OUT-OF-CORE IMPLEMENTATION OF THE PARALLEL MULTILEVEL FAST MULTIPOLE ALGORITHM

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Barışcan Karaosmanoğlu

August 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Levent Gürel (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Vakur Ertürk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Özgür Ergül

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

OUT-OF-CORE IMPLEMENTATION OF THE PARALLEL MULTILEVEL FAST MULTIPOLE ALGORITHM

Barışcan Karaosmanoğlu
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Levent Gürel
August 2013

We developed an out-of-core (OC) implementation of the parallel multilevel fast multipole algorithm (MLFMA) to solve electromagnetic problems with reduced memory. The main purpose of the OC method is to reduce in-core memory (primary storage) by using mass storage (secondary storage) units. Depending on the OC implementation, the in-core data may be left in one piece or divided into partitions. If the latter, the partitions are written out into mass storage unit(s) and read into in-core memory when required. In this way, memory reduction is achieved. However, the proposed method causes time delays because reading and writing large data using massive storage units is a long procedure. In our case, repetitive access to data partitions from the mass storage increases the total time of the iterative solution part of MLFMA. Such time delays can be minimized by selecting the right data type and optimizing the sizes of the data partitions. We run the optimization tests on different types of mass storage devices, such as hard disks and solid state drives.

This thesis explores OC implementation of the parallel MLFMA. To be more precise, it presents the results of optimization tests done on different partition sizes and shows how computation time is minimized despite the time delays. This thesis also presents full-wave solutions of scattering problems including hundreds of millions of unknowns by employing an OC-implemented parallel MLFMA.

Keywords: Out-of-core methods, memory reduction, computational electromagnetics, fast solvers, multilevel fast multipole algorithm, parallel computing, electromagnetic scattering.

ÖZET

PARALEL ÇOK SEVIYELİ HIZLI ÇOKKUTUP ALGORİTMASININ ÇEKİRDEK DIŞI UYGULAMASI

Barışcan Karaosmanoğlu
Elektrik ve Elektronik Mühendisliği Bölümü, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Levent Gürel
Ağustos 2013

Elektromanyetik problemlerini indirgenmiş bellek ile çözebilmek adına paralel çok seviyeli hızlı çokkutup yönteminin (ÇSHÇY) çekirdek-dışı (ÇD) uygulaması geliştirilmiştir. ÇD yöntemlerinin esas amacı, yığınsal bellek (ikincil bellek) birimleri kullanılarak çekirdek-içi bellek (birincil bellek) kullanımını azaltmaktır. ÇD uygulamanın türüne göre, çekirdek-içi veri tek parça halinde bırakılabilir ya da parçalara bölünebilir. Parçalar, yığınsal bellek birimlerine yazıldıktan sonra gerektiğinde geri okunarak çekirdek-içi bellege alınır. Bu sayede bellek indirgenmesi sağlanmış olur. Fakat, önerilen yöntem, yığınsal bellek birimlerine büyük veri yazılmasının ve okunmasının uzun sürmesinden dolayı gecikmelere yol açar. Bizim durumumuzda, yineli bir şekilde yığınsal bellekteki veri parçalarına erişilmesi, ÇSHÇY'nin döngülü çözüm kısmının toplam süresini artırmaktadır. Bahsedilen zaman gecikmeleri, doğru veri türünü ve eniyilenmiş veri parça boyutlarını kullanarak azaltılabilir. Sabit diskler ve katıhal diskleri gibi çeşitli yığınsal bellek birimlerinde eniyileme testleri yapılmıştır.

Bu tezde paralel ÇSHÇY'nin CD uygulaması incelenmiştir. Daha net olarak, farklı parça boylarında yapılan eniyileme test sonuçları sunulmuş ve oluşan zaman gecikmelerine ragmen çözüm süresindeki düşüş gösterilmiştir. Ayrıca bu tezde, paralel ÇSHÇY'nin CD uygulaması ile çözülmüş yüzlerce milyon bilinmeyenli saçılım problemlerinin tam dalga sonuçları sunulmuştur.

Anahtar sözcükler: Çekirdek-dışı yöntemler, bellek indirimi, hesaplamalı elektromanyetik, hızlı çözücüler, çok seviyeli hızlı çokkutup algoritması, paralel hesaplama, elektromanyetik saçılım.

Acknowledgement

I would like to express my gratitude to my supervisor Prof. Levent Gürel for his supervision, guidance, and suggestions throughout the development of this thesis. I would also like to express my deepest gratitude to him for supporting my studies on the computational electromagnetics.

I also would like to thank Assoc. Prof. Vakur Ertürk and Assist. Prof. Özgür Ergül for reading and commenting on this thesis.

