PROCEEDINGS OF SCIENCE

Parallelization of Neutron Transport Code ATES3 on BARC's Parallel System

Kislay Bhatt[1]
Scientific Officer 'F'
Computer Division, BARC, Trombay, Mumbai, INDIA
E-mail: kislay@barc.gov.in

Vibhuti Duggal
Scientific Officer 'D'
Computer Division, BARC, Trombay, Mumbai, INDIA
E-mail: vibhuti@barc.gov.in

Rajesh Kalmady
Scientific Officer 'G'
Computer Division, BARC, Trombay, Mumbai, INDIA
E-mail: rajesh@barc.gov.in

Anurag Gupta
Scientific Officer 'F'
Reactor Projects Design Division, BARC, Trombay, Mumbai, INDIA
*E-mail:*anurag@barc.gov.in

A Neutron Transport Code ATES3 (Anisotropic Transport Equation Solver in 3D), which was developed at BARC for the deterministic solution of 3D steady-state neutron transport problems, has been parallelized using Message Passing Parallel Programming model on BARC's ANUPAM Parallel Supercomputer. The most time consuming step in the code ATES3, which is transport sweep, was made parallel using three different data-decomposition techniques, namely, parallelization in angular variable, parallelization using Diagonal Sweep approach and in the third approach, optimization techniques are being applied in Diagonal Sweeping. In the paper, we discuss the problem domain, parallelization techniques used and present the performance figures obtained with each approach.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research - ACAT 2010
Jaipur, India
February 22–27 2010

[1] Presenter

PoS(ACAT2010)037

1.Introduction

The most fundamental task in the design and analysis of a nuclear reactor core is to find out the neutron distribution as a function of space, direction, energy and possibly time. The most accurate description of the average behavior of neutrons is given by the linear form of Boltzmann transport equation. Due to massive number of unknowns, the solution of the transport equation imposes severe demands on computer processors / memory and requires best of numerical and computational schemes.

The code ATES3[1] (Anisotropic Transport Equation Solver in 3D) had been developed in BARC for the deterministic solution of 3-D steady-state neutron transport problems. The code makes use of advanced Krylov subspace based schemes for the solution. To use ATES3 for practical reactor core simulations, it has been parallelized on BARC's ANUPAM[2] Parallel Supercomputer using Message Passing Parallel Programming model.

The most time consuming step in the ATES3 code is transport sweep, which was targeted for the parallelization. We used three different data-decomposition techniques for parallelization. In first phase, the parallelization was done in angular variable, which yielded very good speed up. In second phase, Diagonal Sweep approach was followed, which couldn't give very good efficiency figures due to idleness of the participating processors. In the third phase, we are trying to apply optimization techniques over Diagonal Sweep approach to improve the efficiency by doing proper load balancing among the participating processors.

In the following sections, we discuss the compute-intensive part of the code, the parallelization techniques used for parallelization of the target portion and present the speed-up and efficiency figures obtained with each parallelization approach.

2.Program description

The general neutral particle transport equation has seven dimensions (3 in space, 2 in directions, 1 each in energy and time). The most commonly used method to discretise the angular variable Ω is the Discrete Ordinate (S_N) method. Similarly other variables are also discretised by appropriate schemes. The discretised transport equation is conventionally solved by the well-known inner-outer iteration procedure. The code ATES3 was developed in-house for the deterministic solution of steady-state neutron transport problems in 3-D Cartesian geometry. One of the remarkable feature of the code is the use of Krylov subspace based schemes for the solution, in addition to the conventional methods. This has resulted in substantial CPU-time reductions and more accuracy. The code is written in Fortran-90/95 with modular format and is user/developer friendly. The most time consuming portion of the code is transport sweep, where iterations are performed one by one for each of the directions of neutron motion under consideration. The iterations for different directions are independent of each other and can be performed in parallel. Considering the inherent parallelism in the code, it was decided to do parallelization on transport sweep. Three different data-decomposition techniques were used for the parallelization, which are discussed in the following sections.

3.Angular Parallelization

The transport sweep part of the code takes 80% of the total execution time, where each of the N angular directions of neutron motion are iterated one by one for calculation of partial angular flux. Since, the calculations of angular fluxes in different directions are independent of each other, therefore, in first phase, the parallelization was done in angular variable, which yielded very good speed up with efficiency of 57% on 288 processors. In this technique of parallelization, the master process reads the input data, which specifies the problem configuration, and then constructs arrays representing the source, angular quadrature, material

composition and cross section data, and then broadcasts all necessary inputs to the slave processes. Then all the processes, master as well as slaves, proceed with mesh sweeps along the angles, statically assigned to them, and accumulate the contribution to the scalar flux in a local array. At the end of all mesh sweeps, all participating processes accumulate their local arrays into the new iterate array. The array accumulation is based on global reduce operations that are implemented in two ways, one using MPI's native global reduce operation and the second one using Bucket Algorithm[3].

