

REACTOR ANALYSIS METHODS

Cosponsored by the Reactor Physics, the Mathematics and Computation,
and the Radiation Protection and Shielding Divisions

1. Parallel MCNP Monte Carlo Transport Calculations with MPI, John C. Wagner, Alireza Haghghat (Penn State)

The steady increase in computational performance has made Monte Carlo calculations for large/complex systems possible. However, in order to make these calculations practical, order of magnitude increases in performance are necessary. The Monte Carlo method is inherently parallel (particles are simulated independently) and thus has the potential for near-linear speedup with respect to the number of processors. Further, the ever-increasing accessibility of parallel computers, such as workstation clusters, facilitates the practical use of parallel Monte Carlo.

Recognizing the nature of the Monte Carlo method and the trends in available computing, the code developers at Los Alamos National Laboratory implemented the message-passing software package parallel virtual machine¹ (PVM) into the general-purpose Monte Carlo radiation transport code MCNP (version 4A) (Ref. 2). The PVM package was chosen by the MCNP code developers because it supports a variety of communication networks, several UNIX platforms, and heterogeneous computer systems. This PVM version of MCNP has been shown to produce speedups that approach the number of processors³ and thus, is a very useful tool for transport analysis.

Due to software incompatibilities on the local IBM SP2, PVM has not been available, and thus it is not possible to take advantage of this useful tool. Hence, it became necessary to implement an alternative message-passing library package into MCNP. Because the message-passing interface⁴ (MPI) is supported on the local system, takes advantage of the high-speed communication switches in the SP2, and is considered to be the emerging standard, it was selected.

IMPLEMENTATION OF MPI IN MCNP

As mentioned, the current version of MCNP makes use of the PVM message-passing library. The parallel MCNP calculation proceeds in the following manner:

1. The master task reads and processes the input file and sends the subtasks the common information.
2. The master task then calculates the workload for each processor, based on the total number of histories to be done and a user-supplied synchronization setting, and sends the subtasks their work assignment.
3. The master task saves the results of its calculations, receives the results from the subtasks, and then combines the results to produce the tally and statistical information for output processing.

Steps 2 and 3 are then repeated until the total number of particle histories have been completed.

The process of modifying MCNP to make use of the MPI message-passing library consisted mostly of translating PVM calls to MPI calls and constructing homemade packing rou-

tines. Calls to PVM-specific routines, and their corresponding arguments, were replaced with calls to MPI-specific routines that perform similar functions, and homemade data packing routines were written to simulate the built-in PVM packing routines. Further modifications were necessary to perform proper initiation and termination. In the PVM environment, the master has complete control and spawns and terminates the slave processes in a smooth and straightforward manner. On the other hand, in the MPI environment, each processor controls itself and therefore must initiate and terminate itself. This difference required the inclusion of additional logical statements to account for both normal and abnormal terminating situations.

ANALYSIS OF PERFORMANCE

The parallel performance of any code depends on several factors. These include

1. grain size (number of operations per communication or synchronization)
2. fraction of the code that is executed in parallel
3. amount of data that is passed between processors
4. load balance.

With the MCNP code, the grain size can be controlled in non-criticality types of problems by simply reducing the number of synchronizations. However, this will result in fewer entries in the MCNP tally fluctuation chart (a reduced amount of statistical information). In the PVM implementation of MCNP, the default is to synchronize 10 times per calculation² and thus produce a reasonable description of the statistical progression of the problem. For criticality problems, on the other hand, processors must share information after each cycle to estimate the fission generation and to provide the fission source for the next cycle. However, depending on the complexity of the fission source distribution, it may be possible to increase the number of histories per cycle and decrease the number of total cycles, thus increasing the granularity.

The fraction of MCNP that is not executed in parallel is the initial input file processing, the tally processing, and the final output processing. Generally for large problems, which require a correspondingly large number of histories, the total calculation time is large, and thus the parallel fraction is near unity.

Load imbalance is possible for problems with small granularity (relatively few histories per processor) and running on nondedicated machines. The small granularity load imbalances can be directly related to the random nature of particle transport (i.e., particle histories may differ significantly, and thus the time required to simulate them may also differ significantly). The current version of MCNP offers no dynamic load balancing.

