PHYSOR-2006, ANS Topical Meeting on Reactor Physics
Organized and hosted by the Canadian Nuclear Society. Vancouver, BC, Canada. 2006 September 10-14

New Algorithms for 3D Characteristics Solvers

R. ROY*
École Polytechnique de Montréal
P.O.Box 6079, Station CV, Montréal, Québec, Canada H3C 3A7

Abstract

Due to the constant increase in high performance computing resources, the Method of Characteristics (MoC) utilization can now be extended to solve large-scale deterministic problems. In MoC applications, it is possible to benefit from the so-called nested parallelism, where a distributed-memory machine (called a cluster) has embedded SMP nodes. Inside each SMP node, a shared-memory programming model is applied. Between nodes or at the cluster level, the message passing is used. In this paper, we focus on new developments that enable nuclear engineers and analysts to obtain accurate 3D large-scale transport calculations using MoC solvers. Some performance results are also given for numerical simulations on hybrid machines.

1. Introduction

This paper will focus on new algorithms that are currently developed for solving 3D transport problems. The method of characteristics (MoC) solves the differential form of the transport equation by following the characteristics (tracking lines) which follow particle paths. [1] The scalar flux is built by collecting all mean angular fluxes obtained in terms of the entering angular fluxes and the sources inside the domain. A sequential 3D characteristics solver was developped to solve transport problems specific to CANDU reactors. [2] In the last years, several extensions were done on this MoC solver regarding boundary conditions, anisotropic scattering and sources, and the use of various acceleration techniques. These developments lead to a generic sequential module now integrated into the recent version of DRAGON under the name `MCU`: for "Méthode des Caractéristiques Universelle," meaning for Universal MoC. [3]

In this presentation, I will review some consistent and scalable algorithms for the MoC solver to accomodate greater computational power (new architectures, more processors...). Parallel algorithms are often a blend of task scheduling in time with proper data distribution in space. Linear MoC solvers have an appropriate granularity for large-scale parallelism: in fact, the local angular flux along each characteristics can be computed without any knowledge of what is outside its path except the nuclear properties and sources that are crossed. This last feature makes a generic parallel MoC solver easy to model and to scale. The outline of the paper is the following. Section 2 is devoted to the basic equations governing MoC solvers. Section 3 gives some acceleration techniques that are used in `MCU`. Section 4 describes some features of nowadays high-performance computing (HPC) ressources. In Section 5, I introduce the generic parallel MoC solver and some results are shown for hybrid machines. Conclusions are drawn in the last section.

*E-mail: robert.roy@polymtl.ca

2. MoC basic equations

In this section, I will review the basic equations used in MoC solvers. Among deterministic solvers, MoC solvers are now commonly used because of the simplicity of their programming and their capacity to be used for any geometry.

2.1 The characteristics formulation of the transport equation

First, the angular domain is covered by choosing a quadrature set of solid angles $(\hat{\Omega}_i, \omega_i)$. For any direction i, a whole set of tracks $\vec{T}_{i,n}$ is generated. When traveling across different regions, the particle beam following the characteristics crosses segments identified by their lengths L_k and the region numbers N_k. Assume that segment k crosses region j, the relationship between the incoming and outgoing angular flux is given by

$$^{\text{out}}\phi_j^g(k) = {}^{\text{in}}\phi_j^g(k) + \left[\frac{Q_j^g}{\Sigma_j^g} - {}^{\text{in}}\phi_j^g(k)\right]\{1 - e^{-\Sigma_j^g L_k}\}, \tag{1}$$

and the average scalar flux in region j can be calculated using

$$\Phi_j^g = \frac{Q_j^g}{\Sigma_j^g} + \frac{1}{\Sigma_j^g V_j}\sum_i \omega_i \sum_n \pi_{i,n} \sum_k \delta_{jN_k}\Delta_{i,n,k}^g; \tag{2}$$

where $\Delta_{i,n,k}^g = {}^{\text{in}}\phi_j^g(k) - {}^{\text{out}}\phi_j^g(k)$ is the flux difference on segment k and $\pi_{i,n}$ is the weight associated with track $\vec{T}_{i,n}$.

