Parallel Out-of-Core MLFMA on Distributed-Memory Computer Architectures

Mert Hidayetoğlu and Levent Gürel
ABAKUS Computing Technologies, Cyberpark, Bilkent, Ankara, TR-06800, Turkey
lgurel@gmail.com

Abstract—We introduce a parallel implementation of the out-of-core method for the multilevel fast multipole algorithm. This implementation utilizes disk space for storing large data structures, and it provides solutions of large-scale problems involving more than one billion unknowns within 2 TB memory.

I. INTRODUCTION

The multilevel fast multipole algorithm (MLFMA) solves a system of equations formulated with the method of moments [1], i.e., \( Z \cdot x = v \), iteratively with \( O(N \log N) \) computational complexity [2], where \( N \) is the number of unknowns. MLFMA partitions a matrix-vector multiplication (MVM) as \( Z \cdot x_i = Z_{NF} \cdot x_i + Z_{FF} \cdot x_i \), where \( Z_{NF} \) is the near-field and \( Z_{FF} \) is the far-field interaction matrices, and \( x_i \) is the evolving guess for the unknown vector in the \( i \)th iteration. \( Z_{NF} \) has \( O(N) \) non-zero elements and is stored in the memory, whereas \( Z_{FF} \) is calculated on-the-fly for each MVM in the iterative solution via aggregation, translation, and disaggregation steps in a multilevel tree structure. Even though \( Z_{FF} \) is not stored in the memory at all, MLFMA stores the far-field patterns of the basis (and testing) functions, which requires \( O(N) \) memory, in order to use them in the iterative solution.

For solving large-scale problems, MLFMA is parallelized on distributed-memory computer architectures [3]. We use the hierarchical partitioning strategy, which achieves good load-balancing performance by partitioning both clusters and field samples (as opposed to partitioning only one of them) in all levels [4]. Despite the excellent performance of the hierarchical partitioning strategy, the memory requirements for storing the near-field interactions and the far-field patterns grow immensely when extremely large geometries are involved. Hence, the amount of total memory available in a computer constitutes a limitation for solving large problems. As a remedy, we propose the out-of-core method on MLFMA, which uses the large data from disk [5], and this paper discusses its parallel implementation on distributed-memory computer architectures.

MLFMA involves pre-processing, setup, and iterative-solution stages. Inputting and processing the geometry data and the tree structure are performed in the pre-processing stage. Near-field interactions, far-field patterns, and the translation operators among the branches of the tree structure are calculated and stored in the setup stage. Finally, the iterative solution is performed.

II. PARALLEL OUT-OF-CORE METHOD

The out-of-core MLFMA stores the near-field interactions and the far-field patterns not in the core memory, but instead, in the disk. Specifically, parallel processes write the out-of-core data on disk(s) during the setup stage, then read them back as needed from the disk(s) during each MVM [5]. The I/O operations are performed on-the-fly through small buffers and the space required for storing the whole portion of the out-of-core data is never allocated in memory.

Hierarchical partitioning strategy distributes the basis functions among processes in a load-balanced manner. Hence, each process has its own portion of basis functions and their corresponding near-field interactions. In the parallel implementation of the out-of-core MLFMA, the processes use the out-of-core data simultaneously from disk and independent from each other without requiring any communications. When multiple-node computer architectures are involved, each process uses the disk space on the node it belongs to, as depicted in Fig. 1.

Fig. 1. As an example, the out-of-core MLFMA is depicted on a two-node computer cluster with 8 processes. In this case, each disk drive handles I/O jobs of four processes simultaneously. A process \( P_i \) names its own file \( F_i \), uniquely in order to access the file when needed. The arrows indicate I/O operations.

Once the processes store the out-of-core data on disks, they read the data multiple times during the iterative solution. In each MVM, the near-field interactions are read once for the near-field multiplication, i.e., \( Z_{NF} \cdot x_i \), and the far-field patterns are read twice for the far-field multiplication, i.e., \( Z_{FF} \cdot x_i \), once for the aggregation and the second time for the disaggregation steps.

III. NUMERICAL RESULTS

To demonstrate the performance of the out-of-core implementation, two scattering problems are solved, involving a conducting sphere and a conducting NASA Almond. The sphere has a diameter of \( 1000 \lambda \) and the NASA Almond has a length of \( 2104 \lambda \), where \( \lambda \) is the wavelength of the illuminating plane wave in free space. The geometries are discretized with
0.1\,\lambda\,\text{mesh size, yielding approximately 1.1 billion unknowns for both problems. The combined-field integral equation with a combination factor of 0.5 is used to formulate the problems. The Bi-CGSTAB iterative solver is employed with a block-diagonal preconditioner [2] for the iterative solution to satisfy 1\% residual error. 64 processes are employed on a 16-node cluster with a total memory of 2 TB. Each node is equipped with a single solid-state disk for storing the out-of-core data.}

**TABLE I**

**CPU Times and Data Amounts for the Solutions**

<table>
<thead>
<tr>
<th></th>
<th>CPU Time (Hours)</th>
<th>Data Amount (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total I/O</td>
<td>Total Memory Disk</td>
</tr>
<tr>
<td>NASA Almond</td>
<td>60.8</td>
<td>10.7</td>
</tr>
<tr>
<td>Sphere</td>
<td>71.3</td>
<td>10.7</td>
</tr>
</tbody>
</table>

Table I shows the CPU times along with the required memory and disk spaces for the solutions. The total data amount indicates the required memory space for solving the same problem in core memory, i.e., without using the out-of-core implementation. The out-of-core implementation decreases the required memory by 52\% and 53\%, but the I/O operations increase the overall CPU time by 12.0 and 10.7 hours, for the NASA Almond and sphere solutions, respectively. In Table I, the total solution times include the I/O times. With the same computer cluster, the largest sphere we can solve in core memory has a radius of 340\,\lambda and the problem involves approximately 540 million unknowns. That evidence shows that the out-of-core implementation is successful for decreasing the required memory space and providing solutions of extremely large problems.

Figure 2 shows the radar cross section (RCS) of the NASA Almond. The solution involves 1,126,503,936 unknowns. Involving two MVMs in each iteration, the iterative solution requires 39 iterations. The MVMs are performed through a 15-level tree structure, where the lowest three levels are “distributed” and the highest five levels are “shared” in the hierarchical partitioning scheme [4].

Figure 3 shows the RCS of the sphere. We observe that the computational values agree well with the analytical Mie-series solution. The computational RCS errors are 3.75\%, 2.80\%, and 0.94\% in the 0–30°, 0–90°, and 0–180° bistatic angle sectors, respectively. The error calculation method is provided in [6], where interested readers can compare their results with corresponding full-wave solutions of large-scale scattering problems including those presented in this paper.

**IV. CONCLUSIONS**

The out-of-core implementation of the parallel MLFMA is presented. Scattering results of a sphere and a NASA Almond with more than one billion unknowns are provided and the solution performance is demonstrated. Results show that the out-of-core method decreases the required memory for solutions and enables large-scale solutions in a limited memory, whereas it increases the required time by a reasonable amount due to the relatively slow disk operations.

**ACKNOWLEDGMENT**

This work was supported by Schlumberger-Doll Research (SDR). The authors would like to thank Intel Corporation and Jamie Wilcox for a generous allocation of parallel computer time.

**REFERENCES**


