Full-wave Scattering from Reflector Antennas on Electrically Large Platforms using low-Memory Computers

Oscar Borries, Peter Demeyer, and Erik Jørgensen
TICRA, Copenhagen, Denmark
{ob,pd,ej}@ticra.com

***Abstract*—We consider the use of full-wave integral equation techniques on scattering problems involving electrically large structures, and consider how an implementation of such techniques could use an inexpensive solid-state drive (SSD) rather than costly random access memory (RAM). We begin by showing how a Multi-level Fast Multipole Method (MLFMM) code based on Higher-Order (HO) basis functions has fundamental properties that make it feasible to use disk storage for the less frequently used algorithm data. Then, we show how the use of the SSD allows us to solve larger problems than the RAM of the computing platform makes room for. Finally we consider how this implementation has only a modest impact on the computational time, particularly when compared to the reduction in financial cost of SSD storage rather than RAM.**

***Index Terms*—Integral Equations, MLFMM, Out-of-Core, Higher-Order**

I. Introduction

Design of antennas for many modern applications, particularly as the pursuit of higher frequencies continues across a wide range of applications, often requires detailed analysis of the electromagnetic scattering. The scattering can be caused by the passive parts of the antenna itself, but the most challenging scattering problems are often encountered when taking into account the surroundings of the antenna, in particular when considering antenna placement, which can be electrically extremely large.

Analysis of such scattering problems using limited computer resources is a challenging problem, despite the impressive progress in recent years in the area of computational electromagnetics. Even with the Multi-level Fast Multipole Method (MLFMM) [1], which reduces the complexity from $\mathcal{O}(f^4)$ to $\mathcal{O}(f^2 \log f)$, where f is the frequency, modern compters will still run out of memory fairly quickly. Indeed, although the asymptotic complexity of an MLFMM implementation is low, the required memory and time for antenna placement or platform scattering problems can be high, particularly when accurate solutions are required for geometrically complex structures at high frequencies. While this has led to efficient algorithms for algorithms designed for a specific type of antenna, there are many applications where a general-purpose MLFMM implementation is the only option.

MLFMM is by now a mature technology, although there is still active research on improving various parts of the textbook implementation. This includes the application of advanced computing platforms for MLFMM, such as distributed computing [2] or Graphics Processing Units [3]. Regardless, when using a laptop or even a desktop, the limited memory availability means that the implementation should focus on reducing the maximum memory footprint.

In this paper, we reduce the maximum memory by storing MLFMM data on other media, a process called an Out-of-Core (OoC) solution, contrary to an in-core solution where only RAM is used for the data. After an initial overview of MLFMM in Section II, we consider the cost and speed of an SSD mounted via the NVMe interface. We then provide several examples in Section IV that demonstrate that the total solution time is not too heavily impacted by the use of an OoC implementation, while the reduction in RAM use is substantial.

II. Multilevel Fast Multipole Method

MLFMM achieves reduced complexity by grouping the N basis functions in an integral equation discretization [4] hierarchically, using the Octree algorithm [5], and letting larger and larger groups interact over greater and greater distances. The grouping is based on the center of the geometric elements of the mesh, and the smallest allowed groups have sidelengths not smaller than the largest geometric element in the mesh.

This splitting allows performing the matrix-vector product as

$$\overline{\overline{Z}}\,\overline{I} = \overline{\overline{Z}}_{\text{near}}\overline{I} + \mathcal{F}(\overline{I}) \tag{1}$$

where $\overline{\overline{Z}}_{\text{near}}$ is the near-matrix, containing the interactions between basis functions that are too closely spaced to apply MLFMM — the elements in this matrix are computed as in the normal MoM approach. $\mathcal{F}$ denotes the operation perfomed by applying MLFMM.