I was fortunate to work with BiLCEM researchers, Mert Hidayetoğlu, Aslan Etminan, Mahdi Kazempour, and Manouchehr Takrimi. I thank them all for their collaboration and for their friendship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we present the implementation of an out-of-core (OC) method for the multilevel fast multipole algorithm (MLFMA). Out-of-core implementations are used on many types of solvers and linear algebra packages [1]. There are various examples of OC implementations, such as simple matrix-vector multiplications (MVM) and N-body simulations [2]. Because OC algorithms aim to reduce current memory consumption, they have been applied on the method of moments (MOM)-based electromagnetic solvers [3]. A solver based on parallel MOM using an OC method is introduced in [4] and a parallel fast multipole method (FMM) using an OC method is introduced in [5]. However, solutions of large-scale electromagnetics problems require solvers with reduced computational complexity and low memory requirement. Therefore, instead of MOM, MLFMA, which is the multilevel implementation of FMM, can be a preferred solver since it has $\mathcal{O}(N \log N)$ computational complexity and memory requirement. Although [6] compares an OC implementation of the sequential MLFMA and in-core MLFMA, the technical details are not presented.

The main objective of this thesis is to reduce memory for the parallel MLFMA using OC techniques without increasing the total computational complexity. It is well known that secondary mass storage usage within an in-core algorithm causes an inevitable increase in work time. However, this time loss in exchange for the memory savings is important. Thus, we investigate OC methods and proper

buffer sizes to find an optimal buffer size with the minimum time loss caused by the OC operations.

In the next section, we provide background information about surface integral equations (SIEs) and present the formulation for converting a 3-D physical problem (discretizing SIEs) into a 2-D matrix equation using MOM. Then, using the addition theorem, we provide the MLFMA formulations for far-field interactions.

In the third section, we explain the implementation of the OC method into MLFMA. We perform memory profiling, detect peak memories and discuss the effect of data types and the storage device types. Last, we share the details of OC implementation on major MVM elements.

In the fourth section, we present the experimental results and perform optimization tests for various sizes of sphere scattering problems. Last, we present the full-wave solutions of large-scale sphere and NASA Almond geometries using optimal OC buffers.

Chapter 2

Background

2.1 Surface Integral Equations

Surface integral equations are widely used to formulate scattering and radiation problems for 3-D arbitrary geometries [7, 8]. Equivalent surface currents are defined on the arbitrary 3-D object and integral equations can be obtained using physical boundary conditions. For the perfect electric conductor (PEC) problems, the electric-field integral equation (EFIE), magnetic-field integral equation (MFIE) and combined-field integral equation (CFIE) are the most commonly used formulations.

The EFIE formulation is obtained by a physical boundary condition that states that the total tangential electric field must be zero on a conducting surface. The mathematical expression of EFIE can be given as

$$\hat{\boldsymbol{t}} \cdot \int_{S'} d\boldsymbol{r'} \overline{\boldsymbol{G}}(\boldsymbol{r}, \boldsymbol{r'}) \cdot \boldsymbol{J}(\boldsymbol{r'}) = \frac{i}{k\eta} \hat{\boldsymbol{t}} \cdot \boldsymbol{E}^{\text{inc}}(\boldsymbol{r}), \tag{2.1}$$

where $\boldsymbol{E}^{\text{inc}}$ is the incident field, S' is the surface of the object, $\boldsymbol{J}$ is the induced surface current and η is the intrinsic impedance of the medium. In scattering problems, $\boldsymbol{J}$ is the unknown. $\overline{\boldsymbol{G}}(\boldsymbol{r}, \boldsymbol{r'})$ is the dyadic Green's function, defined as

$$\overline{\boldsymbol{G}}(\boldsymbol{r}, \boldsymbol{r'}) = \left[\overline{\boldsymbol{I}} + \frac{\nabla\nabla}{k^2} \right] g(\boldsymbol{r}, \boldsymbol{r'}), \tag{2.2}$$

where

$$g(\boldsymbol{r}, \boldsymbol{r'}) = \frac{e^{ik|\boldsymbol{r}-\boldsymbol{r'}|}}{4\pi|\boldsymbol{r}-\boldsymbol{r'}|} \tag{2.3}$$

is the scalar Green's function for the 3-D Helmholtz equation. The scalar Green's function is the response of a point source located at $\boldsymbol{r}$, which is observed at point $\boldsymbol{r'}$.

Similar to EFIE, MFIE can be obtained using physical boundary conditions on a tangential magnetic field on a conducting object:

$$\boldsymbol{J}(\boldsymbol{r}) - \hat{\boldsymbol{n}} \times \int_{S'} d\boldsymbol{r'} \boldsymbol{J}(\boldsymbol{r'}) \times \nabla' g(\boldsymbol{r}, \boldsymbol{r'}) = \hat{\boldsymbol{n}} \times \boldsymbol{H}^{\text{inc}}(\boldsymbol{r}), \tag{2.4}$$

where $\boldsymbol{H}^{\text{inc}}$ is the incident magnetic field and $\hat{\boldsymbol{n}}$ is the normal unit vector on the surface S'.

Note that EFIE can be used on both open and closed geometries, whereas MFIE can only be applied on closed geometries. Therefore, CFIE, the linear combination of EFIE and MFIE, can only be applied on closed geometries. The aim of CFIE is to obtain a better-conditioned linear system from both EFIE and MFIE. The CFIE formulation is given as

$$\text{CFIE} = \alpha\text{EFIE} + (1-\alpha)\text{MFIE}, \tag{2.5}$$

where α is a parameter between 0 and 1. Because it yields minimal iterations, α is set between 0.2 and 0.3 [9].

