Parallelization of the Pin Resolved Variational Nodal Method

Yongping Wang,[a, d] Tengfei Zhang,[b] E. E. Lewis,[c]
W. S. Yang,[d] M. A. Smith,[e] and Hongchun Wu[a]

[a] *Xi'an Jiaotong University, Xi'an, Shaanxi, China*
[b] *Shanghai Jiao Tong University, Shanghai, China*
[c] *Department of Mechanical Engineering, Northwestern University, Evanston, IL*
[d] *Department of Nuclear Engineering and Radiological Sciences, University of Michigan, Ann Arbor, MI*
[e] *Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL*

yongpinw@umich.edu, zhangtengfei@sjtu.edu.cn, e-lewis@northwestern.edu, wonyang@umich.edu, masmith@anl.gov, hongchun@mail.xjtu.edu.cn

INTRODUCTION

In previous work, a three-dimensional Variational Nodal Method (VNM) had been developed for performing whole-core pin resolved transport calculations and encouraging results were obtained [1]. However, it turned out that applicability with serial calculation is restricted as high order expansions for both angular and spatial variables are required to achieve satisfactory accuracy. Thus, in this summary, hybrid parallelization is applied to the pin resolved VNM to enhance its efficiency. The process of VNM can be divided into two parts: the matrix formation and solution. For the matrix formation, the calculations of node-energy matrix sets are evenly distributed to different MPI [2] processors and no communication between the processors is needed during the calculations. Then OpenMP [3] is further used to parallelize the integrals over angle in matrix formation. For the solution portion, MPI-based domain decomposition [4,5] is performed. Data transfer is performed along the subdomain boundaries during the red-black iterations. In addition, OpenMP is adopted to accelerate the loop of updating the outgoing currents of the nodes within each subdomain. The 3D C5G7 problem [6] is calculated to evaluate the parallel performance.

THEORY

The pin resolved VNM result in the following three multi-group equations [1] for energy group g and node n: the group source, the response matrix and the flux equations.

$$\underline{q}_{gn} = \sum_{g'\neq g} \underline{\underline{\sigma}}_{sgg'n}\underline{\phi}_{g'n} + \frac{1}{k}\chi_g \sum_{g'} \nu\underline{\underline{\sigma}}_{fg'n}\underline{\phi}_{g'n}, \quad (1)$$

$$\underline{j}^{+}_{gn} = \underline{\underline{R}}_{gn}\underline{j}^{-}_{gn} + \underline{\underline{B}}_{gn}\underline{q}_{gn}, \quad (2)$$

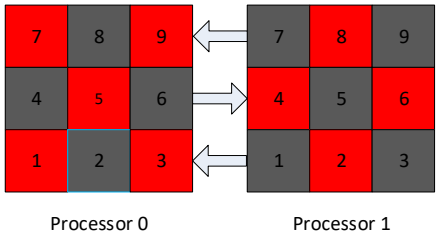
$$\underline{\phi}_{gn} = \underline{\underline{V}}_{gn}\underline{q}_{gn} - \underline{\underline{C}}_{gn}\left(\underline{j}^{+}_{gn} - \underline{j}^{-}_{gn}\right), \quad (3)$$

$\underline{q}_{gn}$ is the group source, $\underline{j}^{\pm}_{gn}$ are the partial current vectors coupling the nodes, and $\underline{\phi}_{gn}$ is the scalar flux. $\underline{\underline{\sigma}}_s$ and $\nu\underline{\underline{\sigma}}_f$ are the diagonal matrices consisting respectively the neutron scattering cross section and production cross section; k and χ are the eigenvalue and fission spectrum. The node-energy matrices sets, $\underline{\underline{R}}_{gn}$, $\underline{\underline{B}}_{gn}$, $\underline{\underline{V}}_{gn}$ and $\underline{\underline{C}}_{gn}$, must be evaluated for each unique node type and are pre-calculated before performing fission source iterations.

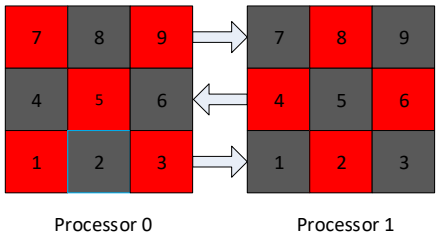
Thus, firstly the calculations of matrix sets are assigned to MPI processors. Assuming there are N node types and G groups, thus $NR=N\times G$ matrix sets to be distributed among P processors. If NR is exactly divisible by P, it is perfect load balance; if not, some of the processors need to calculate one additional matrix set. For example, if there are 5 unique node types (N), 7 energy groups (G) (i.e. 35 matrix sets (NR)) to be calculated with 8 processors (P) available, then 3 processors will be assigned 5 matrix sets while the other 5 processors will be assigned 4 matrix sets. OpenMP is further applied to parallelize the angular integrals in the calculation of matrix sets. The equations of matrices integrated over angle are detailed elsewhere [1]; they are evaluated using a square Legendre-Chebyshev cubature with 25×25 Gauss points over the angular domain. Then the 25×25 tasks can be distributed evenly among available OpenMP treads. As the matrix sets are independent, no communication is needed between MPI processors during the calculation.