The interaction between two well-seperated basis functions $\boldsymbol{f}_j, \boldsymbol{f}_i$, belonging to groups m and m' respectively, can be computed by

$$\overline{\overline{Z}}_{j,i} = \kappa \oiint \boldsymbol{R}_{jm}(k,\hat{\boldsymbol{k}}) \cdot \Big(T_L(k,\hat{\boldsymbol{k}},\boldsymbol{r}_{mm'})\boldsymbol{V}_{im'}(k,\hat{\boldsymbol{k}})\Big)\, d^2\hat{\boldsymbol{k}}, \tag{2}$$

with $\boldsymbol{r}_{mm'} = \boldsymbol{r}_m - \boldsymbol{r}_{m'}$, where $\boldsymbol{r}_m$ is the center of group m, and κ is a normalization constant. For EFIE, the basis function signature $\boldsymbol{R}_{jm}(k,\hat{\boldsymbol{k}}_p) = \boldsymbol{V}_{jm}(k,\hat{\boldsymbol{k}}_p)^*$ and

$$\boldsymbol{V}_{jm}(k,\hat{\boldsymbol{k}}) = \int_{\boldsymbol{r}^2} \boldsymbol{f}_j(\boldsymbol{r}) \cdot [\overline{\overline{I}} - \hat{\boldsymbol{k}}\hat{\boldsymbol{k}}] e^{-jk\hat{\boldsymbol{k}}\cdot(\boldsymbol{r}_m - \boldsymbol{r})}\, d^2\boldsymbol{r}, \tag{3}$$

and Rokhlins translation function T_L [6] is computed as

$$T_L(k,\hat{\boldsymbol{k}},\boldsymbol{x}) = \sum_{l=0}^{L}(-j)^l(2l+1)h_l^{(2)}(k|\boldsymbol{x}|)P_l(\hat{\boldsymbol{k}}\cdot\hat{\boldsymbol{x}}), \quad (4)$$

where $\hat{\boldsymbol{k}}$ is the unit wave vector, $\boldsymbol{x}$ is the vector between two group centers directed towards the receiving group, $\hat{\boldsymbol{x}} = \boldsymbol{x}/|\boldsymbol{x}|$, $h_l^{(2)}$ is the spherical Hankel function of second kind and order l, and P_l is the Legendre polynomial of order l. T_L should be computed as shown in [7].

Discretizing (2), we get

$$\overline{\overline{Z}}_{j,i} = \kappa\sum_{p=1}^{K} w_p \boldsymbol{R}_{jm}(k,\hat{\boldsymbol{k}}_p)\cdot\left(T_L(k,\hat{\boldsymbol{k}}_p,\boldsymbol{r}_{mm'})\boldsymbol{V}_{im'}(k,\hat{\boldsymbol{k}}_p)\right). \quad (5)$$

Here, w_p are the integration weights.

III. Higher-Order MLFMM

One way of reducing the memory in an integral-equation solver in general, and MLFMM in specific, is to apply Higher-Order (HO) basis functions [8], which greatly reduce the required number of unknowns N by increasing the order of the polynomial basis functions used to represent the surface current density.

For the Method of Moments (MoM), which requires at least $\mathcal{O}(N^2) = \mathcal{O}(f^4)$ memory and time, even modest reductions of N allow significant savings. In particular, MoM based on hierarchical HO basis functions is well-suited for problems requiring high accuracy, since the basis function order can be increased to achieve exponential improvement in accuracy, allowing high accuracy at the expense of a very moderate increase of N. This feature is not shared by implementations based on first-order basis functions, such as RWG [9].

Combining HO basis functions with MLFMM was done in [10]. This work resulted in a HO MLFMM algorithm with $\mathcal{O}(f^2\log f)$ scaling and also strong scaling against increasing accuracy requirements and geometric complexity, which is challenging because using HO basis functions lead to increased group sizes and therefore increased L in (4) and K in (5).

Further, as discussed in [10], a number of details in the MLFMM implementation have been tweaked to allow lower memory requirements and/or higher computation speeds without reducing the accuracy.

A. Out-of-Core

The concept of Out-of-Core (OoC) means the application of storage that is slower, but cheaper, than Random Access Memory (RAM). The point of OoC is thus clear - store rarely used data on a cheap medium, reducing the memory requirements of the solver, though at the expense of longer read/write times when accessing that data. In recent years, tremendous advances have been made in the read/write speeds of cheap storage media, making OoC implementations increasingly relevant.

