A Generic Framework for the Parallel MLFMA

Bart Michiels [1], Jan Fostier [1], Ignace Bogaert [1] and Daniël De Zutter [1]

[1] Department of Information Technology
Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
bart.michiels@intec.ugent.be, jan.fostier@intec.ugent.be,
ignace.bogaert@intec.ugent.be, daniel.dezutter@intec.ugent.be

Abstract: Several data partitioning schemes have been proposed for the parallel multilevel fast multipole algorithm (MLFMA). These partitioning schemes greatly affect the required communications between subtasks (processes) and hence the parallel performance. Even though it is fairly straightforward to establish the asymptotic computational, memory and communication complexities of the different parallelization schemes, it is difficult to make an unbiased performance assessment of actual implementations. Indeed, different authors use different parallel hardware on which their benchmarks were performed and variability in implementation quality or programming language has a significant effect on the overall performance. We present a generic MLFMA framework in which it is straightforward to realize several well-known data partitioning schemes (spatial, hybrid and hierarchical partitioning), alleviating the need for separate implementations for each scheme. This framework is used as a test platform in which the different partitioning schemes can be assessed.

Keywords: Electromagnetic scattering, MLFMA, parallelization

1. Introduction

Advances in computational power nowadays rely on the incorporation of an increasing number of cores in a CPU along with the assembly of increasingly powerful clusters of networked computers. This shift in paradigm towards more parallelism poses serious issues for the development of scientific codes. Especially for the Methods of Moments and the associated Multilevel Fast Multipole Algorithm (MLFMA), arguably the most successful algorithm for large-scale electromagnetic simulations, the development of an efficient parallel algorithm is both challenging and time-consuming. There are two major reasons for this. First, the algorithm consists of different computational phases (i.e., levels in the tree). Each level has distinct characteristics. Hence, data partitioning schemes suitable for the lower levels are inefficient for the higher levels (and vice versa). Therefore, different parallelization schemes need to be adopted at each level. Second, even though the calculations per level can be performed in parallel, strong data dependencies exist, giving rise to high inter-process communication volumes.

Over the past decade, several partitioning schemes have been proposed in literature. In previous work, we have assessed the asymptotic behavior (for large problems sizes and a high number of parallel cores) of these schemes by deriving the computational, communication and memory complexities per parallel process. Schemes that have a better asymptotic behavior will surely outperform other schemes for extremely large simulations and a very high number of parallel processes. However, such schemes are also more involved in terms of algorithmic design and implementation. In order to benchmark these schemes for realistic problem sizes and a reasonable number of parallel processes, implementations of comparable quality are required for each of the partitioning schemes.

In order to avoid the need for new implementations ("from scratch") for each new scheme, we developed a parallel MLFMA framework in which the different data partitioning schemes can be implemented in a modular way. The framework has a generic knowledge of the concepts of the MLFMA, including the tree topology, near interactions, aggregation, translation and disaggregation, but makes no a priori assumptions about the particular integral equation, the discretization scheme, how radiation patterns are sampled and how these sample points are distributed over the different computational nodes. In the next section, this framework will be described, followed by a short overview of the data partitioning schemes. Finally, some numerical details are provided.

2. Parallel MLFMA framework

A graphical representation of the parallel MLFMA framework is presented in Fig. 1. During the setup stage, the geometry of the scatterer is fed to the framework in order to construct the MLFMA tree. Note that the framework makes no assumption about the specific basis and/or test functions that are used (e.g., RWG). Rather, a discretization element is represented by a single point in space to specify the box to which it belongs in the MLFMA octree.

Next, the framework is provided with certain parameters which describe the radiation patterns. The following information is provided per level in the MLFMA tree: (a) the number of partitions and (b) the number of sampling points per partition. The number of partitions describes between how many processes a single radiation pattern is divided on that level. This number can range from 1 (spatial partitioning) to P (full k-space partitioning, with P = number of processes). The framework makes no assumptions about how the points are sampled (e.g. uniform sampling) nor does it require any knowledge about the physical layout of the radiation pattern partitions on the sphere.

Subsequently, based on the tree topology and the partitioning of the radiation patterns, the workload is divided between the processes in an even way. The framework assumes that per level there is no significant variation in the amount work to be processed per sampling point. In case a deviation from this default approach is warranted, user-defined load balancing rules can be provided to the framework. At the end of this phase, the different boxes in the tree and the radiation pattern partitions are attributed to specific processes.