2.2 Discretization of Surface Integral Equations

To numerically solve electromagnetic scattering and radiation problems of complicated objects, SIEs must be discretized.

2.2.1 Method of Moments

SIEs can be converted into matrix equations using MOM. The equivalent surface currents are expanded in terms of the basis functions. The coefficients of these

functions are calculated by solving matrix equations obtained by MOM. The integral equations can be written as

$$\mathcal{L}\{\boldsymbol{f}(\boldsymbol{r})\} = \boldsymbol{g}(\boldsymbol{r}), \tag{2.6}$$

where $\mathcal{L}$ is a linear operator on the equivalent surface currents and $\boldsymbol{g}$ is the right-hand-side (RHS) function of EFIE and/or MFIE, which is the combination of the incident electromagnetic fields generated by external sources. Considering $\boldsymbol{f}$ as unknown, expanding $\boldsymbol{f}$ in a series of known basis functions and unknown coefficients, we obtain

$$\boldsymbol{f}(\boldsymbol{r}) \approx \sum_{n=1}^{N} a_n \boldsymbol{b}_n(\boldsymbol{r}). \tag{2.7}$$

Testing (2.6) using the testing functions, which are the same as the basis function (Galerkin scheme), we can obtain

$$\int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \sum_{n=1}^{N} a_n \mathcal{L}\{\boldsymbol{b}_n(\boldsymbol{r})\} = \int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{g}(\boldsymbol{r}). \tag{2.8}$$

Changing the order of summation and integration, the equation becomes

$$\sum_{n=1}^{N} a_n \int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \mathcal{L}\{\boldsymbol{b}_n(\boldsymbol{r})\} = \int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{g}(\boldsymbol{r}), \tag{2.9}$$

which yields a matrix equation

$$\sum_{n=1}^{N} a_n Z_{mn} = v_m, \tag{2.10}$$

where

$$Z_{mn} = \int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \mathcal{L}\{\boldsymbol{b}_n(\boldsymbol{r})\}, \tag{2.11}$$

and

$$v_m = \int d\boldsymbol{r}\boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{g}(\boldsymbol{r}). \tag{2.12}$$

2.2.2 RWG Functions

3-D surfaces are meshed using triangles. On these triangles, Rao-Wilton-Glisson (RWG) functions [7] are used as linear basis and testing functions, which discretize

the SIEs. These functions are defined on each neighbouring pair of triangles. These triangular basis functions can be written as

$$\boldsymbol{b}_n(\boldsymbol{r}) = \begin{cases} \dfrac{l_n}{2A_n^+}(\boldsymbol{r} - \boldsymbol{r}_n^+), & \boldsymbol{r} \in S_n^+ \\ \dfrac{l_n}{2A_n^-}(\boldsymbol{r}_n^- - \boldsymbol{r}), & \boldsymbol{r} \in S_n^- \\ 0, & otherwise. \end{cases} \tag{2.13}$$

In (2.13), l_n is the common edge length and A_n^+ and A_n^- are the areas of the first and second triangles, respectively.

Importantly, RWG functions are divergence conforming, which means their divergence is finite everywhere, shown as

$$\nabla \cdot \boldsymbol{b}_n(\boldsymbol{r}) = \begin{cases} \dfrac{l_n}{A_n^+}, & \boldsymbol{r} \in S_n^+ \\ -\dfrac{l_n}{A_n^-}, & \boldsymbol{r} \in S_n^- \\ 0, & otherwise. \end{cases} \tag{2.14}$$

This property simplifies the further steps of the EFIE and MFIE discretization.

2.2.3 Discretization of EFIE

Once EFIE is discretized using MOM, the matrix elements can be obtained from the formulation

$$\begin{aligned} Z_{mn}^{\mathrm{EFIE}} = & \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \int_{S_n} d\boldsymbol{r'} \boldsymbol{b}_n(\boldsymbol{r'}) g(\boldsymbol{r}, \boldsymbol{r'}) \\ & - \frac{i}{k^2} \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \int_{S_n} d\boldsymbol{r'} \boldsymbol{b}_n(\boldsymbol{r'}) \cdot [\nabla \nabla' g(\boldsymbol{r}, \boldsymbol{r'})], \end{aligned} \tag{2.15}$$

where $\boldsymbol{t}_m$ and $\boldsymbol{b}_n$ are the testing and basis functions, respectively. However, the double differentiation of the scalar Green's function is hyper singular. Using the divergence-conforming property of the RWG functions, this singularity can be overcome and the double differentiation on the Green's function is then distributed into two separate functions: testing and basis. Thus, the matrix element

formulation of EFIE becomes

$$Z_{mn}^{\text{EFIE}} = ik \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \int_{S_n} d\boldsymbol{r'} \boldsymbol{b}_n(\boldsymbol{r'}) g(\boldsymbol{r}, \boldsymbol{r'}) - \frac{i}{k^2} \int_{S_m} d\boldsymbol{r} \nabla \cdot \boldsymbol{t}_m(\boldsymbol{r}) \int_{S_n} d\boldsymbol{r'} \nabla' \cdot \boldsymbol{b}_n(\boldsymbol{r'}) g(\boldsymbol{r}, \boldsymbol{r'}). \tag{2.16}$$

