Performance Evaluation of 3-D Hybrid Parallel Finite Element Method by MPI/OpenMP

Tadashi YAMAGUCHI*1(Mem.), Yoshihiro KAWASE*1(Mem.), Atsuyoshi NAGASE and Shota ISHIMURA

We developed a 3-D hybrid parallel finite element method with the process-parallelization by MPI and the thread-parallelization by OpenMP. In this method, the domain decomposition method is used in the process-parallelization and the multicolor ordering is used in the thread-parallelization. In this paper, we analyze the performance of the hybrid-parallelization through the cogging torque analysis of an IPM motor. As a result, it is possible to calculate the magnetic field faster using MPI and OpenMP at the same time. Hybrid parallelized computation at 8 nodes × 4 threads is 8.7 time faster than non-parallelized computation. Parallelization with OpenMP using the multicolor ordering was able to efficiently parallelize the forward-backward substitutions of ICCG. However, number of ICCG iterations increases due to the multicolor ordering in the hybrid parallelization.

Keywords: 3-D finite element method, MPI, OpenMP, Hybrid parallelization

(Received: 20 July 2018, Revised: 12 December 2018)

1. Introduction

Recently, many core CPUs have been developed and widely used due to higher performance and lower power consumption. OpenMP is one of the useful method for the many core CPUs. However, the many core CPUs are shared memory type computers, and it is necessary to prevent conflict of memory access between threads. Some parallel computing methods such as multicolor ordering for shared memory type computers, were proposed [1].

We have developed a 3-D parallel finite element method by MPI using the domain decomposition method (DDM), and have reported the performance of parallelization [2,3] in distributed memory type computer. MPI is used in the distributed memory type computer, and the interprocess communication becomes overhead of parallel performance improvement.

We developed a 3-D hybrid parallel finite element method with the process-parallelization by MPI and the thread-parallelization by OpenMP. In this method, the DDM is used in the process-parallelization and the multicolor ordering is used in the thread-parallelization. In this paper, we analyze the performance of the hybrid-parallelization through the cogging torque analysis of an IPM motor.

2. Analysis Method

2.1 Magnetic field analysis using finite element method

The fundamental equation of the magnetic field can be written by the magnetic vector potential $A$ as follows:

$$\nabla \times (\nu \nabla \times A) = \nu_0 \nabla \times M$$

where $\nu$ is the reluctivity, $\nu_0$ is the reluctivity of vacuum, and $M$ is the magnetization of the permanent magnet.

2.2 Hybrid parallelization

Fig. 1 shows an overview of hybrid parallel computing system. In this system, the parallelization by MPI using the DDM is performed between nodes, and the parallelization by OpenMP using the multicolor ordering [1] is performed between cores in each node. In this paper, the hybrid-parallelization by MPI and OpenMP is called Hybrid, and the parallelization of only MPI is called FlatMPI.

Fig. 2 shows the DDM for the FEM with the edge elements. The DDM is a parallel calculation method that divides the analysis region into subdomains. Each subdomain is allocated to each processor for calculation. METIS [4] is used to divide the analysis region. It is necessary to communicate between the subdomains for the parallel calculation, and the overlapping elements [2]
are added in each subdomain in order to reduce the communication. Overlapping elements are composed of boundary edges and external edges. Boundary edges are allocated to each domain by the DDM. External edges are added by an overlapping element, and it is an edge which interpolates an element which is incomplete by the DDM.

In the DDM, the block ICCG method, which is also called localized ICCG method, is used to eliminate communication between processes at the time of the forward-backward substitutions in the ICCG method for solving simultaneous linear equations.

In the thread-parallelization by OpenMP, the forward-backward substitutions, the matrix-vector product, the inner product, the vector update, which are the main calculations of the ICCG, are parallelized. As a parallel calculation method for shared memory type computers, techniques for thread-safe parallel calculation in the forward-backward substitutions are being developed by ordering the matrix [1].

Multicolor ordering is one of the ways to enable the forward-backward substitutions to parallelize thread-safely by sorting the non-zero entries of the matrix and is used for the parallelization by OpenMP of this paper. The method described in [1] was used for coloring of unknown variables in the multicolor ordering. Coloring is done on the lower triangular matrix. Under this condition, a number obtained by adding one to the maximum of the number of non-zero entries in the row of the matrix is used as the number of colors in the multicolor ordering. After coloring, the unknown variables are renumbered according to each color, and the matrix is sorted. When parallelizing by OpenMP, synchronous processing is required for all threads for each color.