For the solution portion, no-overlapping domain decomposition is performed and each subdomain is assigned to a MPI processor. Each processor carries out the calculations of Eq. (1) through (3) of its own subdomain. Data transfer between subdomains is needed to update the outgoing currents of nodes as shown by Eq. (2). Red-black ordering has been applied to the whole problem before domain decomposition, therefore, each subdomain preserves the red-black ordering after domain decomposition. To perform the red-black Gauss-Seidel iteration, the outgoing currents of all red nodes are first

updated in all subdomains, then before updating the outgoing currents for black nodes, data transfer along the subdomains boundaries is carried out: each subdomain sends the surface outgoing currents of red nodes along the subdomain boundaries to the adjacent subdomains so that they have the latest incoming currents for black nodes. After that, the outgoing currents for black nodes are updated. Similar communication is also performed after the update of all black nodes is finished. On each boundary, the two adjacent subdomains send and receive partial currents simultaneously by point-to-point communication as shown in Fig. 1. In addition, a collective communication is adopted to collect the iteration error from all the subdomains at the end of each red-black iteration.



(a) Data transfer after solving the black nodes



(b) Data transfer after solving the red nodes

Fig. 1. Data Transfer Between Non-overlapping Subdomains

Within each subdomain, there is a long loop over nodes for calculating the outgoing currents (Eq. (2)) consecutively, similar to that required in evaluating the angular integrals. Therefore, the nodes are further assigned evenly to different OpenMP threads to accelerate the red-black iterations.

RESULTS AND DISCUSSSION

The parallelization algorithms have been implemented in the PANX [1,7] code. To evaluate its performance, the 3D C5G7 benchmark (unrodded case) [6] is calculated. In radial, each pin cell is treated as a node and the axial domain is divided into 18 nodes. Thus the total number of nodes of the problem is 51×51×18 (in *x*, *y* and *z* directions). The problem domain is then decomposed into 3×3×1, 3×3×2 and 3×3×4 subdomains, respectively, when 9, 18 and 36 MPI processors are used. Therefore, in radial, each subdomain will always have 17×17 nodes. The axial nodes of the subdomains are 18 and 9 when the axial domain is decomposed to 1 and 2 subdomains. When the axial domain is divided into 4 subdomains, the numbers of axial nodes in the subdomains are 5, 5, 4 and 4.

Comparisons of computational effort and parallel efficiency for the problem are shown in Table I. The total computational effort consists of two parts: matrix formation and solution. The numbers without parentheses are the actually obtained results while the numbers in parentheses are the theoretically achievable values.

The theoretically achievable values are calculated by considering the following two factors. First, for the matrix formation, load imbalance occurs if the number of matrix sets is not exactly divisible by the number of MPI processors; OpenMP is only applied to the angular integral part which takes approximately 90% of the effort of the matrix formation. Second, for the solution portion, OpenMP is applied only to the code segment that updates the outgoing current for all nodes which occupies 80% of the computational time of solution portion. As it is difficult to theoretically calculate the efficiency loss caused by the communication effort and the load and communication imbalances for the solution part, they are not taken into account in the theoretical values in Table I.

All of the calculations are run on a cluster that consists of 580 nodes. Each node has two 8-core Intel Xeon-E5 CPUs and 64 GB of memory. The parallel efficiencies shown in the table are calculated by:

$$parallel\_efficiency = \frac{T_s / N}{T_p} \times 100\%, \quad (4)$$

where T_s and T_p are the computational times of serial and parallel calculations, respectively, and N is the total number of threads which equals to the number of MPI processors multiplied by the number of OpenMP threads.

It is found from Table I that the same eigenvalue is obtained for different cases, which demonstrates the correctness of the implementation of the parallelization algorithms. The serial calculation shows that matrix formation dominates the total computational effort. When 9, 18, 36 MPI processors with 1 OpenMP thread is applied, the parallel efficiency of the matrix formation is very close to the theoretical efficiency. However, the efficiency for the solution part decreases with increasing number of MPI processors due to the communication effort, local work and communication imbalances. As the domains become smaller with the increasing number of MPI processors, the ratio of communication effort to local computational effort increases. Therefore, the parallel efficiencies are decreasing with increasing number of processors. Additionally, the local work and communication imbalances also affect the parallel efficiency. Table II shows the maximum/minimum

numbers of red nodes, black nodes and communication boundaries when the problem is assigned to 9, 18 and 36 MPI processors. It is observed that the communication and local work imbalances become severer with increasing number of MPI processors. More detailed analysis of the communication effect are left to the future study as the solution portion only takes about 13% of the total computational effort.

