# Towards a Scalable Parallel MLFMA in Three Dimensions

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

[1] *Department of Information Technology (INTEC), Ghent, B-9000, Belgium*
jan.fostier@intec.ugent.be

***Abstract*—The development of a scalable parallel Multilevel Fast Multipole Algorithm (MLFMA) for three dimensional electromagnetic scattering problems is reported. In the context of this work, the term 'scalable' denotes the ability to handle larger simulations with a proportional increase in the number of parallel processes (CPU cores), without loss of parallel efficiency. The workload is divided among the different processes according to the hierarchical partitioning scheme. Crucial to ensure the scalability of the algorithm, is that the radiation patterns – sampled on the sphere– are partitioned in two dimensions, i.e. both in azimuth and elevation directions.**

## I. Introduction

In the past two decades, the Multilevel Fast Multipole Algorithm (MLFMA) has been the focus of intense research in the electromagnetics community as a means to accelerate the matrix-vector product in the iterative Method of Moments (MoM) solution of large scattering problems expressed as boundary integral equations. Indeed, whereas the classical evaluation of the dense matrix-vector product requires $O(N^2)$ calculations, the MLFMA reduces this complexity to only $O(N \log N)$, where $N$ denotes the number of unknowns. In practice, this means that simulations up to one million of unknowns are feasible on a single workstation. Many problems of realistic size however, are discretized into an even larger number of unknowns (tens or even hundreds of millions of unknowns). To allow such simulations, the MLFMA at hand needs to be parallelized. This means that the algorithm needs to be adopted in such a way that it can take advantage of a distributed memory environment, i.e. a cluster of networked computers.

Especially in the past decade, several parallel implementations of the MLFMA have been proposed by various authors. The discriminating factor between the different approaches is the way the workload in the MLFMA is divided among the processes. The goal is to maximize both the problem size that can be handled and the *concurrency*, i.e. the ability to perform certain computations simultaneously. The concurrency is directly related to the parallel speedup, i.e. how many times the parallel algorithm is faster than its sequential counterpart. Because of the many data dependencies that exist within the MLFMA and the rather large communication flows between the processes, this speedup is always lower than the number of processes used, even for the best implementations. As a matter of fact, it is exactly this communication overhead that makes the parallelization of the MLFMA difficult. The degree of concurrency is often expressed as the parallel *efficiency*, the ratio of the speedup to the number of processes used. Several parallel implementations have been reported that can handle hundreds of millions of unknowns using over hundreds of parallel processes, with efficiencies exceeding 70%.

In this contribution, we focus on a second important aspect of parallel algorithms: the development of a *scalable* parallel MLFMA. The term 'scalable' is often (mis)used to indicate the fact that for a problem of certain fixed size, an implementation exhibits a 'good' parallel efficiency using a 'large' number of parallel processes, where the exact interpretation of 'good' and 'large' varies largely between authors. Whether or not the same efficiency can be attained for larger problems and/or larger parallel systems is usually ignored. In this manuscript, we adopt the following definition for scalability: an algorithm is said to be scalable if the number of parallel processes $P$ *can* increase proportionally with the number of unknowns $N$ (i.e. $P = O(N)$) without loss of parallel efficiency. Note that this definition by no means implies that the number of processes *should* increase linearly with the number of unknowns, let alone that the number of processes is defined by the number of unknowns in a fixed way. Rather, scalability denotes that a problem twice as big can be handled with twice the number of processes, with the same parallel efficiency as the original problem. Because the complexity of the sequential MLFMA is $O(N \log N)$, the ability to have $P = O(N)$ processes implies that the memory, computational and communication complexity for each individual process should not exceed $O(\log N)$.

The scalability of the MLFMA is closely related to the partitioning of the workload, as will be further elaborated upon. For the two-dimensional MLFMA, a scalable implementation has been presented in [1]. In this contribution, we extend this work to three dimensions. To the best of the authors' knowledge, no truly scalable parallel MLFMA in three dimensions has been presented up to date.

The development of scalable parallel algorithms is rapidly gaining attention, not only from a theoretic point of view, but also and more importantly because CPU vendors are embracing the multi-core paradigm as a way to advance computational power. Indeed, the speed of a single core has increased only moderately in the past decade and a more powerful computing environment is assembled by incorporating more CPU cores in a system and/or by building large clusters of networked computers. It is likely that this trend will continue in the next years.