The application of Out-of-Core storage to reduce the memory requirement in connection with MLFMM has previously been discussed briefly in the literature. Storing the near-matrix OoC was treated in [11, Section V] and [12] discussed OoC storage of the MLFMM basis function patterns. However, in this paper, we present results for the simultaneous OoC storage of all of the following components:

- Near-matrix,
- Preconditioner,
- Basis function patterns,
- GMRES Solver storage,

as we previously reported in [13], but with several improvements being made since then to greatly increase the speed of the OoC implementation.

We stress that MLFMM based on HO basis functions is particularly well suited for OoC storage, because HO MLFMM changes how the MLFMM memory is used, compared to low-order MLFMM. For OoC, HO MLFMM has two advantages over MLFMM based on RWG basis functions.

First, the memory use of an HO MLFMM code lies predominantly in the four items listed above which are used only once or twice per iteration, while a MLFMM code based on RWG has memory spread over a wider set of data, including translation operators and interpolation data, that is used many times per iteration. That means that the relative reduction from using OoC is greater for HO MLFMM than for RWG-based MLFMM.

Second, the components above, particularly the near-matrix and the preconditioner, lead to the lowest time penalty when using OoC. This is because the near-matrix and preconditioner can be read in chunks, used and then discarded, without affecting the total memory footprint and with a very easily controlled balance between memory and speed. This is in contrast with RWG implementations where the basis function patterns are costly — for such implementations, OoC will be much more costly in terms of computation speed. Further, our use of the modifications detailed in [10, Section III] allows a significant reduction in the amount of data to be transferred to/from the disk in an OoC implementation.

B. Simulation setup and costs of computing resources

We emphasize that to find reliable values for the time required for OoC, one cannot simply run an OoC simulation on a machine that has sufficient memory for an in-core run. Many operating systems will cache file I/O to memory and thus, in some cases, will provide overly optimistic timings for Out-of-Core simulations, because the operating system is simply using the memory (rather than disk) behind the back of the user.

Therefore, all results in this paper are based on running the simulation on a system with sufficient memory for an in-core simulation, and then physically removing the memory from the computer, before running the Out-of-Core simulation. This makes the reduction in available memory the only hardware change between the in-core and Out-of-Core runs.

Further, an important point is the cost of computing resources. The point of OoC is that the financial cost of the storage media is much lower than the cost of RAM while

TABLE I
COMPARISON OF PRICE AND SPEED FOR DIFFERENT STORAGE MEDIA.

	DDR4 Ram	NVMe
Price pr. GB [USD]	11	0.25
Read/Write speed [GB/s]	20	3

only being moderately slower. In Table I, we see a comparison between the DDR4 RAM used for in-core computations and an SSD that can be mounted via the NVMe interface. The prices and read/write speeds are to be taken qualitatively — the point of this paper is not to provide an in-depth analysis of the practically obtainable read/write speeds, nor perform a detailed review of internet pricing of computer hardware.

As the table shows, storage on an SSD mounted via the NVMe interface is about 40 times cheaper than DDR4 RAM, but the read/write speed is $6-7$ times slower. Further, as discussed above, only a modest percentage of the time in an HO MLFMM implementation will be spent accessing the OoC components, so that factor of $6-7$ should not affect run-time significantly. The following experiments will test the performance on a few different cases.

IV. RESULTS

All results in this paper are produced on a HP DL380 Gen9 computer, with 2 Intel Xeon E5-2690 processors @ 2.6 GHz, DDR4 RAM @ 2133 MHz.

In TICRAs software ESTEAM, the possible use of disk storage can automatically be determined when allowed so by the user. If the computer has sufficient memory to run the scattering problem in-core, no disk storage is used. If there is insufficient memory available, and the user allows it, ESTEAM will store some (or all) of the components listed in Section III-A on the disk. This allows the lowest possible computation time for a given machine, while still ensuring that the problem will run.