2.2 Link with the collision probability formulation

The first-flight collision probability (CP) formulation of the transport equation uses the same tracking lines. In the CP method, a linear system whose unknowns are the average flux values of each region (and the currents at the external boundary) is built using the exponential attenuation factors. Unfortunately, the group-dependent CP matrices are dense, and there is no easy way to scale the CP formulation for large problems. [2] have shown that both MOC and CP formulations are equivalent and lead to similar numerical results. The main reason that explains why CP solvers are no longer used in industry is that these solvers cannot scale to large problems; however, the CP solvers are still used for small cell transport problems.

3. Acceleration techniques for MOC solvers

To increase the performance of MOC solvers, a track merging technique can be used to reduce the number of characteristics. The use of exponential tables in Eq. 1 can also help. Acceleration of the scattering and fission iterations is another goal.

3.1 `SCR` preconditioning

This technique was developed in order to rebalance the energy distribution of the scalar flux for each region separately. Assuming G groups, the source is composed of a fission source $S_{f,j}^g$ and a scattering term. The source term after the n-th inner iteration is:

$$Q_j^{g,n} = \sum_{g'=1}^{G} \Sigma_{s,j}^{g\leftarrow g'}\Phi_j^{g',n} + S_{f,j}^g. \tag{3}$$

The local effect of the segment source is driven by the self-collision probabilities in

$$\Phi_j^g = \Phi_{\text{in,j}}^g + \tilde{p}_{jj}^g Q_j^g \tag{4}$$

where $\Phi^g_{\text{in,j}}$ is the flux component generated *without* taking into account the local sources. After substituting Eq. (3) in Eq. (4), we obtain the local rebalancing system for the region V_j:

$$\sum_{g'=1}^{G} \left(\delta_{gg'} - \tilde{p}^g_{jj} \Sigma^{g \leftarrow g'}_{s,j} \right) \Phi^{g',n+1}_j = \Phi^{g,n+\frac{1}{2}}_{\text{in,j}} + \tilde{p}^g_{jj} S^g_{fj} \tag{5}$$

The above SCR system is solved by an iterative method for all the regions one after another. Finally, the outward currents are updated using the updated source taking into account current components. The SCR technique can be easily implemented and needs only one more diagonal matrix to be saved. [2]

3.2 Combining GMRES with SCR

The Generalized Minimal RESidual (GMRES) method, using a Krylov subspace projection, is adapted and implemented to accelerate a 3D iterative transport solver based on the characteristics method. Another acceleration technique called the Self-Collision Rebalancing technique (SCR) can also be used to accelerate the solution or as a left preconditioner for GMRES . The GMRES method is usually used to solve a linear algebraic system ($Ax = b$). It uses $\mathcal{K}(r^{(o)}, A)$ as projection subspace and $A\mathcal{K}(r^{(o)}, A)$ for the orthogonalization of the residual.

To implement the GMRES iterative method, the characteristics equations are derived in linear algebra formalism by using the equivalence between the characteristics and the collision probability formulation as explained earlier to end up with a linear algebraic system involving fluxes and currents. Numerical results show good performance of the GMRES technique especially for the cases presenting large material heterogeneity with a scattering ratio close to 1 (see Table 1). By using the SCR as preconditioner, the number of these iterations is reduced. SCR does not reach convergence for the two last tests when c is close to 1. More details are available in [5].

Table 1: Number of iterations for different scattering ratio

Method	GMRES	GMRES + SCR	SCR
$c = 0.9$	17	15	63
$c = 0.999$	28	19	-
$c = 0.99999$	55	19	-

4. New features of HPC architectures

In order to obtain scalable solvers, computer resources must be expanded at different levels. Computer hardware and architecture can now support an increase in the number of processors, as well as improvements in their communication systems. In this section, I will review the main features of the modern high-performance computing ressources.

