CONF-960415--11

SAND96-0397C

RECENT DEVELOPMENT FOR THE ITS CODE SYSTEM - PARALLEL PROCESSING AND VISUALIZATION

Wesley C. Fan
C. David Turner
John A. Halbleib, Sr.
and
Ronald P. Kensek
Sandia National Laboratories
P. O. BOX 5800
Albuquerque, NM. 87185
(505) 845-7977

RECEIVED
FEB 2 1996
OSTI

ABSTRACT

A brief overview is given for two software developments related to the ITS code system. These developments provide parallel processing and visualization capabilities and thus allow users to perform ITS calculations more efficiently. Timing results and a graphical example are presented to demonstrate these capabilities.

I. INTRODUCTION

The Integrated Tiger Series (ITS)[1] provides state-of-the-art Monte Carlo solutions of linear, time-independent, coupled electron/photon radiation transport problems with or without the presence of external electric and magnetic fields. It has been widely used in simulator design and analysis, radiation dosimetry, radiation effects studies and medical physics research. Since its inception, the goal of the ITS developers has been to simultaneously maximize physical accuracy and operational efficiency. This is accomplished by employing the most complete physical models describing the production and transport of the electron/photon cascade, the best available cross-section data and sampling distributions, and variance reduction techniques for various difficult applications.[2] In this work, we focus on two major software developments for the ITS code system to further improve the operational efficiency. First, we implement parallel-processing capabilities so that ITS calculations can be performed more rapidly, either on a cluster of workstations or on a massively parallel machine. This reduces the notoriously high computational expense often associated with the Monte Carlo method and allows users to solve extremely complex problems with fast turnaround time. Second, ITS has been coupled with the Ballistic Research Laboratory CAD package (BRL-CAD)[3] to provide a means for interactive geometry modeling and to display particle tracks. These capabilities reduce the effort required to construct very complicated three-dimensional geometries, and enable users to gain useful insight into the physics of the problem.

In the following sections we first discuss the parallel algorithm appropriate for the ITS code system and its implementation. Two techniques that proved effective for load balancing across multiple processors and machines are briefly discussed. Timing results and performance evaluation for selected problems are described. Finally, we briefly discuss the interface software which provides arbitrary geometry modeling capability for ITS using the BRL-CAD system.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

II. PARALLEL PROCESSING

Advances in computer hardware and communication software have made it possible to perform parallel computing for many scientific/engineering applications. Monte Carlo calculations are inherently parallelizable because the individual particle trajectories can be generated independently with minimum need for interprocessor communication. Furthermore, the number of particle histories that can be generated in a given amount of wall-clock time is nearly proportional to the number of processors. This is an important fact because the inherent statistical uncertainty in any Monte Carlo result decreases as the square root of the number of histories increases. For these reasons, researchers have expended considerable effort to take advantage of different parallel architectures for a variety of Monte Carlo radiation transport codes, often with excellent results.[4]

In ITS, the particle histories are divided into "batches" of equal size and the evaluation of the estimated quantities are performed using batch-averaged sample statistics. Since the batchwise evaluation can be performed independently, it provides a natural partition for parallel processing. At present, the parallelized version of ITS is based upon a message-passing model in conjunction with a master/slave paradigm. The basic operations of the parallel code can be summarized as follows: (1) the master process performs the input functions, starts up the slave processes, processes the problem-dependent parameters and sends a copy of parameters to all slaves, (2) the slaves perform the Monte Carlo calculations, i.e., generating particle trajectories and scoring, and (3) the master receives and combines the data tallied by the slaves, and finally outputs the results. With efficient network communication, it is obvious that step (2) will require the majority of the computation time. Since each slave process can carry out these tasks concurrently, this requirement can be reduced almost linearly with the number of processors.

The generation of random-number sequences for large-scale Monte Carlo simulations in a parallel-processing environment poses a non-trivial problem. The random number sequences for each processor must be independent, with good "randomness" property, and with sufficient long period. Here, we adopt the pseudorandom number generator, RANMAR, proposed by Marsagalia and Zaman. Detailed information and implementation can be found in the review article by James.[5] The basic algorithm of RANMAR is a combination of two different sequences, namely, a lagged Fabonacci sequence which must be initialized by an integer, and a simple arithmetic sequence. The final random number is then produced by a subtraction operation. The most exceptional property of this generator is the extreme ease of generating independent disjoint sequences, which can be accomplished by initializing the generator with different integers. Furthermore, RANMAR has been tested for randomness, and has been demonstrated to have very long period. It is noted that this generator is more expensive to compute than the simple multiplicative linear congruential generators since all the operations are carried out in floating-point. However, the effect of this may be insignificant since the computing time spent in the random number generation is almost negligible in comparison to that for particle tracking.