This paper is organized as follows: in Section II, we will revise the data structures of the MLFMA. Next, in Section III, a number of partitioning schemes (so-called spatial, $k$-space and hybrid) will be discussed and will be demonstrated that they do not lead to a scalable algorithm. In Section IV, we will revise the hierarchical partitioning scheme and show that it does yield a scalable algorithm provided that a two-dimensional partitioning of the radiation pattern sample points is used. Finally, in Section V, we present such implementation and prove its scalability.

## II. MLFMA Data Structures

For an introduction to the MLFMA, we refer to [2]. In this Section, we restrict ourselves to an outline of the data structures in the MLFMA.

Consider a three dimensional high-frequency boundary integral equation problem (i.e. a problem with a size much larger than the wavelength $\lambda$) for which the geometry under consideration has been discretized into triangular elements of a size of approximately $\lambda/10$. Over these triangles, $N$ unknowns are defined using the standard schemes (e.g. RWGs [3]). The MLFMA recursively subdivides the geometry in an octree of boxes until a box size of approximately $\lambda/2$ is obtained at the lowest level. Only non-empty boxes are retained. At the lowest level in the tree, each box contains $O(1)$ unknowns, hence, there are $O(N)$ boxes. For every next level, the number of boxes decreases roughly by a factor of four. The top level consists of a single box. Hence, the tree consists of $O(\log N)$ levels in total.

Each box in the tree contains a number (the exact number depends on which integral equation formulation is used) of so-called radiation patterns, sampled on the sphere. This radiation patterns is a far field description of the part of the geometry that is enclosed in the box. The number of sampling points required to represent the radiation pattern in an accurate way depends of the electrical size of the box. At the lowest level, the number of sampling points in each of the $O(N)$ boxes is $O(1)$ (i.e. independent of the number of unknowns $N$). For every next level, this number of sampling points increases roughly by a factor of four. Because the number of boxes decreases by the same factor, the number of sampling points at every level remains roughly the same. At the top level, there is a single box that contains $O(N)$ sampling points. Hence, each of the $O(\log N)$ levels contains $O(N)$ sampling points in total. In the MLFMA, the number of calculations to perform per sampling point is constant, hence the total computational and memory complexity of the MLFMA is $O(N \log N)$.

## III. Partitioning Schemes

In this Section, we review a number of partitioning schemes for the distribution of the data structures in the MLFMA among different parallel processes and investigate their scalability. We remind the reader that the scalability implies the ability to have $P = O(N)$ processes and that each individual process should not exhibit a memory, computational or communication complexity that exceeds $O(\log N)$. The

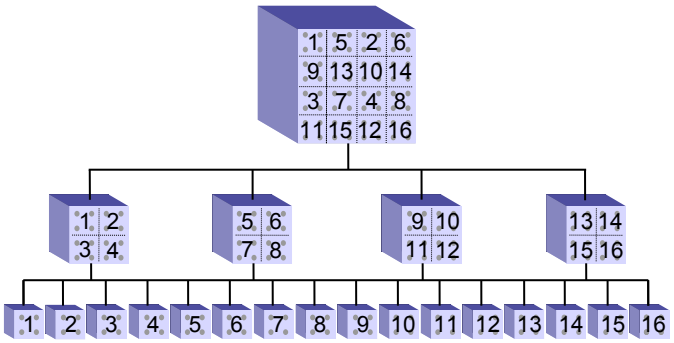


Fig. 1. Hierarchical $k$-space partitioning scheme for an MLFMA tree with 3 tree levels and 16 processes. The cubes represent boxes in the MLFMA tree, the dots in the cubes represent radiation pattern sampling points. The numbers represent the different processes (0-15). At the lowest level, spatial partitioning is used. At the next level, boxes are shared by four processes each holding one fourth of the sampling points. At the top level, full $k$-space partitioning is obtained.

following partitioning schemes all have in common that they, one way or another, distribute the sampling points among the different processes. If a certain sampling point is local in the memory of a certain process, all computations with respect to that particular sampling point will be handled by that process. To perform these calculations, data local in the memory of another process might be required and will have to be communicated first. As a first remark, note that it is useless to partition *between* the different levels in the MLFMA tree. This is because the radiation patterns on a certain level are recursively calculated from the radiation patterns on the underlying levels. Therefore, no concurrency would be achieved with such scheme. In order to achieve concurrency, the sampling points *on each level* need to be distributed among the processes, in the most uniform possible way.

