tables are used for storing: the mean table contain the name of the tracking file, application options and the dimension of the problem like the number of tracks, the spatial dimension, and number of regions… On all our applications, we use MySQL as database server [7].

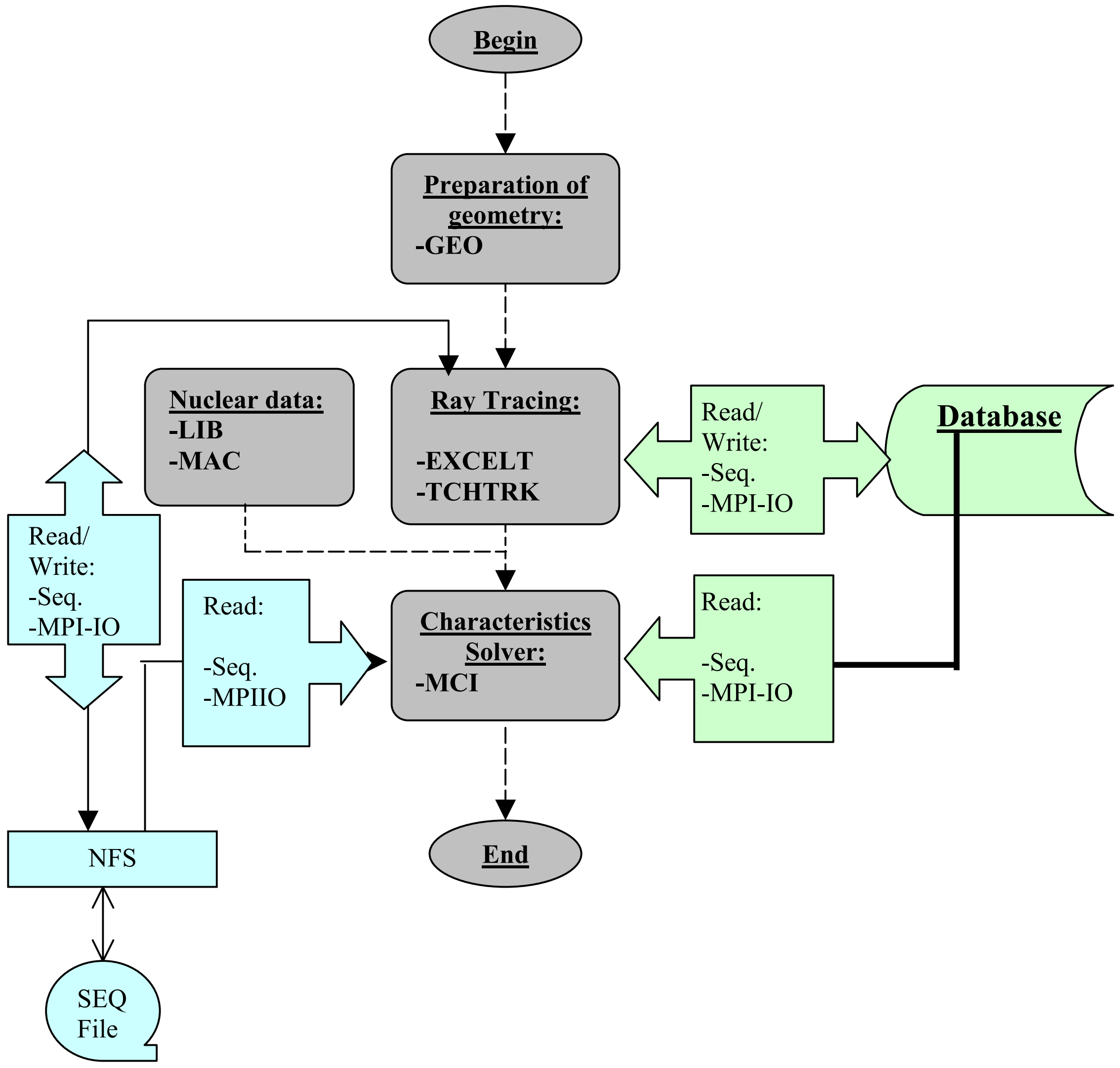

Figure 3. The transport calculation scheme using either a file or a database.