The dependencies between the boxes (and their radiation patterns) are known to the framework by means of the geometry of the MLFMA tree and the general rules of the fast multipole algorithm. For instance, in order to calculate the radiation pattern for a certain box on a certain level, the radiation patterns of the children of that box are required. Similarly, two boxes interact in the translation phase when they are sufficiently separated from each other, while their respective parent boxes are not. However, because the framework has no knowledge about the physical layout of the partitions of a radiation pattern, additional information is required about their dependencies. Specifically, for each partition on a given level *l*, the framework is provided with a list of partitions on level *l-1* on which its calculation (i.e., interpolation) depends. These dependencies, along with the topology of the tree, contain sufficient information for the framework to handle all calculations in the right order, and in order to handle the inter-process communications.

To perform a parallel matrix-vector multiplication during the iterative solving stage, the parallel framework will call user-defined routines that perform near interactions, (dis)aggregation at the lowest level and radiation pattern shifting, inter- and anterpolation and translation. These user-defined routines are 'black-box' routines to the framework that transform an "input" to a certain "output" [1]. For example, a routine that performs a translation will transform (a portion of) an outgoing radiation pattern into (a portion of) an incoming radiation pattern. The 'input' is either data that has been locally computed or received from another node. Similarly, the 'output' will enable the computation of other routines that depend on this intermediate result, or will enable the transfer of the data to other processes that depend on it. To keep the communication overhead low, several 'outputs' can be aggregated in a single message.

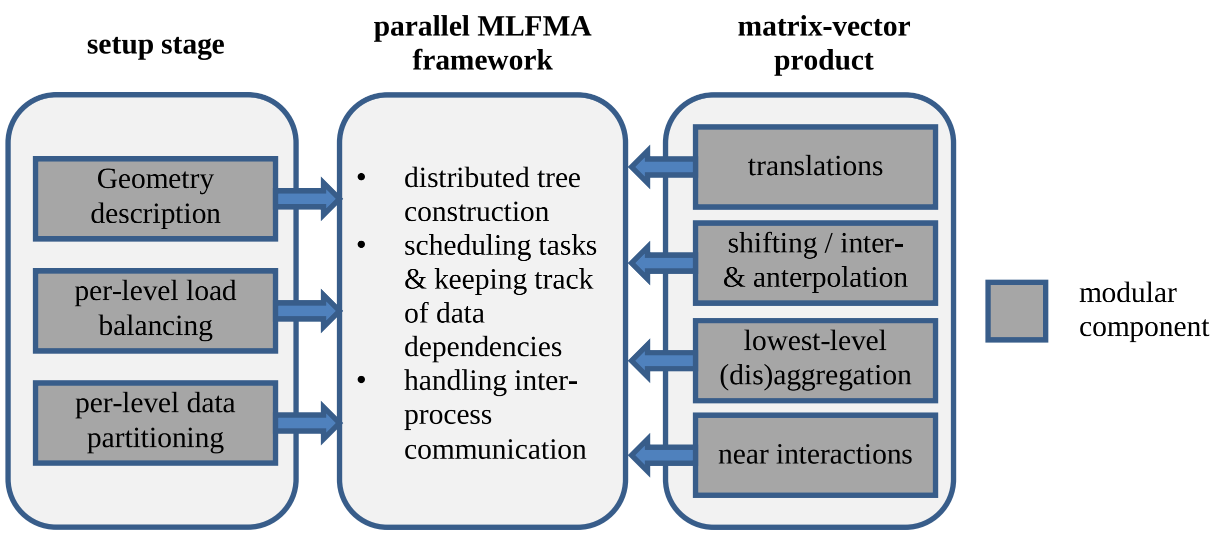
setup stage
parallel MLFMA framework
matrix-vector product
Geometry description
per-level load balancing
per-level data partitioning
distributed tree construction
scheduling tasks & keeping track of data dependencies
handling inter-process communication
translations
shifting / inter- & anterpolation
lowest-level (dis)aggregation
near interactions
modular component

Fig.1: Schematic layout of the modular parallel MLFMA framework.

The modularity of the framework enables the use of different kinds of numerical techniques. For example, we have both modules for global inter- and anterpolation based on Fast Fourier Transforms, and modules for local interpolation based on band-limited interpolation functions (BLIF). Similarly, different versions of the near interaction modules can achieve a tradeoff between speed and memory use, by exploiting the symmetry relations in near interactions.

3. Data partitioning schemes

The following data partitioning schemes are implemented using the parallel framework: spatial, hybrid and two variants of hierarchical partitioning (see below). In previous work, the time, memory and computational complexities per process have been established for the asymptotic case of proportionally increasing problem sizes and number of parallel processes [2], [3]. Table 1 provides an overview of these complexities. Certain schemes give rise to a strong load imbalance between the processes (i.e., some processes have more work than others). Therefore, Table 1 contains the complexities for those processes that have the most work, as they determine the overall runtime. Also, the complexities in Table 1 are provided per level. There are O(log N) levels in total.

