

Enhanced Hybrid MPI-OpenMP Parallel Electromagnetic Simulations Based on Low-Rank Compressions

Xiren Wang and Vikram Jandhyala

Applied Computational Engineering Lab,

Paul Allen Center, Box 352500, Department of Electrical Engineering,
University of Washington, Seattle WA 98195
{xrwang, vj}@u.washington.edu

Abstract

Existing and emerging parallel computing clusters have nodes with multiple-core CPUs. The distributed-memory property across nodes and shared-memory property within a node coexist with each other. The hybrid architecture can be well exploited by combining the MPI (message passing interface) and OpenMP libraries. This combination is able to reduce memory usage and communication costs compared with either individual approach. In addition, the proposed hybrid static-dynamic load scheduling can yield excellent load-balancing without introducing extra cost. Careful implementation of OpenMP threads can diminish parallel overhead significantly, and expedite the iterative solver in several ways. Numerical experiments validate the high performance of the presented hybrid approach.

Introduction

Electromagnetic (EM) simulation is now a necessary and significant part in predicting the behavior of complex chips, packages, and boards in microelectronic systems. The method of moments (MoM) is one of the best suited techniques for analyzing these structures, especially in conjunction with iterative solvers such as GMRES [2] and fast matrix-vector product techniques including but not limited to the low-rank QR compression techniques [1,3]. Even with these advances, there is a strong need to harness the power of parallel computing.

Existing computing clusters typically have multiple nodes, each of which is comprised of multi-core CPUs. Therefore both distributed-memory and

shared-memory properties are present in the computing system. Parallel computing in these environments is typically handled by MPI [5] and OpenMP [6], respectively. A hybrid approach involving both the techniques can potentially have the following advantages: (i) Memory saving by preventing multiple copies of data as in MPI-like distributed approaches on shared-memory nodes, (ii) Time saving in data communication across nodes, and (iii) Helping overcome the limit of the maximum allowable number of processes in MPI in some cases.

The focus of this paper is a hybrid parallel algorithm for MPI-OpenMP use on hybrid architectures, applied to EM MoM simulations. An enhanced static-dynamic load balancing technique is presented. A coarse-grained iterative solver is carefully designed to compress parallel overheads. While traditional parallel solvers tend to pay attention only to the matrix-vector products, the presented approach also focuses on parallelizing other serial steps, as necessitated for true scalability and required by Amdahl's Law. The high-efficiency of this algorithm is demonstrated by numerical experiments.

Parallel iterative MoM solver

For many EM problems, the method of moments with iterative solvers is a useful technique. The low-rank nature [3] of far-field interaction matrices may be utilized to accelerate matrix-vector products. In the multi-level QR compression technique [1], the far-field interaction is grouped into many merged interaction lists (MILs). To set up the interaction matrices Q and R needs $O(r(m+n))$ operations, where r is the expected numerical rank, and m/n is

the number of observers /sources. Then the total work amount can be estimated before any matrix is computed, and divided amongst the processors. With the assignment of the average load to each processor, the load balancing is expected. This is a static scheduling, where no synchronization or communication is needed. As shown in Fig.1, the far-fields F_1, F_2, \dots, F_t and their matrices Q and R are distributed to processors P_1 to P_p .

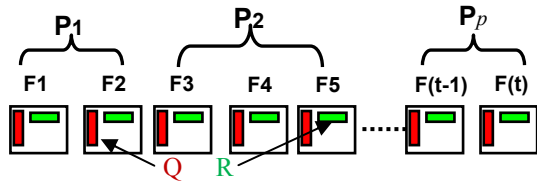


Fig. 1. Far-field matrices distributed statically to processors.

However, this static balancing is rarely obtained in reality. It is difficult to pre-estimate the exact rank, and the setup work load in turn, before matrices Q and R are actually computed. An example of the bad balancing can be found in Fig.3.

Another important factor for high parallel performance is minimum data communication. It has already been considered in the above static scheduling, and will be handled further in the hybrid MPI-OpenMP parallelization discussed herein.