Multicolor ordering is briefly explained using the two-dimensional nodal element in Fig. 3. Fig. 3(a)(2) shows the mesh with the multicolor ordering applied to Fig. 3(a)(1). Figs. 3(b)(1) and 3(b)(2) show the matrix of Figs. 3(a)(1) and 3(a)(2), respectively. Number of colors is four. From Fig. 3(b)(2), we can see that the non-zero entries of each color are independent at diagonal parts in the matrix with the multicolor ordering. Therefore, it is possible to parallelize the forward-backward substitutions in each color.

In the hybrid parallelization system, the unknown variables in the $i$-th subdomain divided by the DDM is colored with the multicolor ordering as follows:

1. Coloring the unknown variables except the external edges. Then, the number of color is $N_i$.
2. Coloring the external edges as $N_i+1$-th color.
3. Renumbering the unknown variables in the color order.
4. Sorting non-zero entries.

![Fig. 2 Domain decomposition method for FEM with edge element.](image)

![Fig. 3 Multicolor ordering with node element](image)
3. Analyzed Model and Analysis Condition

3.1 Analyzed model and condition

The performance of the hybrid-parallelization is analyzed through the cogging torque analysis of an IPM motor.

Fig. 4 shows the analyzed model.

Table 1 shows the discretization data of the 3-D finite element mesh with the tetrahedral edge elements.

The parallel performance is confirmed by two kinds of models: coarse mesh and fine mesh.

Table 2 shows the analysis condition. **FlatMPI** is used to analyze 4 ways with the number of nodes 1, 2, 4, 8. **Hybrid** is used to analyze 12 ways with the number of nodes 1, 2, 4 and the number of threads (number of cores) 1, 2, 3, 4.

Fig. 5 shows the B-H curves of the magnetic materials. The analysis is carried out considering the nonlinearity using 35H300 for rotor and stator, and s45c for shaft.

3.2 System configuration

We used a PC cluster, which is consisted of 8 PCs (Intel Xeon E3-1230 v5 : 4 core, memory 16GB) for the calculation. The operating system is CentOS7. Hybrid parallelization in this system is possible up to 32 parallels of 8 node × 4 threads.

4. Calculated Result and Discussion

4.1 Distribution of flux density and cogging torque

Fig. 6 shows the distributions of flux density vectors in the fine mesh. The result calculated by **Hybrid** at 8 node × 4 thread is in good agreement with the result calculated by **FlatMPI** at 8 node.

Fig. 7 shows the cogging torque waveforms in the fine mesh. It can be confirmed that **FlatMPI** and **Hybrid** have the same results as that calculated with non-parallelization.
4.2 Coefficient Matrix

Fig. 8 shows the distribution of non-zero entries in the matrix of the fine mesh. Fig. 8(a) shows the non-zero entries in the matrix of the whole domain. \(N_w\) is a number of unknown variables of the whole domain. We can see that the non-zero entries concentrate diagonally. Fig. 8(b) shows the non-zero entries in the matrix of one subdomain after the whole domain is divided into eight subdomains. \(N_s\) is a number of unknown variables of the subdomain. The non-zero entries of the external edges are added. Figs. 8(c) and 8(d) show the non-zero entries in the matrix with the multicolor ordering applied to Figs. 8(a) colored with 16 colors and 8(b) colored with 24 colors, respectively. The non-zero entries are distributed throughout the matrix by the multicolor ordering. As a result, it takes a long time to calculate the matrix-vector product because of the access to the right-side vectors, which are distributed widely. Therefore, it may be possible to increase memory access time by multicolor ordering.

4.3 Calculation time

Fig. 9 shows the elapsed times of each analysis. Compared with FlatMPI, Hybrid of 2, 4 threads are able to reduce the elapsed times. Regardless of mesh fineness, Hybrid of 8 nodes × 4 threads was the fastest. However, Hybrid of one thread increases the elapsed time compared with that of FlatMPI. This is because that it is caused by sorting the matrix by multicolor ordering.