4.1 Going from 32-bit to 64-bit architectures

New computer architectures, especially the ones needed for extensive floating point operations, are 64-bit. High performance computer nodes adresses more memory and can be used with increased accuracy. In order to port the 32-bit version of DRAGON to 64-bit, the following steps were needed:

- increase the word representation for integers and reals to 8 bytes;
- allow large file system (LFS) support for files over 2Gb;
- modify the record length definition when opening some files;
- change the allocation routine to double size the dynamic arrays.

4.2 Shared memory computers

In the shared-memory programming model, many tasks can run in parallel on nodes. The performance of the this model is limited essentially by synchronization issues between the tasks. In the context of characteristics transport solvers, thread-level parallelism is easy to implement for the energy groups: the characteristics, available in the shared memory, are pushed into each processor as its cache already contains some cross-section data. To scale well, this energy group sharing process must be load balanced between nodes.

4.3 Distributed computers

In the message-passing programming model, several processors also run in parallel, but they communicate with each other. In this context, we generally use a distributed tracking file. [4] Newer parallel algorithms have also been tried: a database containing the characteristics, the characteristics are recomputed on-the-fly at each iteration.

4.4 High-performance networks

The cost of network material has drastically decreased over the last years. Nowadays, most clusters can use high-performance switched network such as:

- Gigabit Ethernet,
- Myrinet Fiber, or
- others (QsNet, SCI, VIA,...).

The bandwidth obtained by these modern networks helps to decrease the per-word communication time, allowing for faster parallel algorithms. However, latencies (of the order of μs) are still high for applications with sustained repetitive communications.

5. Parallel algorithms for MoC solvers

Although MoC seems simple to implement, its application to *realistic* 3D applications is made difficult by the fact that sophisticated algorithms are often necessary to deal with problems with many regions.

5.1 Parallel algorithm

At the solver level, each processor takes control of batches of tracks to accumulate contributions to average angular fluxes; a partial sum of flux differences is accumulated by region and energy group in a flux scalar array. At each inner iteration, the processors communicate their results to all other processes by using a reduce/broadcast procedure. These sums are defined by

$$\Gamma_p \phi_j^g = \frac{1}{\Sigma_j^g V_j} \sum_{(i,n)\in M^{-1}(p)} \omega_i \pi_{i,n} \sum_k \delta_{jN_k} \Delta_{i,n,k}^g; \tag{6}$$

where ϕ_j^g is the flux for region j and energy group g. After the reduction operation, at the end of the inner loop, all processes have the same copy of the scalar flux

$$\phi_j^g = \frac{Q_j^g}{\Sigma_j^g} + \sum_{p=0}^{P} \Gamma_p \phi_j^g. \tag{7}$$

Figure 1: The reduction process used for a generic MoC parallel solver.