The RHS of the discretized EFIE formulation is obtained by testing the incident electric field, which gives

$$v_m^{\text{EFIE}} = -\frac{i}{k\eta} \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{E}^{\text{inc}}(\boldsymbol{r}). \tag{2.17}$$

2.2.4 Discretization of MFIE

Using MOM for this discretization, the matrix elements can be obtained from the formula

$$Z_{mn}^{\text{MFIE}} = \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{b}_n(\boldsymbol{r}) - \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \hat{\boldsymbol{n}} \times \int_{S_n} d\boldsymbol{r'} \boldsymbol{b}_n(\boldsymbol{r'}) \times \nabla' g(\boldsymbol{r}, \boldsymbol{r'}). \tag{2.18}$$

Because the Galerkin scheme is used, the first term of (2.18) becomes a simple integral, but the second term still includes a singularity from the differentiation of the scalar Green's function. After the limit-term extraction [10], (2.18) becomes

$$Z_{mn}^{\text{MFIE}} = \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \boldsymbol{b}_n(\boldsymbol{r}) - \int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \hat{\boldsymbol{n}} \times \int_{S_n, PV} d\boldsymbol{r'} \boldsymbol{b}_n(\boldsymbol{r'}) \times \nabla' g(\boldsymbol{r}, \boldsymbol{r'}), \tag{2.19}$$

where PV is the principal value of the integral. Modifying the second integral in (2.19) gives

$$\int_{S_m} d\boldsymbol{r} (\boldsymbol{t}_m(\boldsymbol{r}) \times \hat{\boldsymbol{n}}) \cdot \boldsymbol{b}_n(\boldsymbol{r}) \times \int_{PV, S_n} d\boldsymbol{r'} \nabla' g(\boldsymbol{r}, \boldsymbol{r'}). \tag{2.20}$$

Similar to EFIE, the RHS of the discretized MFIE formulation is obtained by testing the incident magnetic field, which gives

$$v_m^{\text{MFIE}} = -\int_{S_m} d\boldsymbol{r} \boldsymbol{t}_m(\boldsymbol{r}) \cdot \hat{\boldsymbol{n}} \times \boldsymbol{H}^{inc}(\boldsymbol{r}). \tag{2.21}$$

2.2.5 Discretization of CFIE

CFIE is a linear combination of EFIE and MFIE. The discretized formulations of EFIE and MFIE give the CFIE matrix element formulation, shown as

$$Z^{\mathrm{CFIE}} = \alpha Z^{\mathrm{EFIE}} + (1-\alpha) Z^{\mathrm{MFIE}}. \tag{2.22}$$

2.3 Multilevel Fast Multipole Algorithm

To solve the matrix equation obtained from the discretized EFIE or MFIE, one can use a direct solver or an iterative solver. Because direct solvers, such the Gaussian elimination, have the computational complexity of $\mathcal{O}(N^3)$, large problems require an impossible time duration. On the other hand, the iterative solutions require at least one MVM for each iteration. Direct MVM has both the computational and memory complexity of $\mathcal{O}(N^2)$, which still requires huge amounts of time and memory. Using the addition theorem, FMM may be applied, and the direct MVM computational complexity will reduce to $\mathcal{O}(N^{1.5})$. Applying FMM in a multilevel fashion (MLFMA) [11] would result in the memory and computational complexity of $\mathcal{O}(N \log N)$.

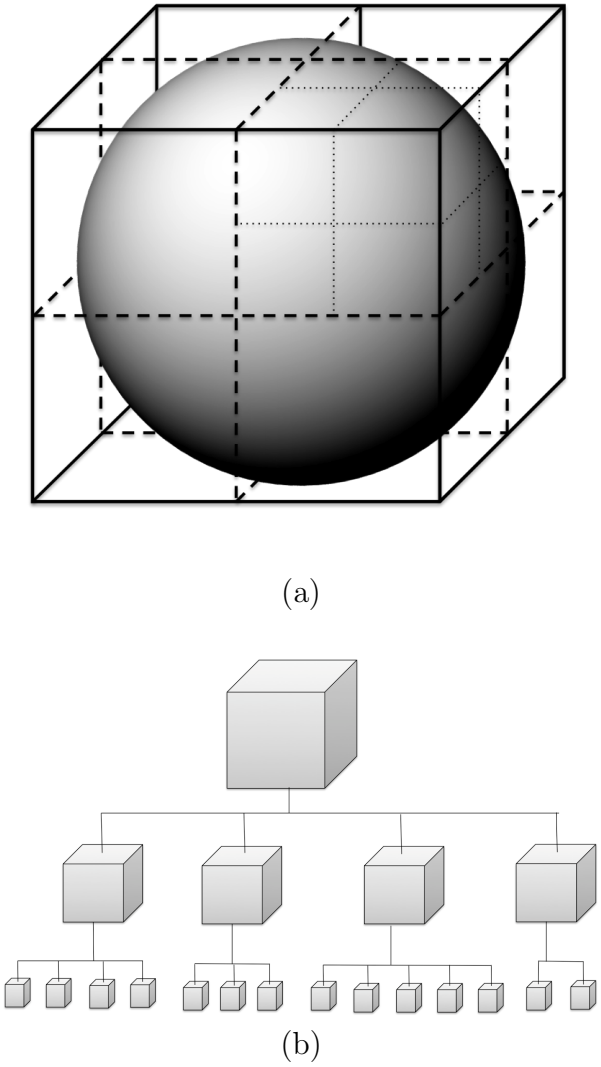
The integro-differential operators $\mathcal{L}$ for SIEs include interactions in close distances and far distances. These two kinds of interactions can be handled separately as