*Spatial partitioning*: in spatial partitioning, the boxes and their associated radiation patterns as a whole are partitioned among the participating processes at each level. Clearly, this scheme is not scalable. Stated simply: to partition boxes among $P = O(N)$ nodes in a balanced way, there have to be $O(N)$ boxes. This is only the case at a constant number of lowest levels. At the top levels, certain processes are attributed a box with $O(N)$ radiation pattern sampling points. Hence, the memory and computational complexity for that node will also be $O(N)$, exceeding the $O(\log N)$ constraint. Spatial partitioning however, has been used frequently in early parallelization attempts. The reason for that is that (a) the scheme is fairly easy to implement and (b) for a smaller (and fixed) number of processes (e.g. 16 processes), a reasonably balanced workload division can be achieved with this scheme. For a larger number of processes, however, the workload will become unevenly divided, a problem that get worse if the number of processes increases further, no matter how large the number of unknowns. Therefore, spatial partitioning allows for only $P = O(1)$ processes. In order to minimize

communication flows between processes, parent and child boxes are attributed to the same process as much as possible. For the translation phase however, significant communications flows are required when the two interacting radiation patterns are stored in the memory of different processes.

*k-space partitioning*: $k$-space partitioning is the dual variant of spatial partitioning. Instead of distributing the boxes among all processes, the sampling points *within* each box are distributed among all process. Similarly, in order to partition sampling points among $P = O(N)$ processes in a balanced way, one needs $O(N)$ radiation patterns. Such radiation patterns can only be found at a constant number of top levels. For the lower levels, $k$-space partitioning can not provide for a balanced load division. For the same reason as for spatial partitioning, $k$-space partitioning allows for only $P = O(1)$ processes. Note that even though $k$-space partitioning is asymptotically is as good (or as bad) as spatial partitioning, no pure $k$-space implementations have been reported, to the best of our knowledge. Note that for $k$-space partitioning, the translation phase is completely communication-free. The aggregation and disaggregation phase however, do require substantial communications.

*Hybrid partitioning*: Hybrid partitioning [4], [5] combines the two approaches mentioned above: spatial partitioning is used for the lowest levels, whereas $k$-space partitioning is used for the higher levels. The optimal transition level is the middle level in the tree. This level consist of $O(\sqrt{N})$ boxes each containing $O(\sqrt{N})$ sampling points. It is easy to show that the computational, memory and communication complexity per node is then also $O(\sqrt{N})$. Hence, hybrid partitioning allows for $P = O(\sqrt{N})$ processes, a clear improvement over spatial and $k$-space partitioning.

*Hierarchical partitioning*: The hierarchical partitioning scheme, first introduced in [6], is the next logical step to assign workload to the different processes in a more balanced way. This is achieved by adapting the partitioning strategy dynamically at each level. At the lowest levels, spatial partitioning is used. At the next level, a box is shared among four processes, however, each process now holds only one fourth of the sampling points of the radiation pattern within that box. At the next level, the sampling points are partitioned among 16 processes, and so on, until full $k$-space partitioning is obtained at the top levels (see Figure 1). Note that we assume for simplicity that $P$ is a power of four. In the next Section, we will build upon the idea of hierarchical $k$-space partitioning and address the question *how* the radiation patterns should be partitioned among the different processes. We will then show that, given and appropriate partitioning of the radiation patterns, this scheme does lead to a scalable algorithm.

## IV. A Scalable Partitioning Scheme

In the previous Section, it was explained that the radiation patterns are gradually partitioned in an increasing number of $1, 4, 16, \ldots, P$ partitions. In this Section, we investigate *how* this partitioning should be achieved.