Coarse-grained MPI-OpenMP parallelization

General algorithm

For illustration purpose, it is assumed without loss of generality that a cluster has 3 nodes, each having 4 separate cores. A pure MPI implementation may utilize 12 concurrent processes, and all the processes have to send and receive data. The hybrid MPI-OpenMP counterpart is shown in Fig. 2, where only 3 MPI processes need to transfer data. Obviously, the transfer cost becomes much less because of the reduced reliance on a slower inter-processor bus. Moreover, the computing scalability is maintained by forking each process into four OpenMP threads for intensive computation in matrix setup and system solution.

Additional merits include better load balancing and accelerated GMRES solution procedure, which are the topic of the following sections.

Hybrid static-dynamic load balancing

Usually, data communication across MPI processes is computationally expensive. Although static scheduling helps to avoid the expense, load balancing is difficult to achieve due to difficulties in exact rank-estimation.

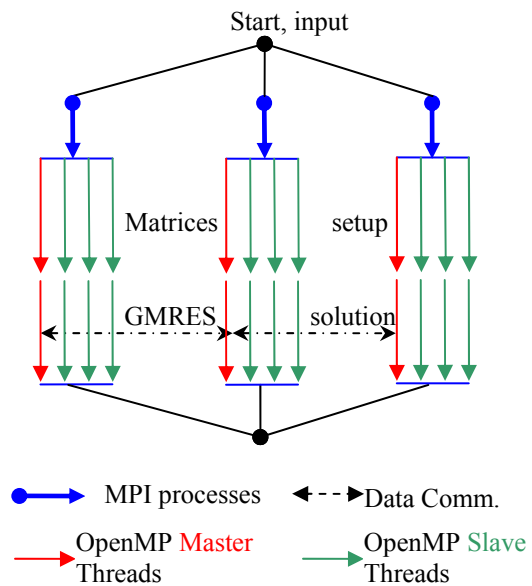


Fig.2. Hybrid MPI-OpenMP parallelization scheme for iterative MoM solver.

This difficulty is alleviated considerably by distributing the errors fairly to all processors. Suppose each processor owns heterogeneous far-fields, the rank estimation errors become impartial to any one. This means a better balancing, provided that each processor receives enough numbers of various kinds of far-fields. In implementation, the far-fields are assigned in a round-robin fashion; when one process estimates that its load is enough, it will reject new assignments.

Fig.3 illustrates the effects. The rank estimation based distribution merely delivers equal amount of expected computation. One process may only own one kind or limited kind of far-fields. If the estimation of this kind of rank has big errors, the expected balancing is impacted greatly. This

explains the obvious imbalance in the figure. In the round-robin scheduling, one process has various different kinds of far-fields, and thus the error risk is evenly distributed, resulting in a much better balancing.

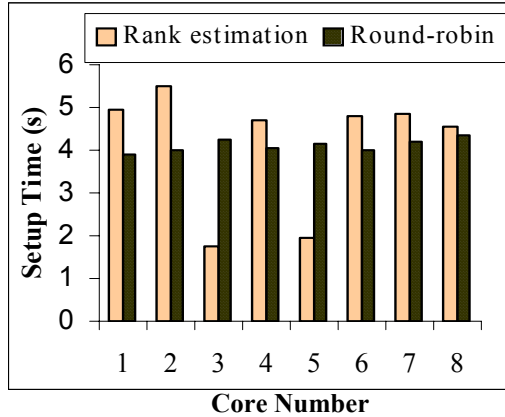


Fig.3. Round-robin static scheduling brings much better balancing for matrix setup.

On the other hand, the above round-robin but still statically-determined balancing may lose its advantage for OpenMP threads, because the number of far-fields becomes very limited for a single multi-core CPU relative to the total problem size. On the other hand, dynamic scheduling is better in this case. One does not have to worry about its extra cost. The synchronization among the threads is much more efficient than among MPI processes running on different multi-core CPUs, due to the difference in bus proximity, architectures and protocols.

