Performance Improvement of Treecode Algorithm for N-Body Problem with Hybrid Parallel Programming

Truong Vinh Truong Duy, Katsuhiro Yamazaki, and Shigeru Oyanagi

Graduate School of Science and Engineering, Ritsumeikan University

Abstract— The hybrid MPI-OpenMP paradigm is an emerging trend to exploit SMP clusters. In this paper, we improve performance of treecode algorithm for n-body simulation by employing the hybrid paradigm. The work load of force calculation which accounts for upwards of 90% of computational cycles is shared among OpenMP threads after ORB domain decomposition among MPI processes. Besides, loop scheduling of OpenMP threads is adopted with appropriate chunk size to provide better load balance. Experimental results demonstrate that the hybrid MPI-OpenMP program outperforms the corresponding pure MPI program by average factors of 1.52 on 4-way cluster and 1.21 on 2-way cluster.

I. Introduction

A combination of shared memory and message passing parallelization paradigms within the same application, hybrid programming, is expected to provide a more efficient parallelization strategy for clusters of SMP nodes. This paper describes the performance improvement of n-body simulation by employing the hybrid MPI-OpenMP programming paradigm on SMP clusters. The n-body is a classical problem, appearing in astrophysics, molecular dynamics, and graphics. In parallelization of the tree code algorithm, the hybrid model exploits two levels of parallelism on an SMP cluster where MPI is used to handle parallelism across nodes and OpenMP is employed to carry out parallelism within a node. Multiple OpenMP threads are used with each occupies one processor of an SMP node to process the calculation of gravitational forces on the bodies in parallel. Besides, loop scheduling with static, dynamic, and guided methods is adopted to ensure sufficient work with better load balance for the threads.

II. The n-body problem

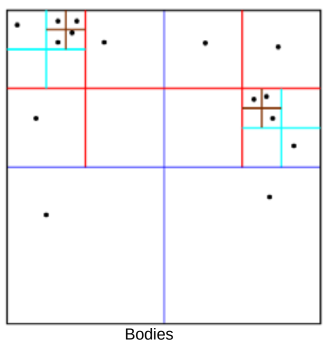
A. Tree Code Algorithm

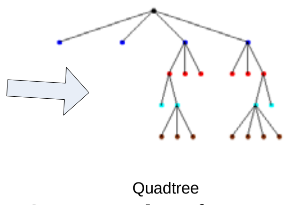
The n-body problem involves advancing the trajectories of n bodies according to their time evolving mutual gravitation field. The essence of the tree code is the recognition that a distant group of bodies can be well-approximated by a single body, located at the center of mass with a mass equal to total mass of the group. It represents the distribution of the bodies in quad-tree for 2D space which is implemented by recursively dividing the 2D space into 4 subspaces, until each subspace contains one body as illustrated in Figure 1. After the tree construction phase, the force on a body in the system can be evaluated by traversing down the tree from root.

B. Parallelization of Tree Code

Since the tree is very unbalanced when the bodies are not uniformly distributed in their bounding box, it is important to divide space into domains with equal work-loads to avoid load imbalance. Therefore, the Orthogonal Recursive Bisection (ORB) domain decomposition is adopted to divide the space into as many non-overlapping subspaces as processors, each of which contains an approximately equal number of bodies, and assign each subspace to a processor.

After the domain decomposition, each process has only the local tree for local bodies. In principle, they need the global tree to determine the forces due to the effect of influence ring along the borders. For example, node n belonging to process 0 has influence on bodies along the borders with P1, P2, and P3 as displayed in Figure 2. Thus node n, as well as other necessary nodes which represent a cluster of bodies, is called essential and must be known by P1, P2, and P3 to compute the forces of bodies in the influence ring of n. Each processor first collects all the nodes in its domain deemed essential to other processors by walking down its local tree from root, and then exchanges these essential nodes directly with the appropriate destination processors. Once all processors have received and inserted the data received into the local tree, each processor can proceed exactly as in the sequential case.





Gravitational force F between two bodies with masses m_a, m_b and distance r is given by:

$$F = \frac{Gm_a m_b}{r^2}$$

Fig. 1. Bodies in 2D space and the quad-tree.

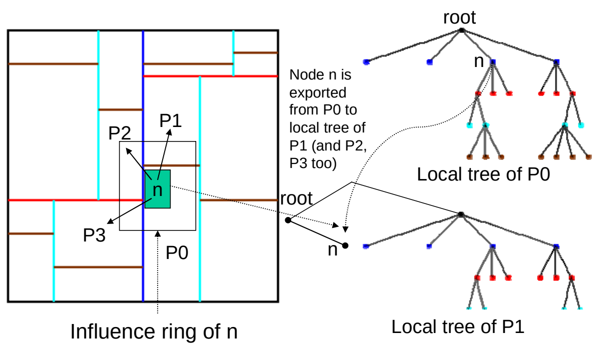


Fig. 2. ORB decomposition and the influence ring of a node.