$$\overline{\boldsymbol{Z}} \cdot \boldsymbol{a} = \overline{\boldsymbol{Z}}^{\mathrm{NF}} \cdot \boldsymbol{a} + \overline{\boldsymbol{Z}}^{\mathrm{FF}} \cdot \boldsymbol{a}, \tag{2.23}$$

where $\overline{\boldsymbol{Z}}^{\mathrm{NF}}$ is the matrix of near-field interactions and $\overline{\boldsymbol{Z}}^{\mathrm{FF}}$ is the matrix of far-field interactions. The near-field matrix is directly used and multiplied with the unknown coefficient vector but the far-field interactions are used following a tree structure.

The tree structure is obtained from the clustering operation, as shown in Fig. 2.1. Clustering basically places the geometry into a cube and then recursively divides it into smaller cubes. During the clustering operation, cubes that are part of the geometry will be divided into smaller cubes, and count as a parent cube

in the tree structure, whereas empty cubes will not be divided and not included in the tree structure. This recursive dividing operation will continue until the smallest cubes contain only a few basis functions. All these cubes are called clusters.



(a)

(b)

Figure 2.1: (a) Multilevel clustering of the scatterer. (b) Construction of the multilevel tree structure.

At the lowest level, the near-field matrix is calculated, and the interactions of the basis functions with testing functions either share the same cluster or are in clusters touching each other. Unlike the near-field matrix, the far-field matrix

is never calculated. At each level, the radiated fields of the basis functions or clusters are aggregated into the centers of the parent clusters, using the local interpolation method [12] to match the different field sampling rates between two levels. Then, these fields are translated into the centres of the neighbouring parent clusters. Last, the fields are disaggregated into child clusters or basis functions using their receiving patterns. Interactions between clusters with no neighbouring parents are not calculated for that level. To achieve the complexity of $\mathcal{O}(N \log N)$, interactions are calculated within an error range of the desired accuracy level.

2.3.1 Factorization of the Green's Function

Both FMM and its multilevel implementation, MLFMA, are derived from the factorization and diagonalization of the Green's function. Factorization of the Green's function is based on the addition theorem.

Consider two clusters, C and C', which are at their far zone. To find the interaction between the basis functions in cluster C and the testing functions in cluster C', the scalar Green's function can be factorized as an integration on the unit sphere:

$$\begin{aligned} g(\boldsymbol{r}, \boldsymbol{r'}) &= \frac{e^{ik|\boldsymbol{r}-\boldsymbol{r'}|}}{4\pi|\boldsymbol{r}-\boldsymbol{r'}|} = \frac{e^{ik|\boldsymbol{D}-\boldsymbol{d}|}}{4\pi|\boldsymbol{D}-\boldsymbol{d}|} \\ &\approx \frac{1}{4\pi}\int_{S_m} d^2\hat{\boldsymbol{k}} e^{i\hat{\boldsymbol{k}}\cdot\boldsymbol{d}} \alpha_T(k, D, \psi), \end{aligned} \tag{2.24}$$

where $D = |\boldsymbol{D}|$ is the distance between cluster C and cluster C', and $\hat{\boldsymbol{k}}$ is the normal unit vector on the unit sphere. In (2.24) α_T is the translation function, given as

$$\alpha_T(k, D, \psi) = \sum_{t=0}^{T} i^t (2t+1) h_t^{(1)}(kD) P_t(\cos\psi), \tag{2.25}$$

which is a truncated sum. In (2.25), $h_t^{(1)}$ denotes the spherical Hankel function of the first kind, P_t is the Legendre polynomial, and ψ is the angle between unit vectors $\hat{\boldsymbol{k}}$ and $\hat{\boldsymbol{D}}$.

The truncation number is T_l in any level l of MLFMA, and is obtained by the excess bandwidth formula [13]

$$T_l \approx 1.73ka_l + 2.16d_0^{2/3}(ka_l)^{1/3}, \tag{2.26}$$

where a_l denotes the cluster size and d_0 is the necessary accurate digit number in MLFMA.

After the diagonalization of the scalar Green's function [14], the far-field matrix equation is obtained from the integration:

$$Z_{mn}^{\mathrm{FF}} = \left(\frac{ik}{4\pi}\right)^2 \int d^2\hat{\boldsymbol{k}} \overline{\boldsymbol{F}}_{C_m'}^{\mathrm{rec}}(\hat{\boldsymbol{k}}) \cdot \alpha_T(k, D, \psi) \overline{\boldsymbol{F}}_{C_n}^{\mathrm{rad}}(\hat{\boldsymbol{k}}), \tag{2.27}$$

where $\overline{\boldsymbol{F}}_{C_m'}^{\mathrm{rec}}$ is the receiving pattern of the mth testing function in cluster $C^{'}$ and $\overline{\boldsymbol{F}}_{C_n}^{\mathrm{rad}}$ is the radiation pattern of the nth basis function in cluster C.