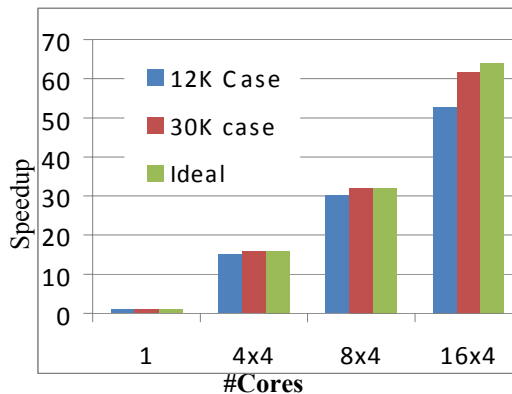


Fig.4. Hybrid static-dynamic load balancing produces almost ideal speedup for matrix setup. “30K” means the number of unknowns in the case is 30, 000.

In summary, static scheduling is efficient for MPI processes, while dynamic scheduling is suited to OpenMP threads. The hybrid scheduling performs well, and an example is depicted in Fig. 4. Clearly, the speedup is very close to ideal for as many as 64 processes. For smaller cases, the less-than-ideal speedup is mainly caused by the inefficiency of the static balancing for a small number of far-field blocks.

Coarse-grained GMRES solver

As specific sections of a parallelized implementation are successfully implemented, the remaining operations become the bottleneck in scaling (Amdahl’s Law). For better efficiency, the iterative solver in Fig.2 is parallelized in a coarse-grained way. In other words, OpenMP threads are forked and joined only once, and remain active throughout the iterative solving procedure. In the fine-grained counterpart, OpenMP threads are created many times, once to get one matrix-vector product. See Fig. 5 for their clear comparison. One potential problem with the latter is that the fork-join overhead may be very large. Fig.6 experimentally demonstrates that the fine-grained version brings almost nothing more

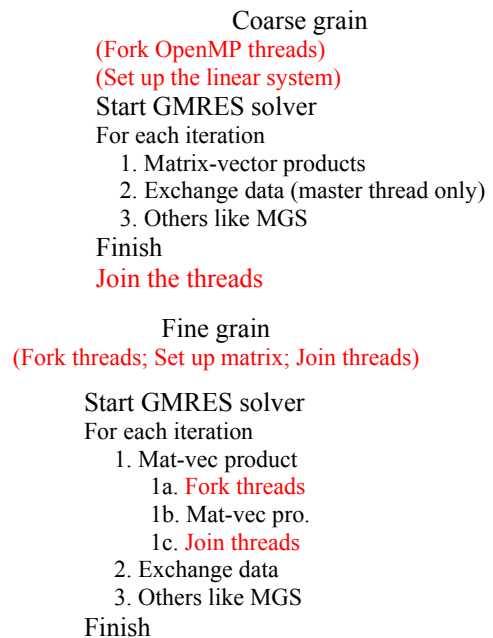


Fig.5. Coarse-grained vs fine-grained iterative solvers. The former saves fork-and-join overheads, and parallelizes MGS operations.

positive than not employing OpenMP (where only 3 threads active in the solution). In contrast, the coarse grain reduces the solution time by more than a half. Reasons lie in the condensed fork-join overheads, and the parallelized modified Gram-Schmidt (MGS) operations within GMRES.

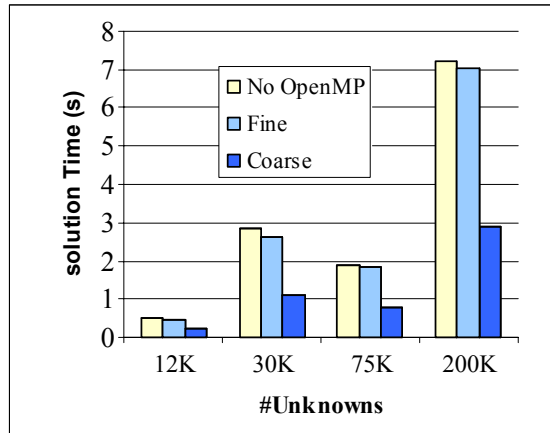


Fig.6 Solution time (4 nodes \times 4 cores/node) with different OpenMP fork-join configurations. The coarse-grained solution saves parallel overhead, and expedites the solution by a factor of 2.