In Figure 3., we represent the transport calculation scheme using the DRAGON code with the two strategies described above. The first step is to prepare geometry of the problem, to define meshes using the GEO module. This geometry is passed to the ray tracing generator and a tracking file is created by the EXCELT module. At this step, we have to choose either to use regular files or the database to manage the data:

a) using files: in this case, every processor will have a copy of the tracking file by calling the EXCELT module and will be used by the TCHTRK module to do the merging using the TMT techniques and then each processes will take a group of tracking allocated to him according to one of the load-balancing options. At the end, all the a file containing the tracking data is created by each processes; this will be the file used by the characteristics solver MCI to solve the transport equation on each processor and then the partial results will be communicated between a processes using the message passing as we have explained above. The read/write operations use either the sequential mode or the MPI-IO interface.

b) using the database: one process will create the database and stores all the tracking data generated by the EXCELT module in the database using the database tables. After that, the merging operations will be performed using the operations on the database tables; and then using one of the load-balancing options, we create for each process a table containing its group of data and the number of processor in the communicator. Once this is done, each process will access its group of data, as client of the database (server), from the MCI module via the read/write high level interface which use the MPI-IO.

4. NUMERICAL RESULTS

Tests were done for CANDU6 adjuster rod supercell calculations in 3D geometry. Two horizontal fuel bundles (a bundle is a cluster grouped of rods containing Uranium oxide) and one vertical absorber rod at center of the assembly compose the domain. There is 1/8 symmetry and angular quadrature EQ$_4$ is used. Dimensions are $N_G=89$ (number of energy groups), $N_R=48$ (number of regions).
Tests were essentially performed on a two Beowulf clusters located at Center for Research on Computation and its Applications CERCA [8]:

- **Beowulf 1** is 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 small SMP composed of 8 QuadXeon multiprocessors; each of these 8 nodes has 4 processors sharing a 4 GB memory. Nodes are connected with a 1 Gb/s Myrinet switch.

Our tests are done for two different densities of tracks (DT) 4.0 and 10.0 (tracks/cm^2). However, in some cases, with using the merging operations we can loose performance [3]. It is clearly shown, for especially the TMT-2 option in Figure 4, that the performance is so poor because of the increasing time of the communications over the computations. This is due to the insufficient number of the tracks in the domain. For large scale problems, where the number of tracks may be excessive, the TMT options will be more efficient. In Table I, II, and III, we present the distribution load (in Bytes) on each processor for three load-balancing options that we described above, respectively: SPLT, STRD, and ANGL. The STRD option is most balanced one, each of processors is loaded almost equally than others. With SPLT option, we see some differences of the size of load on each processor. The ANGL option shows large differences in load sizes, in addition to the limitation on the number of processors. This can be due to the interaction of the cosines directions with the cylinders main axis; in fact, for some angles, the various regions are more likely to be crossed by tracks. The limit case would be a solid angle in the same direction

as the cylinder axis, in which case the cylindrical regions would be crossed only once. The results for the MCRB option are shown in Table IV. In this option, an equivalent number of segments are distributed on each processor. Considering the load sizes on each processor, the MCRB option can be set between STRD and SPLT options. No TMT option is allowed in this case. In the following tests, we will therefore use only the STRD option.
In Figure 5, we represent the speed up curves corresponding to two different calculations: the first one is done on SMP machine (Beowulf 2) using the MPI-IO and the second one are done on the Beowulf 1. We can see that the performance is better on the Beowulf 2 with MPI-IO because of the speed of the network and the performance of the collective access to the files allowed by the MPI-IO. In the past, we had some concurrently access problems when running the applications on Beowulf 2 with the sequential mode access. All the results shown were performed with the FS option.

5. CONCLUSION

We have presented in this paper the parallel software development implemented in our 3D characteristics solver. Different techniques for distributing and merging the tracks are used. To manage the data associated with tracks, we can use either regular files or a database to store and access the data. MPI-IO was also implemented to allow collective file access. The use of the database allows us to access a set of data in same time and to manipulate data as arrays, this introduces another new approach to solve the problem.

Next step would be to subdivide the problem on sets where a minimum of nuclear data is locally collected: using a solver on groups of tracks. These mobile agents will do the most of work almost independently one of another. To attempt to solve the problem of the whole-core, we will distribute these agents on a heterogeneous grid of computers.

ACKNOWLEDGMENTS

This work has been carried out partly with the help of a grant from the Natural Science and Engineering Research Council of Canada.