As top level radiation patterns contain $O(N)$ sampling points, there are $O(\sqrt{N})$ sampling points in the azimuthal direction ($\phi$) and $O(\sqrt{N})$ sampling points in elevation ($\theta$). In the first reports of the hierarchical scheme [6], [7], the radiation patterns are partitioned in a 'strip-wise' fashion by distributing the values in one direction only, e.g. the azimuthal direction (see Fig. 2 left). Clearly, it is not sustainable to partition $O(\sqrt{N})$ values in the $\phi$ direction among $O(N)$ processes. Therefore, a strip-wise approach allows for only $O(\sqrt{N})$ processes. A different way to look at this is by considering the communications required to perform local interpolations (usually by using Lagrange or BLIF interpolation) on radiation patterns. In order to perform these interpolations accurately, certain sampling points near the boundaries of adjacent partitions need to be exchanged between processes. In the strip-wise scheme, every partition has only two neighbors. The amount of communications required however, is proportional to the length of the boundary, in this case $O(\sqrt{N})$. Therefore, the communication complexity of the strip-wise scheme is $O(\sqrt{N})$. Even though this is asymptotically the same complexity as the hybrid scheme, the scheme does reduce the communications, and allows for a higher parallel efficiency [7].

In this work, we present an implementation of the hierarchical scheme that is a based on a partitioning of the radiation patterns in both azimuth and elevation [8] (see Fig. 2 right). For simplicity, we make use of the uniform sampling method introduced in [9]. This allows for a two-dimensional cartesian grid layout of the sampling points that is can be readily partitioned. Given a number of processes $P = O(N)$, it is clear that this scheme allocates $O(1)$ sampling points to each process at each level. This means that the memory complexity for each node does not exceed $O(\log N)$ in total. Because the calculation time is proportional to the number of sampling points, the computational complexity for each individual node is also $O(\log N)$. Because of the two-dimensional layout, the communications required for the inter- and anterpolation stage become more complex to implement. Indeed, each partition has in general eight neighboring partitions instead of just two for a strip-wise layout. Even though the number of communication events will increase because of this, the amount of communications remains $O(1)$. It is easily demonstrated that the communication complexity for the translation phase, near interactions, and repartitioning also does not exceed $O(1)$. This will be experimentally confirmed in the next Section.

## V. Results

The hierarchical scheme and the two dimensional partitioning of the radiation patterns has been implemented in C/C++. The communications between the processes are handled using the Message Passing Interface (MPI). As a computer infrastructure, we use a cluster of 128 machines with two quad-core Intel Xeon CPUs each (1024 CPUs in total). As interconnection network, an Infiniband network is used.

We will experimentally demonstrate that the communication complexity of the proposed scheme is indeed $O(1)$ per level.

TABLE I
THE MAXIMUM AMOUNT OF COMMUNICATION (INCOMING + OUTGOING) FOR A SINGLE NODE, PER LEVEL USING DIFFERENT PARTITIONING SCHEMES (SPATIAL, HYBRID AND HIERARCHICAL) FOR AN INCREASING NUMBER OF PROCESSES AND PROBLEM SIZE.

| #processes P | #MLFMA levels | #unknowns N | Max. communication per node and per level (MByte) | | |
|---|---|---|---|---|---|
| | | | Spatial partitioning | Hybrid partitioning | Hierarchical partitioning |
| 4 | 4 | 18 432 | 4.82 | 3.00 | 4.74 |
| 16 | 5 | 73 728 | 14.36 | 7.21 | 9.83 |
| 64 | 6 | 294 912 | 27.11 | 15.54 | 11.65 |
| 256 | 7 | 1 179 648 | 48.12 | 31.81 | 11.52 |
| 1024 | 8 | 4 718 592 | 114.01 | 75.03 | 11.40 |
| 4096 | 9 | 18 874 368 | 312.70 | 197.53 | 11.30 |

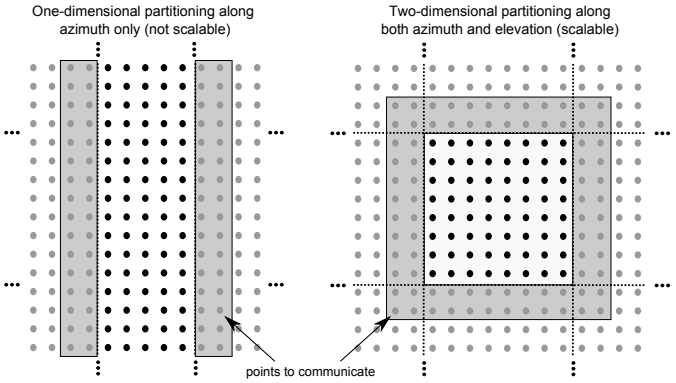


Fig. 2. One dimensional partitioning (along e.g. the azimuth) of the radiation pattern sampling points versus a two-dimensional partitioning.