To analyze parallel performance, one must be aware of several relevant issues, such as

1. Parallel processing is important for performing computationally intensive calculations for large/complex systems.

TABLE I
Parallel Performance

Number of Processors	Shielding Problem			Criticality Problem		
	Wall-Clock Time ^a (min)	Speedup (T_s/T_p)	Efficiency (%)	Wall-Clock Time ^a (min)	Speedup (T_s/T_p)	Efficiency (%)
1	91.5	—	—	82.2	—	—
2	49.1	1.9	93.1	42.0	1.9	97.7
4	28.1	3.3	81.5	22.0	3.7	93.1
8	13.9	6.6	82.3	12.3	6.7	83.6
16	8.8	10.4	65.1	7.1	11.6	72.3

^aWall-clock time determined from the AIX *time* utility.

2. There are penalties associated with manipulating the number of calculational synchronizations.

3. The synchronization requirements for criticality calculations differ from those of noncriticality calculations.

With these issues in mind, two reactor application problems were selected for the performance analysis. These include a three-dimensional pressurized water reactor (PWR) shielding type problem and a PWR assembly criticality problem.

The first problem is a three-dimensional model of the Three Mile Island unit 1 (TMI-1) reactor.⁵ The objective of this problem is to calculate reaction rates at the cavity dosimeter (~350 cm from the core midplane). In an effort to balance the criteria of a realistic calculation and the criteria of a reasonable amount of required computer time, calculations were performed for 1 million particle histories. (The default value of ten synchronizations was used.)

On a single processor, this calculation required 90.16 min of computer time and 91.47 min of wall-clock time. The wall-clock time was measured with the IBM AIX time utility. The parallel speedups and efficiency associated with using 2, 4, 8, and 16 processors are given in Table I. As expected, the efficiency is shown to decrease with the number of processors for this problem. These losses of efficiency can be attributed to the granularity and the parallelizable fraction of the code. Nevertheless, an order of magnitude speedup is demonstrated for 16 processors.

The second problem consists of a criticality calculation for a reflected 1/8 PWR assembly. This problem uses the MCNP lattice capabilities, and its objective is to calculate the value of k_{eff} for the system. In the interest of computational time, the calculation was performed for 50 cycles with 5000 n/cycle.

On a single processor, this calculation required 80.88 min of computer time and 82.15 min of wall-clock time. The parallel speedups and efficiency associated with using 2, 4, 8, and 16 processors are given in Table I.

The decrease in efficiency with increasing number of processors is again observed. However, for criticality calculations, the grain size does not increase with larger number of cycles because the processors must communicate after every cycle. The grain size does, however, increase with greater number of histories per cycle.

CONCLUSIONS

The MCNP code has been modified to make use of the MPI message-passing library and tested to verify proper operation. The implementation of MPI into MCNP was done in a manner such that no current features are effected, and the user interface is not changed.

Parallel performance for two reactor application problems was shown to be similar to that predicted by Amdahl's law with

a parallelizable fraction of ~0.97. Specifically, speedups of ~7 for 8 processors and ~11 for 16 processors were observed. Higher speedups are possible for the shielding problem if the number of histories is increased by orders of magnitude.

The use of parallel computing (even nondedicated) is capable of decreasing the amount of time required for large/complex transport calculations by more than an order of magnitude.

Because MPI uses the IBM high performance switch (HiPS), it is expected to perform better than PVM. However, the latest version of PVM (3.3.10) is built on top of IBM's MPI implementation and thus can also take advantage of the SP2 HiPS. Therefore, the performance advantages of MPI become less clear. Further, it should be noted that since the grain size (number of operations per communication or synchronization) for most Monte Carlo calculations is relatively large, modest differences in message-passing efficiency may not have a noticeable effect on parallel performance. Future work will include the implementation of MCNP with PVM version 3.3.10 on the SP2 and subsequent comparisons of parallel performance with PVM.

1. A. GEIST et al., "PVM 3 User's Guide and Reference Manual," ORNL/TM-12187, Oak Ridge National Lab. (May 1993).
2. J. F. BRIESMEISTER, Ed., "MCNP—A General Monte Carlo N-Particle Transport Code, Version 4A," LA-12625, Los Alamos National Lab. (1993).
3. G. W. MCKINNEY, "Parallel Processing Monte Carlo Radiation Transport Codes," *Proc. 8th Int. Conf. Radiation Shielding*, Arlington, Texas (1994).
4. W. GROPP, E. LUSK, A. SKJELLUM, *Using MPI—Portable Parallel Programming with the Message-Passing Interface*, MIT Press (1994).
5. J. C. WAGNER, A. HAGHIGHAT, B. G. PETROVIC, "Monte Carlo Transport Calculations and Analysis for Reactor Pressure Vessel Neutron Fluence," *Nucl. Technol.*, **114**, 373 (1996).