C. Exploiting Two Levels of Parallelism with Hybrid Parallel Programming Paradigm

Multiple levels of parallelism are achieved with the hybrid program as shown in Figure 3. For the first level, the bodies are distributed in a balanced way among the MPI processes using ORB domain decomposition. After local tree construction, the processes collect and exchange essential nodes to each other to insert into and expand the local trees. Each process then walks through its own tree to create a list of interactive nodes for each body similar to the case of sequential algorithm. For the second level, the force calculation is eminently suitable for parallelizing with OpenMP work-sharing threads running in each MPI process. The bodies and their corresponding list of interactive nodes are assigned to different threads for calculating the force on each body using static/dynamic/guided scheduling with appropriate chunk size. Hence, the hybrid program is expected to speed up the performance.

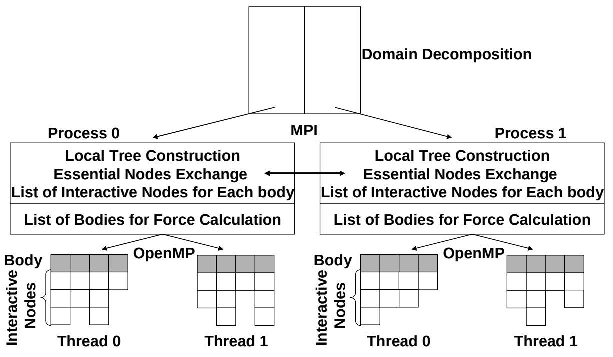


Fig. 3. Multiple levels of parallelism with the hybrid program.

III. Experimental results

Figure 4 displays the execution time of the MPI and hybrid codes simulating the interactions among 10^5 bodies on our 4-way Diplo cluster and 2-way Atlantis cluster, both with Xeon processors and Gigabit Ethernet. In both clusters, no matter how many processors are used, the hybrid implementation outperforms the pure MPI one by average factors of 1.52 on Diplo and 1.21 on Atlantis at all times. We also observe that the factor on Diplo is higher than on Atlantis, resulting from a greater number of OpenMP threads. As a result, it is expected that this factor will be even higher in 8 or 16-way clusters.

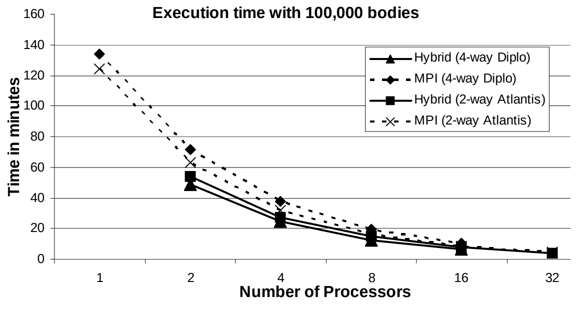


Fig. 4. Execution time on 4-way Diplo and 2-way Atlantis clusters.

In addition, another important advantage of the hybrid model compared to pure MPI model is that it lowers the number of sub-domains in ORB domain decomposition. For instance, we need to create only 4 sub-domains for the hybrid program while 16 sub-domains are necessary for the MPI program on 4-way Diplo cluster. As the number of sub-domains increases, the shapes of domains have a larger range of aspect ratios forcing tree walks to proceed to deeper levels. The complexity involved in determining essential nodes also rises with the number of sub-domains. We found that the number of interactions grows with the number of sub-domains because of these effects. Thus, the hybrid model helps reduce this interaction overhead.

We also tested performance of the hybrid program with a combination of different chunk sizes and scheduling methods, including static, dynamic, and guided on 4-way cluster using 8 and 16 CPUs. The execution time obtained by running the hybrid code is listed in Table I. Schedule dynamic outweighs schedule static and guided in most cases, and the chunk size has an important impact on the performance. This means that choosing a chunk size is a trade-off between the quality of load balancing and the synchronization and computation costs.

TABLE I
EXECUTION TIME WITH DIFFERENT SCHEDULES AND CHUNK SIZES (SECONDS)

	CHUNK SIZE	100	500	1000	2000	5000
8 CPUs	Static	755.7	754.8	757	752.3	906.8
	Dynamic	754.3	751.8	756.2	752.1	906.8
	Guided	758	765.5	756.3	767.7	908.8
16 CPUs	Static	389.9	387.8	385.7	409.8	447.4
	Dynamic	388.2	386.3	384.1	408.1	445.3
	Guided	387.1	388.7	390.6	407.9	445.6

IV. Conclusion

This paper described performance and programming efforts for n-body problem under pure MPI and hybrid MPI-OpenMP programming models. With the hybrid model, after ORB domain decomposition among MPI processes, the work load of time-consuming routines for calculating forces of the bodies is shared among OpenMP threads with loop scheduling. Given these abilities, the hybrid program outperforms the pure MPI programs by average factors of 1.52 on 4-way cluster and 1.21 on 2-way cluster.

REFERENCES

[1] K. Yamazaki, K. Ikegami, and S. Oyanagi, “Speed Improvement of MPEG-2 Encoding using Hybrid Parallel Programming”, IPSJ and IEICE, FIT 2006, Information Technology Letters, LC-005, Vol.5, 2006.
[2] J. Dubinski, “A parallel treecode”, New Astronomy 1 (1996) 133-147.
[3] Treecode, http://ifa.hawaii.edu/~barnes/treecode/