Numerical experiments

Two test cases are taken from a realistic circuit (IBM package, refer to [7]), and their electrostatic capacitance is obtained using the presented parallel iterative solver. Fig.7 shows the scaling of the total time, including both matrix setup time and linear system solution time. As we know, system solution requires inter-node communication, which will necessarily disrupt computation, and degrade the parallel performance. However, the presented hybrid parallel scheme reduces the communication time considerably, and then the speedup here stays close to the ideal. For more cores, the speedup scaling will become better for larger cases, as shown below.

Two parallel conductor planes are examined here, with different number of unknowns. Their scaling is depicted in Fig. 8.

If the solver time is excluded, the speedup for matrix setup is much higher. This can be easily observed in Fig.4.

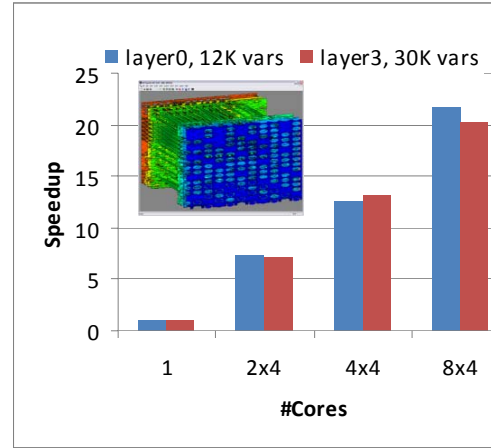


Fig.7 For the whole simulation, including linear system setup and solution, the speedup remains significant.

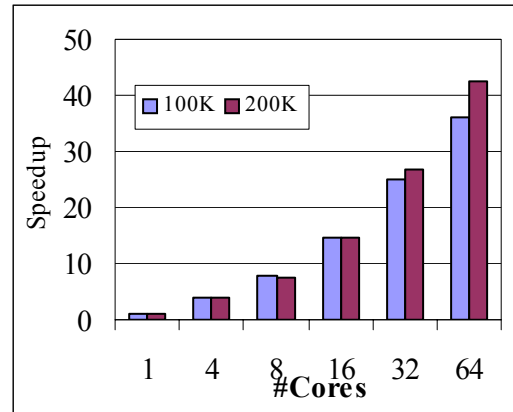


Fig.8. The speedup grows better for cases with more unknowns.

Conclusions

A hybrid MPI-OpenMP parallelism is proposed for low-rank compression based method of moments electromagnetic simulations. Its advantages include memory saving, and communication time cost reduction. In addition, two enhancing techniques are presented. One is the hybrid static-dynamic load balancing for linear equation system setup. Static scheduling is implemented for MPI processes, which doesn't require any data transfer. Dynamic scheduling is used for OpenMP threads, where the efficiency is well retained. The other is the coarse graining of the GMRES iterative solver, where the parallel overhead is condensed, and the MGS operations are also expedited. The hybrid scheme

succeeds in enhancing the parallel performance greatly, which is demonstrated by numerical experiments.

References

- [1] D. Gope and V. Jandhyala, "Oct-tree based multilevel low-rank decomposition algorithm for rapid 3D parasitic extraction," *IEEE Trans on CAD*, vol. 23, no. 11, pp. 1575-1580, 2004.
- [2] Y. Saad, and M. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems." *SIAM J. Sci. Statist. Comput.* vol 7, pp. 856-869, 1986.
- [3] S. Kapur, and D. E. Long, "IES³: efficient electrostatic and electromagnetic simulation," *Computational Science and Engineering*, vol 5, issue 4, pp. 60 – 67, 1998.
- [4] X. Wang, and V. Jandhyala, "Parallel algorithms for fast integral equation based solvers," *Electrical Performance of Electronic Packaging*, pp 249-252, 2007.
- [5] M. Snir, S. Otto, S. Huss-Lederman, D. Walker and J. Dongarra, *MPI: The Complete Reference*, 2nd Edition, The MIT Press, 1998.
- [6] <http://www.openmp.org>.
- [7] V. Jandhyala, C. Yang, S. Chakraborty, I. Chowdhury, J. Pingenot, and D. Williams, "A framework and simulator for parallel fast integral equation simulation of microelectronic structures," *Electrical Performance of Electronic Packaging*, pp. 287 – 290, 2006.