To accomplish this, we consider an increasingly larger problem that is handled with an increasingly larger number of processes. As a starting points, a perfectly electrically conducting (PEC) cube discretized into $18,432$ unknowns and $4$ parallel processes are taken. Each step, both the number of unknowns and processes are increased by a factor of four. This adds one extra level to the MLFMA tree. For each process, the total amount of (incoming and outgoing) communication is measured and the process that has the largest amount of communication is considered. Table I shows this communication for the spatial, hybrid and hierarchical partitioning scheme with a two dimensional partitioning of radiation patterns. Note that the amount of communication was scaled by $1/\log P$. Also note that we did not implement the hierarchical scheme with the 'strip-wise' layout. From Table I, one can clearly see that the amount of communication increases rapidly when using the spatial and hybrid scheme. For the hierarchical scheme however, the amount of communication per process and per level is constant. Therefore, the communication complexity of the scheme is also $O(\log N)$. As a final remark, note that the result with 4096 CPUs was obtained using only 1024 physical cores. Hence, each core was oversubscribed with 4 processes.

## VI. CONCLUSION

We have presented a scalable implementation of the MLFMA for three dimensional scattering problems. A hierarchical scheme, augmented with a two-dimensional partitioning of the radiation pattern sampling points was used. This scheme leads to a parallelization where the computational, memory and communication complexity of each individual process does not exceed $O(\log N)$. We believe that such scheme paves the way for extremely large-scale simulations using hundreds if not thousands of parallel processes.

## ACKNOWLEDGEMENT

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by Ghent University, the Hercules Foundation and the Flemish Government department EWI. Bart Michiels was supported by a grant from Ghent University (BOF). Ignace Bogaert acknowledges the Research Foundation Flanders (FWO) for a post-doctoral grant.

## REFERENCES

[1] J. Fostier and F. Olyslager, "Provably scalable parallel multilevel fast multipole algorithm," *IET Electronics Letters*, vol. 44, no. 19, pp. 1111–1113, Sep. 2008.

[2] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*. Boston: Artech House, 2001.

[3] S. Rao, D. Wilton, and A. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Trans. Antennas Propag.*, vol. 30, no. 3, pp. 409–418, 1982.

[4] S. Velamparambil and W. C. Chew, "Analysis and performance of a distributed memory multilevel fast multipole algorithm," *IEEE Trans. Antennas Propag.*, vol. 53, no. 8, pp. 2719–2727, Aug. 2005.

[5] S. Velamparambil, W. C. Chew, and J. Song, "10 million unknowns: is it that big?" *IEEE Antennas Propag. Mag.*, vol. 45, no. 2, pp. 43–58, Apr. 2003.

[6] Ö. Ergül and L. Gürel, "Hierarchical parallelisation strategy for multilevel fast multipole algorithm in computational electromagnetics," *Electronics Letters*, vol. 44, no. 1, pp. 3–5, Jan. 2008.

[7] ——, "A hierarchical partitioning strategy for an efficient parallelization of the multilevel fast multipole algorithm," *IEEE Trans. Antennas Propag.*, vol. 57, no. 6, pp. 1740–1750, Jun. 2009.

[8] B. Michiels, J. Fostier, J. Peeters, I. Bogaert, and D. De Zutter, "Provably scalable parallel multilevel fast multipole algorithm in three dimensions," *Submitted to IET Electronics Letters*, 2011.

[9] J. Sarvas, "Performing interpolation and anterpolation entirely by fast Fourier transform in the 3-D multilevel fast multipole algorithm," *SIAM J. Numer. Anal.*, vol. 41, no. 6, pp. 2180–2196, 2003.