4.4 Parallel Performance

Fig. 10 shows the parallel performance. The horizontal axis is the calculation resource, which is the product of the number of nodes and the number of threads. The vertical axis is speed up normalized by the elapsed time of non-parallelization. The plots filled black are the result of Hybrid of 1 thread. Parallel effect of the fine mesh is larger than that of the coarse mesh. Hybrid of 8 nodes × 4 threads can calculate 5.2 times in the coarse mesh and 8.7 times in the fine mesh faster than non-parallelization. In the case of Hybrid of 4 threads in the fine mesh, the obtained parallel efficiency is more than number of PCs.
The effect of thread parallelization by OpenMP is up to about 2.5 times when comparing Hybrid of 4 threads with FlatMPI under the condition that the number of nodes are same. The effect of thread parallelization decreases as the number of subdomains increases. This is because that the number of unknown variables at one node decreases as the number of subdomains increases. In other words, the proportion of synchronous costs of thread parallelization within the elapsed time increases.

Hybrid of 2 nodes × 4 threads and FlatMPI of 4 nodes were similar speeds regardless of the number of nodes. In addition, Hybrid of 4 nodes × 4 threads and FlatMPI of 8 nodes are also similar speeds. Therefore, Hybrid is useful for reducing calculation time with fewer subdomains than FlatMPI.

Fig. 11 shows the total number of ICCG iterations. Fig. 11 shows the total number of ICCG iterations. In the case of 1 node, the total number of ICCG iterations is smaller than those in the case of 2 nodes or more. This is because that the DDM is not applied and some non-zero entries of the matrix are not ignored by the block ICCG. In Hybrid, the ICCG iteration converges with similar number of iterations regardless of the number of threads. This is because that the matrix is not changed by the number of threads. The total number of ICCG iterations of Hybrid is approximately 1.15 times that of FlatMPI.

Fig. 12 shows the elapsed time of forward-backward substitutions per ICCG iterations. Hybrid of 8 nodes × 4 threads is 28 times faster than non-parallelization. The calculation of 4 threads by OpenMP is up to 3 times faster than that of 1 thread. Therefore, Hybrid can reduce the calculation time for the forward-backward substitutions when the number of ICCG iteration increases.

Fig. 13 shows the elapsed time of matrix-vector product per ICCG iteration. Hybrid of 8 nodes × 4 threads is 15.6 times faster than non-parallelization. The calculation time of Hybrid of 1 thread is increase compared with FlatMPI. This is because that non-zero entries are distributed in the matrix by the multicolor ordering, and it takes time to access the memory. Hybrid of 4 thread by OpenMP is up to 2 times faster than that of one thread. Therefore, Hybrid cannot little reduce the calculation time for matrix-vector product when the number of ICCG iteration increases.
5. Conclusion

In this paper, we analyzed the performance of Hybrid using the domain decomposition method and the multicolor ordering by MPI and OpenMP. Consequently, it is possible to calculate faster using MPI and OpenMP. Hybrid of 8 nodes × 4 threads is 8.7 time faster than non-parallelization. It is possible that Hybrid reduce calculation time with fewer subdomains than FlatMPI.

It is possible that Hybrid using the domain decomposition method and the multicolor ordering is efficiently parallelize forward-backward substitutions. However, the matrix-vector product is not efficiently parallelized. Therefore, it is necessary to review the ordering method for further speeding up. In large-scale analysis, it takes time to prepare data of each subdomain by the domain decomposition method, Hybrid may be able to reduce pre-processing than FlatMPI, so further study is needed.

References

[1] T. Iwashita and M. Shimasaki, “Algebraic Multicolor Ordering for Parallelized ICCG Solver in Finite-Element Analyses.” IEEE Trans. Magn., Vol. 38, No. 2, pp. 429-432 (2002).

[2] T. Nakano, Y. Kawase, T. Yamaguchi, M. Nakamura, “Parallel Computing of 3-D Eddy Current Analysis with A-phi Method for Rotating Machines.”, IEEE Trans. Magn., Vol. 48, No. 2, pp. 975-978, 2012.

[3] T. Yamaguchi, Y. Kawase, S. Hori and Y. Iwai, “3-D Parallel Finite Element Method with Prismatic Edge Elements for Dynamic Analysis of Electromagnets”, Proc. of 18th Int. Conf. on Electrical Machines and Systems (ICEMS 2015), Pattaya (Thailand), Oct. 2015.

[4] http://glaros.dtc.umn.edu/gkhome/views/metis