A. Reflector

We begin by considering a canonical reflector case. A $1\,\mathrm{m}$ diameter circular paraboloidal reflector system is illuminated by a simple gaussian beam placed in the focal point $0.6\,\mathrm{m}$ away from the center of the reflector. We consider a frequency of $250\,\mathrm{GHz}$, where the electrical size of the reflector is $569,231\lambda^2$, discretized with patches of sidelength up to 1.5λ. This results in 15.5 million Higher-Order unknowns, with up to 5^{th} order polynomials used on each patch.

The results are reported in Table II, although we stress that at these frequencies, the analysis would be done much more efficiently by applying Physical Optics, perhaps augmented with the Physical Theory of Diffraction. Thus, the intent here is only to provide a configuration that can be scaled in frequency in a simple way.

From the table we see clearly the benefits of Out-of-Core. A reduction of a factor of 6 in memory, while the increase in computational time is fairly modest, although noticeable.

TABLE II
PERFORMANCE OF OUT-OF-CORE SIMULATION OF THE REFLECTOR IN SECTION IV-A.

	In-core	Out-of-core
Memory	162 GB	27 GB
Disk	0 GB	135 GB
Time pr. iteration	2 min	3 min
Total time	102 min	152 min

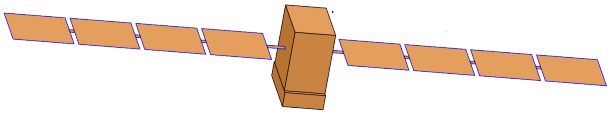

Fig. 1. Satellite platform used in Section IV-B.

B. Satellite Platform

We examine a fairly primitive satellite platform, including solar panels, as shown in Figure 1. The platform is illuminated by a theoretical feed, intended to illuminate an on-board reflector (not shown). With the solar-panels deployed, the maximum width is about 24 meters, making for a challenging simulation problem at 30 GHz.

The scattering problem has just shy of 17 million Higher-Order unknowns, corresponding to about 85 million RWG unknowns, with polynomials up to 5^{th} order being used on each patch. The electrical surface area is 757,533 λ^2. The satellite body is discretized as a closed scatterer, using the CFIE, while the solar panels are modified as infinitely thin using the EFIE. The convergence is very fast using an inner-outer GMRES solver [14], achieving convergence in 7 iterations.

We see from Table III that the reduction of memory from 305 GB to 57 GB is substantial, as the most memory demanding parts of the MLFMM implementation are put on the NVMe SSD. The time per iteration is roughly doubled, but with convergence being achieved rapidly, the total time is increased by only about 60%.

C. Feed Array on Satellite Platform

The final example is a more realistic satellite platform, based on a detailed model used for test purposes at TICRA, and originally given by Marco Sabbadini of ESTEC. The satellite is analyzed at $40\,\mathrm{GHz}$ where it is electrically quite large, $1,863,702\lambda^2$, requiring nearly 31 million HO unknowns, corresponding to about 150 million RWG unknowns. The structure is illuminated by two orthogonal modes originating

TABLE III
PERFORMANCE OF OUT-OF-CORE SIMULATION OF THE $750 \cdot 10^3\lambda^2$ SATELLITE PLATFORM IN SECTION IV-B.

	In-core	Out-of-core
Memory	305 GB	57 GB
Disk	0 GB	248 GB
Time pr. iteration	10 min	20 min
Total time	170 min	290 min

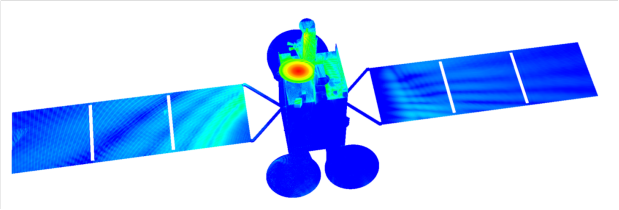

Fig. 2. Surface current distribution as the result of the MLFMM solution. Section IV-C.

TABLE IV
PERFORMANCE OF OUT-OF-CORE SIMULATION OF THE $1.9 \cdot 10^6 \lambda^2$ PLATFORM IN SECTION IV-C.