We have developed two updates to ITS 3.0 which will support various parallel-processing environments. These updates are designed in such a way that users can adapt them to construct and tailor the codes for their specific applications. The first update is designed to work on a cluster of UNIX workstations (either homogeneous or heterogeneous), where the communication tasks and process control are handled by the software Parallel Virtual Machine (PVM).[6] The second version is designed to run on a massively parallel computer such as the Intel PARAGON. In this case, the message-passing is handled by the native operating system calls (SUNMOS/NX).[7] It is noted that these updates also provide the basic structure to port ITS to other parallel computing platforms (such as IBM SP2), which can be accomplished by simply replacing the PVM or NX calls with the system specific functions.

II.A. LOAD BALANCING

The goal of load balancing is to enhance performance and achieve the greatest possible speedup of a parallel program. A balanced program can usually keep all processors busy and have them finish roughly at the same time. Otherwise, valuable processor cycles are wasted if some processors have to wait on others to finish. Load balancing is essential in parallel processing ITS since the computing time for each batch may vary, and thus adversely affect the performance. This variation in computing time may result from the following reasons: (1) the stochastic nature of particle histories, (2) difference in computational power on each machine in a heterogeneous configuration, and (3) change in computational performance in a multiuser environment. These considerations must be taken care of by adjusting the way the problem is distributed in a parallel system.

The current version of ITS provides two load-balancing schemes, namely, the static and dynamic methods. The static method is simple and easy to implement. In this method the required tasks (or batches) are divided up and assigned to the available machines or processors. The number of batches can vary from machine to machine to account for different computation power for different machines. These assignments are set at the start and will not be adjusted to the actual loading and performance. As one may expect, this scheme can be quite effective on a dedicated or lightly loaded system (either homogeneous or heterogeneous). Dynamic load-balancing is accomplished by the classic "pool-of-tasks" paradigm. Initially, each slave process is given a batch just as in the static scheme. As a slave process finishes its task it will receive another one. With this scheme all the slave processes are kept busy as long as there are batches remaining in the pool. The work load for each machine is adjusted according to the "realistic" computational performance which can be changing dynamically as other users share the resources. An intuitive way to implement the dynamic scheme is by dividing up the problem into small batches (small number of particles per batch) which may be easier to balance across the available machines than the large batches. However, the number of batches should not be so excessive as to incur extra overhead in communication and output processing.

II.B. PERFORMANCE EVALUATION

The goal of parallel processing is to make the program run faster (shorter wall-clock time) than it would in the corresponding serial run. A speedup ratio is often used to evaluate the performance of a multitasking program. On a dedicated system, the speedup ratio can be calculated in the following manner:[4]

$$S_N = \frac{T_S}{T_N} = \frac{1}{1 - F_P + \frac{F_P}{N}} , \qquad (1)$$

where S_N is the speedup ratio if N processors are used in the calculation, T_S is the elapsed wall-clock time for a single processor, T_N is the wall-clock time for N processors, and F_P is the fraction of the program that can be run in parallel (sometimes called the parallel efficiency). The second part of this equation is known as the Amdahl's law from which one can estimate the parallel efficiency based on a set of measured speedup ratios. It is important to note that the parallel efficiency gives a measure of the extent to which a given program can potentially be parallelized, but it does not account for other effects such as multitasking and communication overhead.

Table 1 summaries the measured speedup ratios for seven test problems on a cluster of SUN workstations. These test problems include the three standard codes (TIGER, CYLTRAN, and ACCEPT), two P-codes, and two M-codes of the ITS system, and utilize many tally and biasing options of the system. Sufficient particle histories were required so that the input/output and communication times were negligible in comparison to the overall CPU times. It is observed that the speedup ratios increase almost linearly with the number of processors. The parallel efficiency approaches 99%, except for the ACCEPT-M code, where it is around 93%. Further studies indicated that the relative poor performance of the ACCEPT-M code was caused by an anomalous batch which consumed ~ 50% more CPU time than the other batches. It is believed that one or more electrons entered a vacuum region with a uniform magnetic field with velocities almost perpendicular to the field so that they drifted very slowly through this region. Consequently, extra computing time was needed to calculate these orbits, thus prolonging the CPU time for that batch.