Chapter 3

Implementation

OC or external-memory algorithms are designed to process data that is too large to fit into a computer's main memory at one time. We use OC methods frequently in daily life. One example is when taking notes from a textbook instead of memorizing all the information in it. Indeed, it would be faster to use the memorized data than to read the entire text. Further, memorizing some texts might be impossible, which represents lack of memory in our case.

These experiments use this "existing" OC methodology and implement MLFMA to increase the current capacity of the program. Thus, the magnitude of the active memory in use is decreased for the same configurations.

In this section, we explain memory profiling for MLFMA and give disk/data benchmarks, which are required to obtain efficient implementation. Then, we provide the OC implementation of MLFMA on near-field MVM, radiation/receiving patterns and translation.

3.1 Memory Profiling

To implement the OC method successfully, we perform MLFMA memory profiling, investigating the memory requirements of major elements in MVM. Main

bottlenecks, large memory allocations, and deallocations are tracked for different sizes of problems. During this profiling, the program flow is divided into three main parts: preprocessing, setup, and solution. Then, each part is investigated separately.

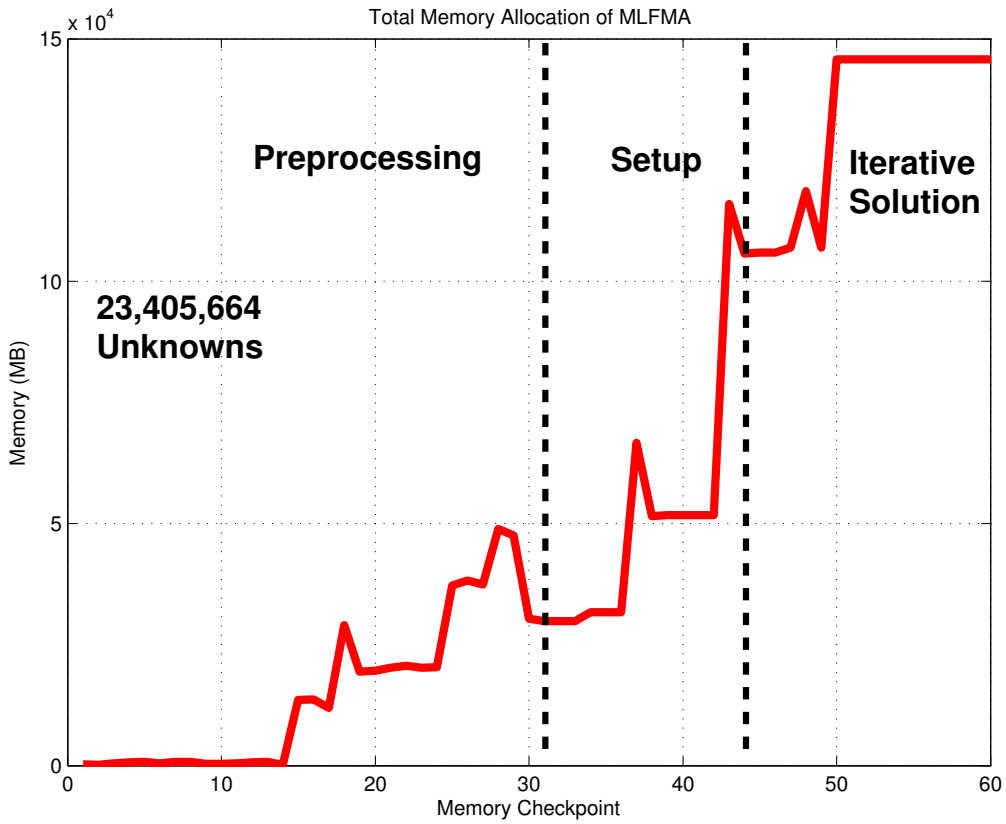


Figure 3.1: Total memory allocation of a 23M-unknowns sphere problem.

Figure 3.1 shows the memory plot of a relatively large problem, a sphere with 23 million (23M) unknowns. Because the solution part is the main bottleneck, that part is analyzed first. The main objective of this investigation is to understand the memory distribution of the MLFMA structure. Relatively larger arrays used in MLFMA would be the first candidates for OC implementation. The major parts of MLFMA are the near-field matrix (with memory complexity $\mathcal{O}(N)$), radiation/receiving patterns (with memory complexity $\mathcal{O}(N)$) and MVM (with memory complexity $\mathcal{O}(N \log N)$). Thus, memory reduction with the OC method is implemented on the near-field MVM, radiation/receiving patterns, and

aggregation array. Aggregation array is used OC in the translation part.

The setup part is considered similar to the solution part. Calculating the near-field matrix and obtaining radiation/receiving patterns is implemented by the OC method. The preprocessing part is considered in another project.