When only 4 OpenMP threads are used, the efficiencies for matrix formation and solution are decreased to 73% and 40%, respectively, which are lower than those with 4 MPI processors. However, this is expected as OpenMP is only applied to limited parts of the algorithm.

The hybrid parallelization efficiency with 36 MPI processors and 4 OpenMP threads is very poor as it suffers the problems of both MPI (load imbalances and communication efforts) and OpenMP (limited code segments are parallelized). However, the significant reduction of wall clock time (from 51.8 hours to 0.78 hour) greatly enhances the code's applicability to either larger spatial domains or higher angle-space approximations.

Table I. Results of 3D C5G7 Problem

Number of MPI processors (P) / OpenMP threads (T)	1 P / 1 T (Serial)	9 P / 1 T	18 P / 1 T	36 P / 1 T	1 P/ 4 T	36 P / 4 T
k_{eff}	1.14276	1.14276	1.14276	1.14276	1.14276	1.14276
Time for matrix formation (h)	45.000	5.680 (5.625)	3.060 (3.214)	1.610 (1.607)	15.230 (12.938)	0.621 (0.462)
Time for solution (h)	6.820	0.980 (0.758)	0.550 (0.379)	0.300 (0.189)	4.210 (2.984)	0.155 (0.083)
Total time (h)	51.820	6.660 (6.383)	3.610 (3.593)	1.910 (1.796)	19.440 (15.922)	0.776 (0.545)
Efficiency for matrix formation (%)	--	88.0 (88.9)	81.7 (77.8)	77.6 (77.8)	73.9 (87.0)	50.3 (67.6)
Efficiency for solution (%)	--	77.3 (100.0)	68.9 (100.0)	63.1 (100.0)	40.5 (57.1)	30.6 (57.1)
Overall efficiency (%)	--	86.5 (90.2)	79.7 (80.1)	75.4 (80.1)	66.6 (81.4)	46.4 (66.0)

Table II. Local work and communication imbalances of 3D C5G7 Problem

Number of MPI processors (P)	9 P	18 P	36 P
Max/Min red nodes	2601/2601	1301/1300	723/578
Max/Min black nodes	2601/2601	1301/1300	723/578
Max/Min communication boundaries	4/2	5/3	6/3

CONCLUSION

To extend the ability of the pin resolved VNM for calculating larger spatial domains with higher angle-space approximations, the hybrid parallelization algorithms have been implemented in which OpenMP is employed within a MPI framework for both the matrix formation and the solution procedure. At present, due to data communications, load imbalances and the partial implementation of the OpenMP parallelization algorithm, the code falls short of ideal scaling. However, the wall clock time is significantly reduced by a factor of 98% for the 3D C5G7 problem with 36 MPI processors and 4 OpenMP threads. Applications of the parallelized code to larger problems and to obtain more accurate results with higher angle-space approximations are on the way.

ACKNOWLEDGMENT

This work is supported by the China Scholarship Council, the National Natural Science Foundation of China (No. 11605130), and the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under contract # DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the government.

REFERENCES

1. T. ZHANG, Y. WANG, E. E. LEWIS, M. A. SMITH, W. S. YANG and H. WU, "A Three-Dimensional Variational Nodal Method for Pin Resolved Neutron Transport Analysis of Pressurized Water Reactors," Nucl. Sci. Eng., 188, 160 (2017).
2. "MPI: A Message-Passing Interface Standard," version 3.1, University of Tennessee, Knoxville, Tennessee, June 4, 2015.
3. OpenMP Architecture Review Board "OpenMP Application Program Interface Version 3.1," http://openmp.org/wp/openmp-specifications/, July, 2011.
4. A. TOSELLI and O. WIDLUND, Domain Decomposition Methods – Algorithms and Theory, Springer Series in Computational Mathematics, 2005.
5. Y. WANG, C. RABITI, G. PALMIOTTI, "Parallelization of the Red-Black Algorithm on Solving the Second-Order PN Transport Equation with the Hybrid Finite Element Method", Trans, Am. Nucl. Soc. 104, 319(2011).
6. M. A. SMITH, E. E. LEWIS, and B. C. NA, "Benchmark on Deterministic Transport Calculations without Spatial Homogenization: A 2-D/3-D MOX Fuel Assembly 3-D Benchmark," NEA/NSC/DOC (2003).
7. T. ZHANG, E. E. LEWIS, M. A. SMITH, W. S. YANG, and H. WU, "A Variational Nodal Approach to 2D/1D Pin Resolved Neutron Transport for Pressurized Water Reactors," Nucl. Sci. Eng., 186, 268 (2017).