To further demonstrate the benefit of parallel processing, we consider a timing benchmark problem proposed by Rogers and Bielajew.[8] This problem was originally designed to assess the feasibility of using Monte Carlo method in radiotherapy treatment planning. To make a Monte Carlo code clinically useful on a routine basis, the patient dose calculation should be done in 5-10 minutes. The benchmark problem consists of a $(10\ \mathrm{cm})^2$ plane parallel 20-MeV electron beam incident on a patient. The patient is modeled as a $(19\ \mathrm{cm})^3$ phantom (mainly water) with 1 cm^3 voxels everywhere except in the 1 cm^2 central region, where the voxels are $(2.5\ \mathrm{mm})^2$ by 1 cm thick except at the peak dose region where there are four voxels, each 2.5 mm thick. There total number of voxels is 7624.

For timing assessment, we used the ACCEPT code to determine the dose distribution in all voxels. Simulation of 3.6 millions source electrons produced an uncertainty of 2% at the peak of dose-depth curve. The elapsed computing time is 108 minutes on a SUN Sparc1000 with 8 processors and 25 minutes on an Intel PARAGON using 120 processors. These timing results are very encouraging. Although these values are unacceptable relative to the requirement for routine treatment planning, they are short enough for research and development in treatment methods. Moreover, further reduction in the wall-clock time can be easily achieved by increasing the number of processors and/or by using more powerful processors which are currently available.

III. VISUALIZATION

Visualization capabilities for the ITS code system have been realized with a software tool that is designed to interface with the Ballistic Research Laboratory CAD package (BRL-CAD). Originally developed for military vehicle design, BRL-CAD is a powerful solid modeling system which provides an interactive geometry editor, a ray tracing library, and a large collection of related tools and utilities. Its basic functions allow users to construct a geometry interactively and use ray tracing for model interrogation. Moreover, an image of a geometry model may be displayed with proper three-dimensional perspective from any viewpoint. As an example, Figure 1 shows a BRL-CAD Model of the bremsstrahlung convertor and collimator configuration for the EG&G LINAC.

In BRL-CAD, geometrical models are represented by the Constructive Solid Geometry; that is, a model is constructed by combining a suite of solid primitives through boolean operations. This is essentially the same combinatorial method used in the ITS/ACCEPT codes. For this reason, a simple interface software (GIFTEX) has been developed to extract information from the BRL-CAD geometry database and convert it to the ITS input format for subsequent calculations.

We are currently developing computer software to process particle tracks generated by the ITS calculations and display them together with the problem geometry using BRL-CAD. In this implementation, ITS generates a file containing selected information on the particle history, which includes particle type, energies, weights and positions. A utility is then used to filter this information at the user's discretion and prepare a data file for BRL-CAD to access. The final display will support three-dimensional rotations, translations, zoom features, and provide illustration of particle types and energies by color.

ACKNOWLEDGEMENTS

This work was supported by the U. S. Department of Energy under Contract No. DE-AC04-94AL8500.

REFERENCES

1. J. A. Halbleib et al., "ITS: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes - Version 3.0," *IEEE Trans. Nucl. Sci.*, Vol. 39, No. 4, pp.1025-1030, 1992.

2. J. A. Halbleib et al., "Version 4.0 of ITS Electron/Photon Monte Carlo Transport Codes," *Trans. Am. Nucl. Soc.*, Vol. 75, pp.329-330, 1995.

3. M. J. Muuss et al., "The Ballistic Research Laboratory CAD Package Release 4.0," U.S. Army Ballistic Research Laboratory Technical Report, Volume I-V, 1991.

4. W. R. Martin, "Monte Carlo Methods on Advanced Computer Architectures," *Advances in Nuclear Science and Technology*, Vol. 22, pp.105-164, 1991.

5. F. James, "A Review of Pseudorandom Number Generators," *Computer Physics Communications*, 60, pp.329-344, 1990.

6. G. A. Geist et al., "PVM 3 User's Guide and Reference Manual," Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, 1994.

7. K. S. McCurley, "Intel NX Compatibility Under SUNMOS," Technical Report SAND 93-2618, Sandia National Laboratories, 1994.

8. D. W. O. Rogers and A. F. Bielajew, "Monte Carlo Techniques of Electron and Photon Transport for Radiation Dosimetry," in *The Dosimetry of Ionizing Radiation*, Eds. K. R. Case, B. E. Bjarngard, and F. H. Attix, Academic Press, 1990.