3.2 Disk Type and Data Type Benchmark

Data transfer speeds between in-core and OC algorithms are very fast. In-core methods use memory and thus all data are saved in a binary format. However, OC methods use massive storage devices such as hard disk drives (HDDs) and solid-state drives (SSDs) with various file formats. To observe the timing differences between the disk types and data formats, we prepare a benchmark.

We allocate an array of size 4 and measure the times for writing out to the disk and reading in. We then double the size of the array and measure the times for writing out and reading in. This operation is repeated until the array size reaches 2^{27}. Times are obtained for both binary and ASCII formats on an HDD and an SSD. As evident from Fig. 3.2, the disk type does not affect data transfer speed, but data type results in a huge time difference for both types; the ASCII format requires much more space than the binary format so the data transfer speed decreases in the former.

3.3 Out-of-Core Near-Field Matrix-Vector Multiplication

In the first implementation, the near-field array is filled completely and then saved into the hard drive in small pieces. The second implementation aims to reduce memory during the calculation of the near-field array. Two buffer arrays are filled rather than one large near-field array, and then saved onto the disk.

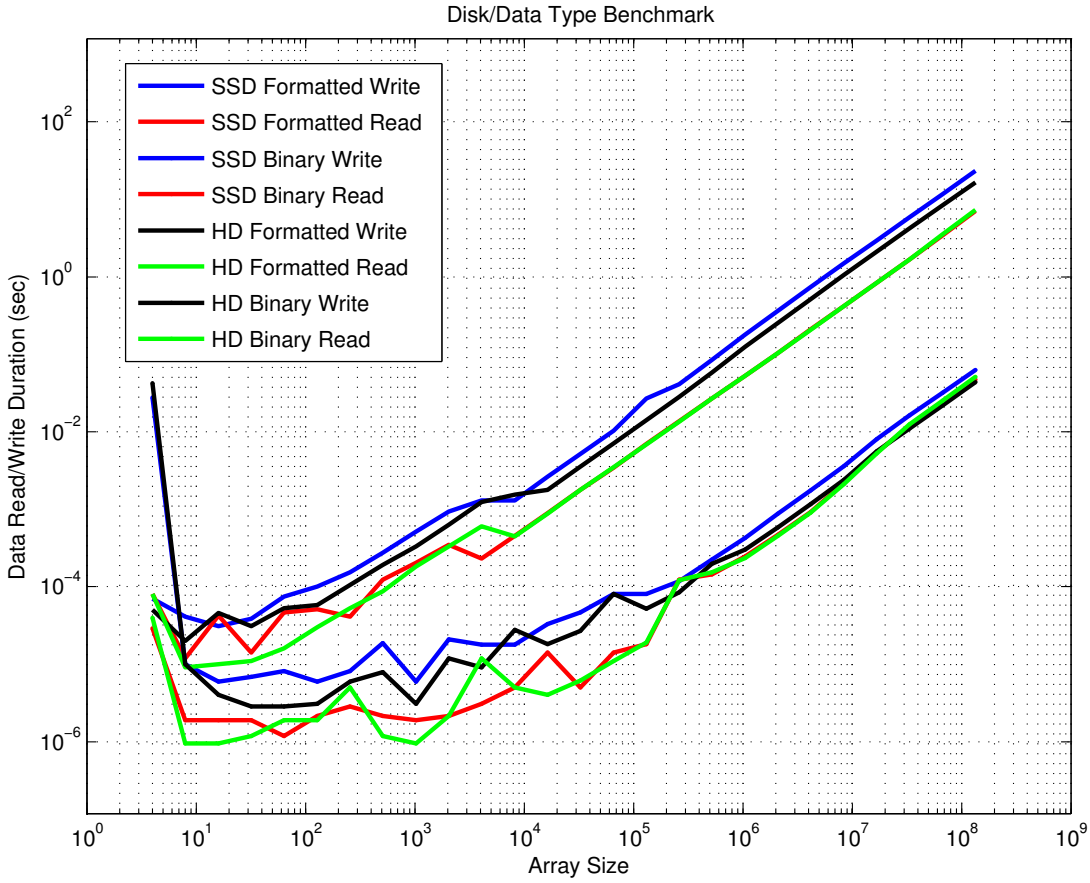


Figure 3.2: HDD and SSD write and read benchmarks with binary and ASCII data.

In MLFMA, the object to be solved is placed into the smallest cube and then divided into eight cubes, and non-empty cubes are divided into another eight cubes (clusters), and so on, as explained earlier. In our case the smallest cluster size is mostly set to 0.25λ. In MOM, each triangle (basis function) interacts with every other triangle. However, MLFMA calculates the interactions of triangles sharing the same cluster at the last level and the interactions of the triangles in the neighbouring clusters. These interactions are the near-field interactions and are stored in memory, then used in the iterative solution directly.

Near-field interactions allocate almost 20% of the total memory. Thus, they are both calculated and use OC. In this study, there are two OC implementations of the near-field interactions. The first implementation, the near-field array is

filled and then divided into small pieces of data. However, for some problems, the memory required in the near-field calculations in the setup part of MLFMA becomes a bottleneck. The second implementation, instead of allocating the whole the near-field array, only two small buffers are allocated. Thus, memory reduction in the calculation of near-field interactions is achieved.