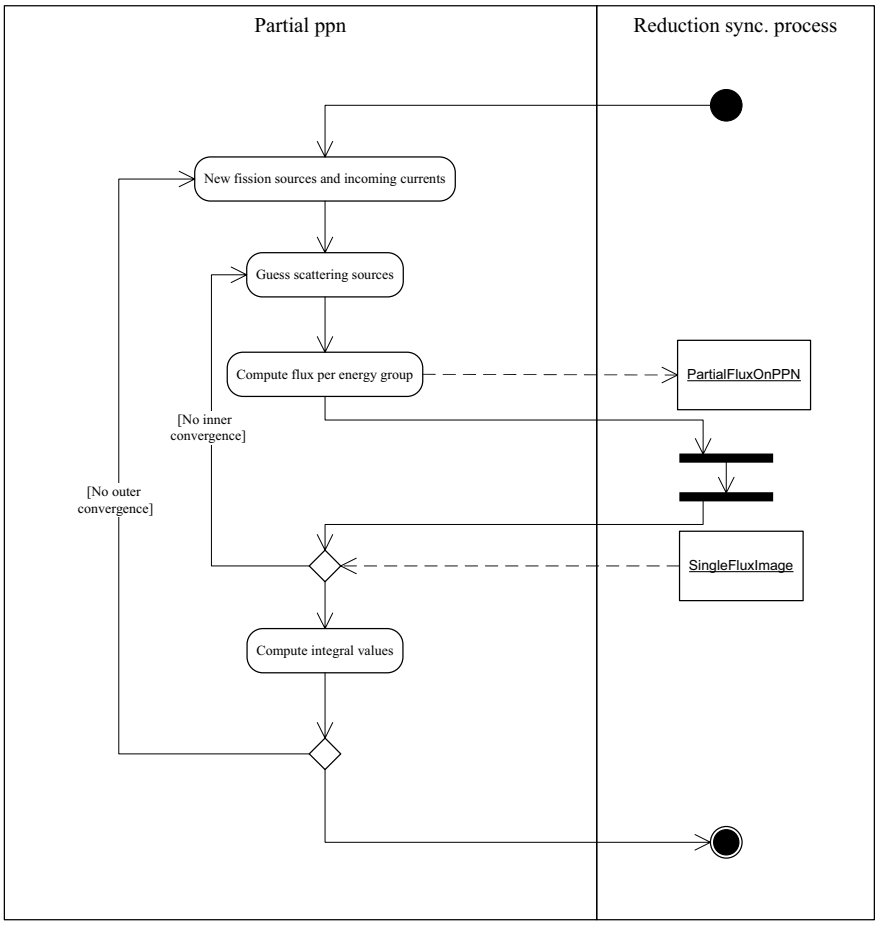


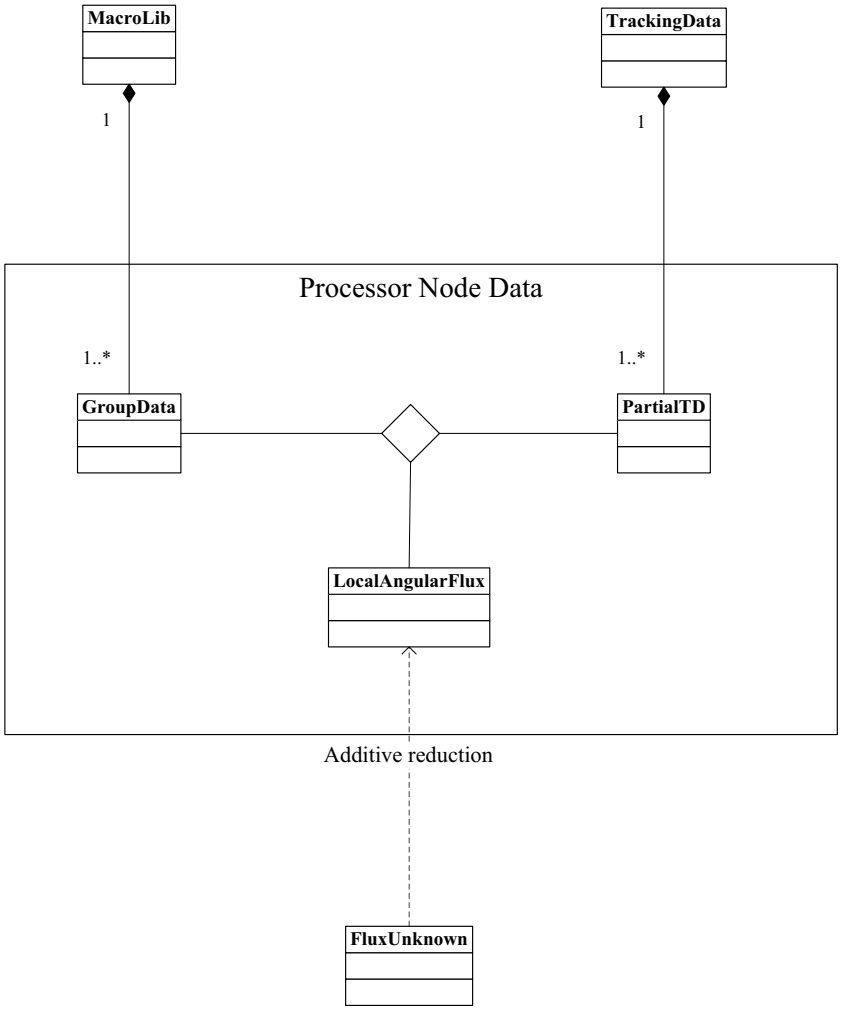
Figure 1 describes the reduction process that is used in MoC parallel solvers. The main idea is to distribute the tracks over processors. In the case of hybrid machines, each node can have more that one processor, and we speak of *ppn*: processors per node. The minimal requirements to obtain local angular flux values over each processor of a node are that:

- each processor generates (or read) a part of the tracking data, that is a collection of characteristics that covers part of the domain;

- each processor generates (or read) the cross-section data, here multigroup cross sections, pertinent to the regions covered by its collection of chararacteristics.

Figure 2 shows all data needed by a processor in order to participate to a MoC parallel calculations. Calculations of new fission sources and currents, as well as scattering from other groups, need a multigroup flux map that is consistent for every node. This parallel process of reducing every local angular fluxes into a single flux image made available for next iteration of the solver is an *heartbeat* algorithm.

Figure 2: Minimal requirements for a generic heartbeat-like MoC solver.



5.2 Putting it all together

In the last year, recent advances have been done on the parallelization on *hybrid* machines. [6] These parallel machines are composed of nodes with more than one processor. For newer architecture containing nodes with several processors (like cluster of SMPs), the communication overhead function must take into account the two mechanisms used for the communication:

- the shared memory when processors belong to the same node;
- the network communication when the processors belong to the different nodes.

Let us assume that there are $\mathcal{M}$ processors per node and that the total number of processors in the parallel machine is P. If the recursive doubling algorithm for communications is applied, the first $\log \mathcal{M}$ communication steps will not involve message passing between nodes. Thus, the communication overhead function is

$$T_0 = P\left(\log P - \log \mathcal{M}\right)\Lambda_{\mathrm{comm,Net}} + P(\log \mathcal{M})\Lambda_{\mathrm{comm,sh}}; \tag{8}$$

where parameters $\Lambda_{\mathrm{comm,Net}}$ and $\Lambda_{\mathrm{comm,sh}}$ are linear functions expressing a single reduction message using the network and the shared memory respectively. These linear functions have slopes related to the problem size (in fact, the length of the multigroup flux vector) and constant latency term.

5.3 Performance tests for parallel MoC

The efficiency of the parallel system is given by

$$\mathcal{E} = \frac{\mathcal{S}}{P} = \frac{T_{seq}}{PT_{par}(P)} \tag{9}$$

where $\mathcal{S}$ is the speedup for P processors, T_{seq} is the sequential time and $T_{par}(P)$ is the parallel time. After neglecting second order terms, it is possible to show that the efficiency, for a given architecture, can be approximated by:

$$\mathcal{E} \sim \frac{1}{1 + P\lceil \log P \rceil / \mathcal{D}}. \tag{10}$$

where $\mathcal{D}$ represents the domain dimensionality. [6] This dimensionality parameter obviously depends on the problem size (number of regions and external surfaces) and the number of tracks, but also on the ratio of calculation time and communication time at each iteration.

Here we are interested on performance tests on different hybrid machines. For those tests, we use the following hybrid machines:

- *Charybde* : 32-bit cluster of SMPs composed of 8 QuadXeon multiprocessors, each of these 8 nodes has 4 processors sharing a 4 GB memory. Nodes are connected with a Myrinet switch.
- *Hydra* : 64-bit cluster NUMAs composed of 32 nodes, each node has two AMD Opteron processors (2 GHz) sharing 5 GB memory. Nodes are connected with Myrinet Fiber.