REFERENCES

1. R. Thakur, W. Gropp, and E. Lusk “Data Sieving and Collective I/O in ROMIO,” *Proc. Of the 7th Symposium on the Frontiers of Massively Parallel Computation,* pp. 182-189 (1999).
2. G.Marleau, A Hebert and R. Roy, “A User’s Guide for DRAGON”, *Technical Report* IGE-174 Rev. 3., École Polytechnique de Montréal, December (1997). Available at http://www2.polymtl.ca/nucl
3. G. J. Wu, “Développement d’une methode des caracteristiques tridimensionelle et application aux calculs de supercellules d’un reacteur CANDU,” *Ph.D. dissertation,* Ecole Polytechnique de Montreal (2001).
4. M. Dahmani, G. J. Wu, R. Roy, and J. Koclas, “Development and Parallelization of the Three-Dimensional Characteristics Solver MCI of DRAGON,” *Proc. Of PHYSOR 2002,* October 7-10, (2002).

5. G. J. Wu, and R. Roy, "Self Collision Rebalancing Technique for the MCI Characteristics Solver," *20th Annual Conference of Canadian Nuclear Society,* Toronto (2000).
6. Message Passing Interface Forum. "MPI-2: Extensions to the Message-Passing Interface," http://www.mpi-forum.org/docs/docs.html (1997).
7. MySQL Reference Manual. http://www.mysql.com, (1999).
8. For information see: http://www.cerca.umontreal.ca

Table I. The distribution of the tracks on processors for the SPLT option (in Bytes)

	Number of processors			
Merging	**2**	**4**	**8**	
TMT-1	3463004 3764920	1783116 1679908 1991080 1773860	878376 904760 811704 868224	793920 979960 987744 1003356
TMT-2	413060 475092	221288 191792 261576 213536	105676 115492 94004 97716	94900 118372 141196 120916

Table II. The distribution of the tracks on processors for the STRD option (in Bytes)

	Number of processors			
Merging	**2**	**4**	**8**	
TMT-1	3613224 3614700	1806456 1807756 1806945 1806788	902884 903872 904344 903656	903592 903940 903152 902640
TMT-2	444436 443716	222616 222056 221680 221840	111152 110816 110848 110760	111260 111484 110852 111100

Table III. The distribution of the tracks on processors for the ANGL option (in Bytes)

	Number of angles and processors	
Merging	**3**	**6**
TMT-1	2271152 2244576 2712216	2117432 2243744 2676188 2107908 2048536 2698076
TMT-2	291028 220160 376984	262560 242168 434628 169020 199200 364824

Table IV. The distribution of the tracks on processors for the macro-band option (MCRB) (in Bytes)

Number of processors				
1	**2**	**4**	**8**	
11812268	5435376 6377848	2587036 2846704 3158712 3222684	1376168 1211824 1491400 1356260	1398244 1761424 1448072 1775568

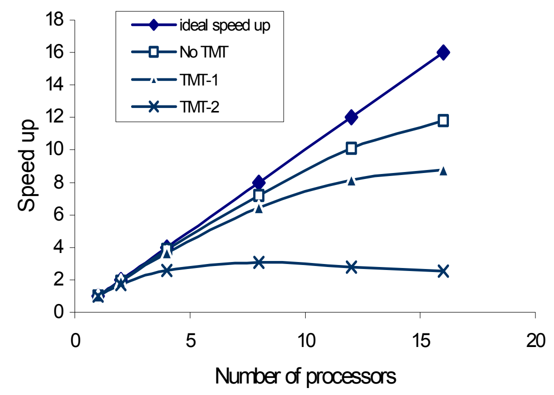

Figure 4. Speed up with and without TMT (DT=10)

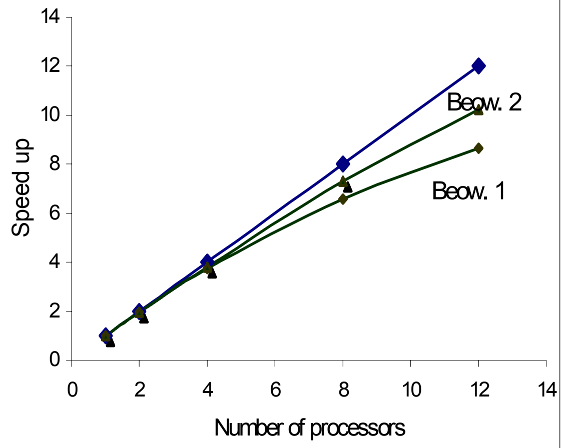

Figure 5. Speed up on two Beowulfs using files without TMT