MPI's global reduce operation is based on a spanning tree topology among the participating processes. In this scheme, half the participating processes send their partial contributions to the other half, that performs the sum, then half the remaining processes send again, and so on. This is followed by a broadcast stage in which the new iterate is sent to all participating processes along the same spanning tree topology described above. The main advantage of this scheme is that it reduces the number of messages exchanged to a minimum, a crucial benefit on platforms that possess high communication latency. On the other hand, its disadvantages include a substantial idleness, as the number of active processes is cut by half in each step of each of the two stages, and the additional idleness results if the number of processes is not a power of 2.

Bucket Algorithm performs the global reduce operation on a monodirectional ring topology. In this scheme, each of the participating processes start by passing a bucket containing the subvector of its local contribution to the scalar flux to its neighbor along the monodirectional ring. Each process then sums the subvector it just received into the corresponding local subvector which contains its own contribution, then sends the bucket to its neighbor, and so on. By the time each bucket has circumnavigated the entire ring, each process will contain the completely updated iterate for its subvector. In the broadcast stage, each process sends the new iterate subvector to its neighbor again along the monodirectional ring. The bucket algorithm typically results in a larger number of messages than the spanning tree scheme, however, it produces a smaller volume of data traffic that is not concurrent, hence outperforming the MPI's global reduce on platforms with small communication latency. The main advantage of the Bucket Algorithm is that it does not constrain the number of participating processes to powers of 2, thereby all but eliminating idleness if the number of processes divides the number of angles. Its disadvantages include a greater number of arithmetic operations, and of messages exchanged. In addition, it requires implementation by the programmer since it is not available in standard MPI libraries.

The observed timings for transport sweep part of the code for test cases of Light Water Reactors, with above mentioned Angular Parallelization approaches, are presented in the following table. In the table, Ortho, TSA, Power and SI are abbreviations for different iterative methods, where Ortho stands for ORTHOMIN(1), TSA stands for Transport Synthetic Accelerations, Power stands for Power Iteration and SI stands for Source Iteration.

Problem size	Directions	Method	Sequential	Parallel Runs with Angular Parallelization							
nx,ny,nz						MPI_Allreduce			Bucket Algorithm		
			Sweep time in secs	No of Cores	Cores used per Server	Sweep time	Speed Up	Efficiency	Sweep time	Speed Up	Efficiency
100	24	Ortho,TSA	236	24	4	56	4.21	17.54			
100	24	Power,SI		24	4						
100	80	Ortho,TSA	740	80	4	96	7.7	9.63			
100	80	Power,SI	4541	80	4	229	19.82	24.77	228	19.91	24.89
100	80	Power,SI	4541	80	1	121	37.52	46.91			
100	288	Ortho,TSA	3068	288	4	70	43.82	15.22			
100	288	Power,SI	16272	288	1	149	109.2	37.91	138	117.91	40.94
180	288	Power,SI	127312	288	4	1596	79.76	27.69	1350	94.3	32.74
180	288	Power,SI	127312	288	1	816	156.01	54.17	778	163.64	56.81
180	288	Ortho,TSA	28024	288	1	171	163.88	**56.9**	183	153.13	53.17

Plots for two particular cases from the above table are drawn and presented in figure-1, which is placed on the last page of the paper.

4.Diagonal sweep

In the second phase of parallelization, a different approach called Diagonal Sweep[4] was followed. In this technique, the iteration starts from the corner cell, as shown in figure-2(a), and in the successive iterations, the adjacent cells on the successive diagonal planes are solved. Every cell on the sweep or diagonal plane is independent of all other cells on the same plane, therefore, all the cells in a sweep plane can be processed in parallel. The sweeps for different directions are also independent and can be processed in parallel. The number of cells per diagonal plane follows the progression '1, 3, 6, 10, …', which goes up to 3*N*N/4 and then decreases in reverse order. Here, N is the number of meshes in each of the three dimensions in the orthogonal grid. The maximum number of processors required in this scheme is 3*N*N/4, which is equal to the number of cells in the biggest diagonal plane in the grid.

We tried diagonal sweeping in two ways. In the first case, an outer loop was implemented to iterate over the directions of neutron motion and an inner loop was implemented to iterate over the successive diagonal planes. Iteration from one diagonal plane to the successive one requires inter-process communication. In this method, as we increased the number of directions from 24 to 288, the inter-process communication also increased linearly and we could not get any speedup. Then we tried another way, where, in the outer loop we implemented iterations over the diagonal planes and in the inner loop, iterations were implemented over the directions. This way, the inter-process communication reduced drastically, as the intermediate flux calculations for all the directions in a cell were done in one go and stored in an array. This array was communicated to the cells in the successive plane during the next outer iteration. Thus, there was very minute effect of increase in number of directions on communication overheads.