In Table 2, results are given for MoC problems of similar dimensionality (although the number of tracks was almost doubled between $\mathcal{D} = 49747$ and $\mathcal{D} = 55720$). These results show that similar dimensionality leads to similar efficiencies; the efficiency model seems to work

Table 2: Measured efficiencies for two test cases of similar dimensionality

Performance test		Number of processors				
Machine	$\mathcal{D}$	2	4	8	16	32
Charybde	49747	99.1%	97.2%	90.8%	80.4%	60.2%
Charybde	55720	99.1%	96.4%	89.7%	80.1%	60.7%
Hydra	28291	99.4%	93.8%	87.4%	71.2%	52.6%
Hydra	31687	99.5%	94.3%	84.8%	67.7%	48.2%

well. When comparing both hybrid machines, one can see that the efficiencies obtained with *Hydra* are smaller than those obtained with *Charybde*, this can be explained by the imbalance between the computational power (greater by a factor of 5) and the network parameters (almost the same). The dimensionality on *Hydra* is about 57% less than the one in *Charybde* due to this imbalance.

5.4 Characteristics on the fly

Recently, a new approach was taken to deal with the issues of using large-scale HPC servers for scalable deterministic transport MoC solvers. When tracks are stored in files, these must be read at each iteration. The `MCI` parallel algorithm assumes that the tracks are available in files. The main scalability problem for large-scale problems is to store and access a large number of characteristics.

The `MCG` parallel algorithm computes characteristics (that is track lengths and zone numbers crossing the domain) on the fly at each iteration step. In Table 3, efficiencies for both `MCI` and `MCG` algorithms are compared. Although the global CPU time spent in `MCG` is larger than in `MCP`, the tracks recalculation has little effect on the efficiencies. This means that `MCG` is a scalable parallel MoC solver with no I/O bound.

Table 3: Measured efficiencies for `MCI` and `MCG` with same problem dimensionality. Machine is *Charybde*

Performance test		Number of processors				
Code	$\mathcal{D}$	2	4	8	16	32
MCI	55720	99.1%	96.4%	89.7%	80.1%	60.7%
MCG	55720	99.1%	95.9%	89.2%	79.8%	60.1%

6. Conclusion

The aim of this paper was to present the current state of MoC parallel algorithms, under the hypothesis that the code is kept as portable as possible with regard to different types of parallel machines. Shared memory multiprocessors are now standard nodes in high performance computer clusters and network media are faster than ever. Numerical results were thus given for two hybrid shared/distributed memory parallel computer architectures. These results show that it is possible to accurately predict the performance of such algorithms by using the problem dimensionaly. Such scalability prediction is very important in order to select the optimal cluster architecture for future industrial large-scale transport calculations.

Acknowledgements

This work has been carried out partly with the help of grants from the Natural Science and Engineering Research Council of Canada. The author would also like to thank Mohamed DAHMANI and Louis-Alexandre LECLAIRE for their help in collection data.

References

1) J.R. Askew, "A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries," Report AEEW-M 1108, United Kingdom Atomic Energy Establishment, Winfrith (1972).

2) G.J. Wu and R. Roy, "A New Characteristics Algorithm for 3D Transport Calculations," *Ann. Nucl. Energy* **30**, 1-16 (2003) *and* "Acceleration Techniques for Trajectory-based Deterministic 3D Transport Solvers," *Ann. Nucl. Energy* **30**, 567-583 (2003).

3) G. Marleau, A. Hébert and R. Roy, "A User's Guide for DRAGON. Version 3.05,"Report IGE-xxx, École Polytechnique de Montréal (2005).

4) M. Dahmani, R. Roy and J. Koclas, "Parallel Distribution of Tracking for 3D Neutron Transport Calculation," Int. Conf. on Nuclear Mathematical and Computational Sciences, on CD-ROM, Gatlinburg, Tennessee (2003).

5) M. Dahmani, R. Le Tellier, R. Roy and A. Hébert, "An efficient Preconditioning Technique using Krylov Subspace Methods for 3D Characteristics Solvers," *Ann. Nucl. Energy* **32**, 876-896 (2005).

6) M. Dahmani and R. Roy, "Solving Three-Dimensional Large-Scale Neutron Transport Problems using Hybrid Shared-Distributed Parallelism and Characteristics Method," submitted to *Nucl. Sci. Eng.* (2006).