In spatial partitioning, there is only a single radiation pattern partition on each level. In other words, the different boxes in the tree and their associated radiation patterns as a whole are distributed among the different processes. Because for higher levels, the radiation patterns increase in size while the number of boxes decreases, severe bottlenecks emerge using the scheme.

The hybrid scheme [4] was proposed to alleviate this bottleneck. It combines the spatial partitioning on the lower levels with the use of P radiation pattern partitions on the higher levels (called k-space partitioning). In this scheme, the transition from spatial to k-space partitioning is the main bottleneck.

In [5], a hierarchical scheme was proposed that provides for a smooth transition between spatial partitioning (at the lowest level(s)) and k-space partitioning (at the highest levels). For every next level during the transition stage, the number of partitions is increased by a factor of four. An important aspect is the physical layout of the radiation pattern partitions in this scheme. In [5], a partitioning of the radiation pattern samples in a single direction of the sphere was adopted, e.g., only in azimuth (strip-wise partitioning of the radiation patterns). Careful analysis learns that in that case, the hierarchical scheme does not improve the asymptotic behavior, compared to the hybrid scheme (see Table 1). By adopting a two dimensional partitioning in the hierarchical scheme, an asymptotic optimal partition scheme is obtained [2].

Table 1: The maximum time, memory and communication complexities per level and per parallel process for the different partitioning schemes. (Note that N = number of unknowns).

Partitioning scheme	Max. time complexity per process and per level	Max. memory complexity per process and per level	Max. communication complexity per process and per level
Spatial	O(N)	O(N)	O(N)
Hybrid	O(√N)	O(√N)	O(√N)
Hierarchical (strip)	O(√N)	O(√N)	O(√N)
Hierarchical (block)	O(1)	O(1)	O(1)

4. Numerical example

The parallel framework is implemented in C/C++ and the inter-process communication is handled using the Message Passing Interface (MPI). As geometry, we consider a perfectly electrically conducting (PEC) cube that is illuminated by an incoming plane wave. The cube has a size of 200 λ, the minimal box size of the MLFMA tree is 0.5 λ (10 levels), a one-box buffer limit is used for the far interactions. 1024 CPU cores are used in the simulation. Only the hierarchical scheme in combination with a block-wise partitioning of the radiation patterns results in a truly uniform load balancing. The communication requirements for the spatial, hybrid and the hierarchical scheme (using strip-wise partitioning of the radiation patterns) are respectively 87%, 38% and 31% higher than for the block-wise hierarchical scheme. At the time of conference, extensive numerical details will be provided.

5. Conclusions

We present a parallel MLFMA framework in which different data partitioning schemes can easily be implemented. This framework serves as test platform to experiment with and validate different partitioning schemes. We have compared the spatial, hybrid and two variants of the hierarchical partitioning scheme. Numerical experiments show that the asymptotic behavior can already be observed for modest problem sizes and a realistic number of CPUs.

6. Acknowledgments

The work of Bart Michiels was supported by a grant from the “Special Research Funds” from Ghent University. Ignace Bogaert was supported by a grant from the “Fund for Scientific Research-Flanders”.

References

[1] J. Fostier and F. Olyslager, “An Asynchronous Parallel MLFMA for Scattering at Multiple Dielectric Objects,” *IEEE Trans. Antennas and Propagation*, 56 (8), pp. 2346-2355, Aug. 2008.

[2] B. Michiels, J. Fostier, I. Bogaert, and D. De Zutter, “Weak Scalability Analysis of the Parallel MLFMA”, submitted to the *IEEE Trans. Antennas and Propagation*.

[3] B. Michiels, J. Fostier, I. Bogaert, and D. De Zutter, “Towards a scalable parallel MLFMA in three dimensions”, in proceedings of the “*Computational Electromagnetics international workshop (CEM)*”, pp. 132-135, Izmir, Turkey, 2011.

[4] S. Velamparambil and W.C. Chew. “Analysis and performance of a distributed memory fast multipole algorithm,” *IEEE Trans. Antennas and Propagation*, 53(8):2719-2727, 2005.

[5] Ö. Ergül and L. Gürel, "A hierarchical partitioning strategy for an efficient parallelization of the multilevel fast multipole algorithm," *IEEE Trans. Antennas and Propagation.*, 57 (6), pp. 1740–1750, June 2009.