	In-core	Out-of-core
Memory	530 GB	145 GB
Disk	0 GB	385 GB
Time pr. iteration	40 min	88 min
Total time	990 min	1780 min

from a horn in the feeding array, which results in two right-hand sides, solved simultaneously using a GMRES solver with deflation.

As seen in Table IV, the difference in memory requirement is now about 400 GB, which can be decisive in whether or not it is possible to run the simulation on the available hardware. The increase in computational time is obviously unfortunate, but if the alternative is not to run the simulation at all, the computational time is likely acceptable.

V. CONCLUSION

While most of the published scientific research on large-scale scattering problems has focused on solving large problems as quickly as possible, this paper has explored a different scenario: Solving as large problems as possible as *cheaply* as possible. The economic cost of computing hardware is obviously of practical importance and, in many cases, the computing hardware is fixed. Thus, in the hunt for solving larger and larger problems, we must instead utilize as much of the existing hardware as possible.

In this paper, we used an NVMe mounted SSD and demonstrated that the computation speed does not suffer as much as one could expect. Further, we noted that the NVMe SSD is much cheaper per GB of storage than purchasing the corresponding DDR4 memory. Thus, with an efficient HO MLFMM implementation, adding Out-of-Core capabilities allows for much bigger problems to be solved in reasonable time on a cheaper computer. This has been done in TICRAs antenna siting and placement software ESTEAM, which has produced all results in this paper.

REFERENCES

[1] C.-C. Lu and W. C. Chew, "A Multilevel Algorithm for Solving a Boundary Integral Equation of Wave Scattering," *Microwave and Optical Technology Letters*, vol. 7, no. 10, pp. 466–470, Jul. 1994.

[2] S. Velamparambil and W. C. Chew, "Analysis and Performance of a Distributed Memory Multilevel Fast Multipole Algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 8, pp. 2719–2727, Aug. 2005.

[3] J. Guan, S. Yan, and J.-M. Jin, "An OpenMP-CUDA Implementation of Multilevel Fast Multipole Algorithm for Electromagnetic Simulation on Multi-GPU Computing Systems," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 7, pp. 3607–3616, Jul. 2013.

[4] R. F. Harrington, *Field Computation by Moment Methods*. New York: MacMillan, 1968.

[5] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, Jun. 1982.

[6] V. Rokhlin, "Diagonal Forms of Translation Operators for Helmholtz Equation in Three Dimensions," Yale University, Tech. Rep., 1992.

[7] I. Hänninen and J. Sarvas, "Efficient Evaluation of the Rokhlin Translator in Multilevel Fast Multipole Algorithm," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 8, pp. 2356–2362, Aug. 2008.

[8] E. Jørgensen, J. Volakis, P. Meincke, and O. Breinbjerg, "Higher Order Hierarchical Legendre Basis Functions for Electromagnetic Modeling," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 11, pp. 2985–2995, Nov. 2004.

[9] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic Scattering by Surfaces of Arbitrary Shape," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 3, pp. 409–418, May 1982.

[10] O. Borries, P. Meincke, E. Jørgensen, and P. C. Hansen, "Multi-level Fast Multipole Method for Higher-Order Discretizations," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 9, pp. 4695–4705, Sep. 2014.

[11] I. van den Bosch, M. Acheroy, and J.-P. Marcel, "Design, Implementation, and Optimization of a Highly Efficient Multilevel Fast Multipole Algorithm," in *Computational Electromagnetics International Workshop*. IEEE, 2007, pp. 1–6.

[12] M. Hidayetoğlu and L. Gürel, "MLFMA memory reduction techniques for solving large-scale problems," *Antennas and Propagation Society International Symposium*, pp. 749–750, 2014.

[13] O. Borries, E. Jørgensen, and P. Meincke, "Solution of electrically large scattering problems on a laptop," in *IEEE Antennas and Propagation Symposium*, 2015.

[14] T. F. Eibert, "Some Scattering Results Computed by Surface-Integral-Equation and Hybrid Finite-Element—Boundary-Integral Techniques, Accelerated by the Multilevel Fast Multipole Method," *IEEE Antennas and Propagation Magazine*, vol. 49, no. 2, pp. 61–69, Apr. 2007.