Table 1. **Measured Speedup Ratios* for Various ITS/PVM Applications**

Code	**Number of Processors**				
	2	4	8	12	16
TIGER	1.99	3.81	7.39	10.88	14.19
CYLTRAN	1.97	3.92	7.32	10.93	14.31
ACCEPT	1.99	3.93	7.42	10.64	14.2
TIGER-P	1.97	3.93	7.77	11.5	14.39
ACCEPT-P	1.96	3.88	7.64	11.35	14.03
CYLTRAN-M	1.96	3.87	7.42	11.06	14.56
ACCEPT-M	1.80	3.49	5.83	7.75	10.78
Amdahl's Law with 99% Efficiency	1.98	3.88	7.48	10.81	13.91

* Speedup relative to a single SUN4/75 workstation.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

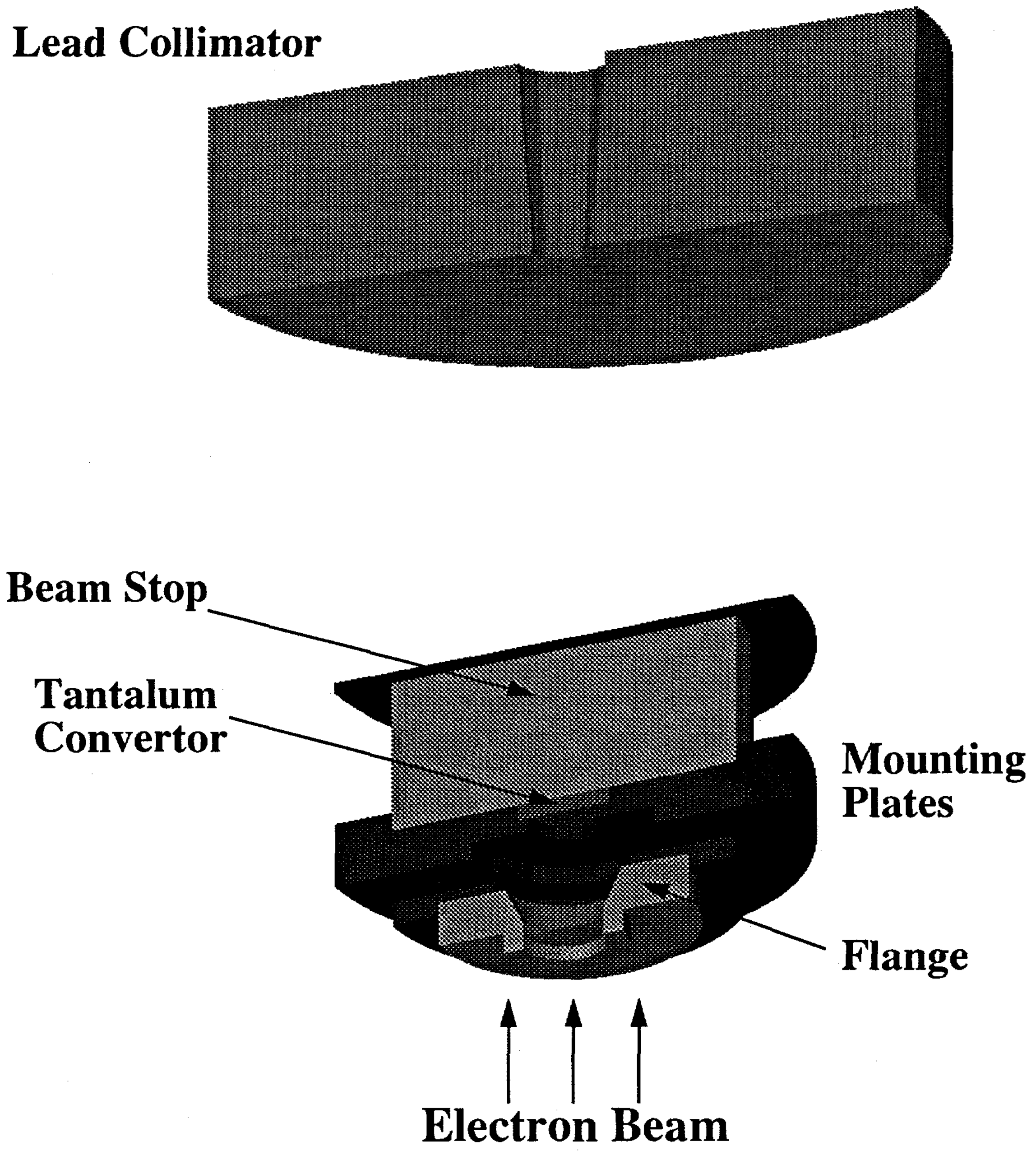

Figure 1. A Cross-Sectional View of the BRL-CAD Model of the EG&G LINAC Bremsstrahlung Convertor and Collimator.