With the currently available computational resources in the organization, we could try a problem of size N=10, considering 288 directions of neutron motion, on 75 processors and achieved the speedup of around 5x over the sequential program. In this scheme, most of the participating processors remain idle for almost half of the time, so, we couldn't get a good efficiency figure with this approach. There was a need of optimizations in this approach for proper load-balancing across the employed processors. We applied one optimization technique on Diagonal Sweep, which is briefed in the following section.

5.Optimized Diagonal Sweep

In the third phase, we applied an optimization technique on Diagonal Sweeping to utilize the participating processors efficiently. For this, the problem domain was decomposed block-wise among the processors using Volumetric decomposition[5] technique, as seen in figure-2(b). In this scheme, all the cells in one cubical block are handled by one processor. On the boundaries, inter-process communication is required to move data from an `upstream' cell on one processor to an adjacent `downstream' cell on another processor, where which cells are upstream and downstream depends on the sweep direction.

Since, we decomposed the grid in 8 blocks, we tried a problem of size N=180 and 288 directions on 8 processors. Due to too much communication overheads and search operations involved in this technique, we could achieve speedup of 2x over the sequential run. Efforts are still going on to optimize the inter-process communication and other overheads.

6.Conclusion and Future work

In the paper, we presented parallelization techniques applied to transport sweep in the ATES3 code. Up till now, the parallelization techniques were tested for reactor simulations involving isotropic scattering. Subsequently, similar techniques will be tested for problems involving anisotropic scattering. The so-called whole-core simulations, which do not involve

approximations such as spatial homogenization of fuel assemblies, are extremely CPU-intensive. It is planned to perform these calculations explicitly with large number of meshes, without the spatial homogenization, on a future larger ANUPAM parallel system at BARC.

References

[1] Anurag Gupta and R.S. Modak, ***ATES3: Anisotropic Transport Equation Solver in 3D***, Proceedings of 16th annual meeting of Indian Nuclear Society (INSAC-05), Mumbai, Nov 15-18, 2005.

[2] Rajesh Kalmady, Vaibhav Kumar, Digamber Sonvane, Kislay Bhatt, B.S. Jagadeesh, R.S. Mundada, A.G. Apte and P.S. Dhekne, ***ANUPAM – Ameya: A Teraflop Class Supercomputer***, the International joint Conferences on Computer, Information and System Sciences and Engineering (CISSE) 2006, held in US.

[3] Y. Y. Azmy, ***Communication Strategies for Angular Domain Decomposition of Transport Calculations on Message Passing Multiprocessors***, Joint International Conference on Mathematical Methods and Supercomputing for Nuclear Applications, Oct. 5-9, 1997, Saratoga Springs, NY, Vol. 1, p. 404 (1997).

[4] R.S. Baker and K.R. Koch, ***An S_n algorithm for the massively parallel CM-200 computer***, Nucl. Sci. Engrg. 128 (1998), pp. 312–320.

[5] Mark M. Mathis, Nancy M. Amato, Marvin Adams, ***A General Performance Model for Parallel Sweeps on Orthogonal Grids for Particle Transport Calculations***, In Proc. ACM Int. Conf. Supercomputing (ICS), pp. 255-263, Santa Fe, NM, May 2000.

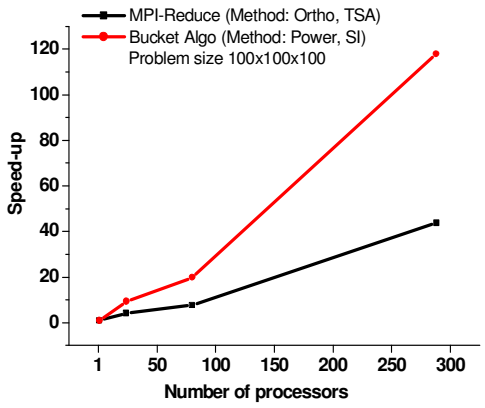

Figure-1 Plots of Angular Parallelization data

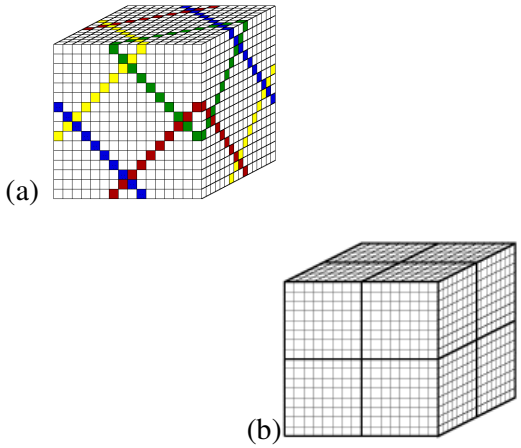

Figure-2 a) Diagonal Sweep
b) Block-wise decomposition of grid