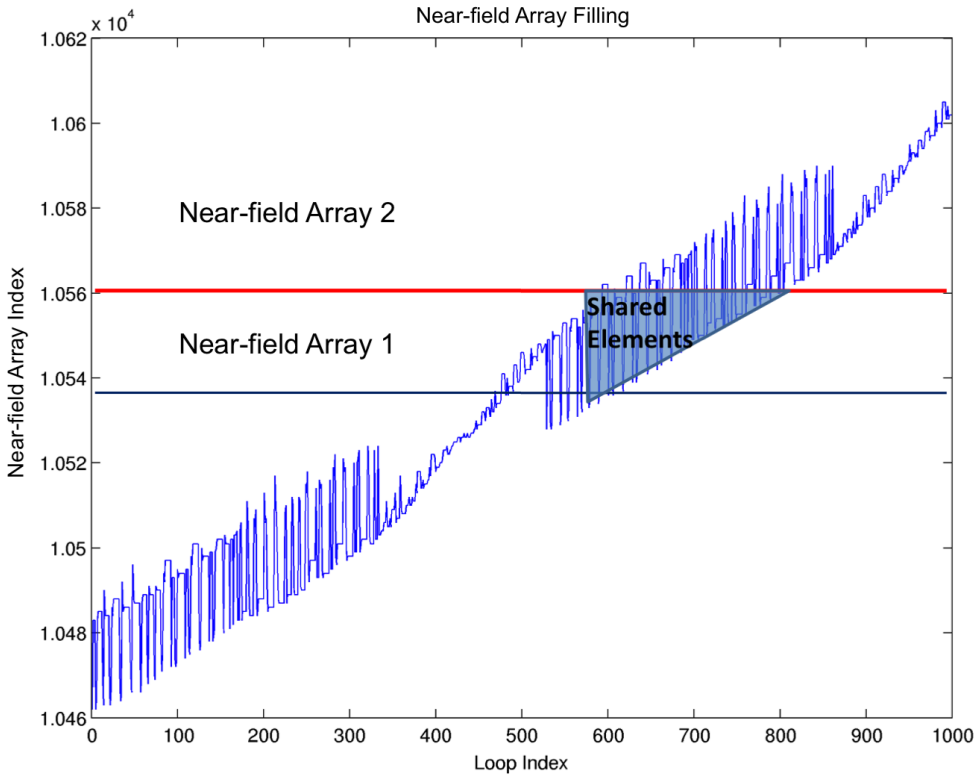


Figure 3.3: Change of the index of the near-field matrix (array) during calculation.

The near-field indexing used in the calculation of the near-field interactions is shown in Fig. 3.3. Because the required index does not monotonically increases, two buffers are necessary. The loop index points to a region of intersection between the first and the second buffers. We need to guarantee that the whole partition is saved into the hard drive when the loop index leaves the first buffer. Then, the first buffer is emptied and the data in the second buffer is transferred into the first buffer. Finally, the data in the second buffer is emptied and the calculation for the next partition begins.

3.4 Out-of-Core Radiation and Receiving Patterns

Similar to the near-field implementation, OC radiation and receiving patterns are implemented in two different ways. In the first case, data is written out in small partitions following the calculation of the patterns. This case aims to test the memory reduction in the solution part. Second, the radiation and receiving patterns are used out of core completely, from the beginning of the calculation to the end of solution part. Instead of using the whole radiation/receiving pattern matrix, small buffers are used only during the calculation. Unlike OC near-field implementation, the required index for the pattern calculation increases monotonically. Thus, the calculation of the patterns requires a single buffer for each partition. When the calculation of a partition is finished, it is written out into the disk.

Radiation and receiving patterns are filled in the order of theta angle, phi angle, basis functions and each triangle of that basis function. The pattern is defined as a 4-D matrix. Therefore, a base index is needed for keeping the OC implementation simple. The chosen base is the index of unknowns. According to this index, the loops of the radiation and receiving pattern calculations in the original implementation are modified. The loop of unknowns and the triangles loop are determined to be the main loops.

3.5 Out-of-Core Translation

In our MLFMA, during translation, incoming fields are obtained by multiplying translation, aggregation and the surface currents, which means both aggregation and disaggregation arrays are needed at the same time. Because the aggregation and disaggregation arrays allocate major memory spaces, aggregation array is used out of core. In the first implementation, aggregation array is saved into the disk in a fixed buffer size, which is 1/50 of the array size. However, this causes too

much data reading from the disks because the required data blocks are separated. To overcome this problem, the aggregation array is saved onto the disk by lining up data blocks larger than the specified buffer size.

During the solution part of MLFMA, MVM follows three steps: aggregation, translation and disaggregation. The aggregation part sums up every field generated by the surface currents on the basis functions and shifts them to the centre of the clusters at each level. In translation, the fields are translated into the center of the cluster that will be interacted with. Last, in disaggregation, the translated fields are distributed to the test functions. The memory reduction of each OC implementation for the different parts of MLFMA is given in Fig. 3.4.

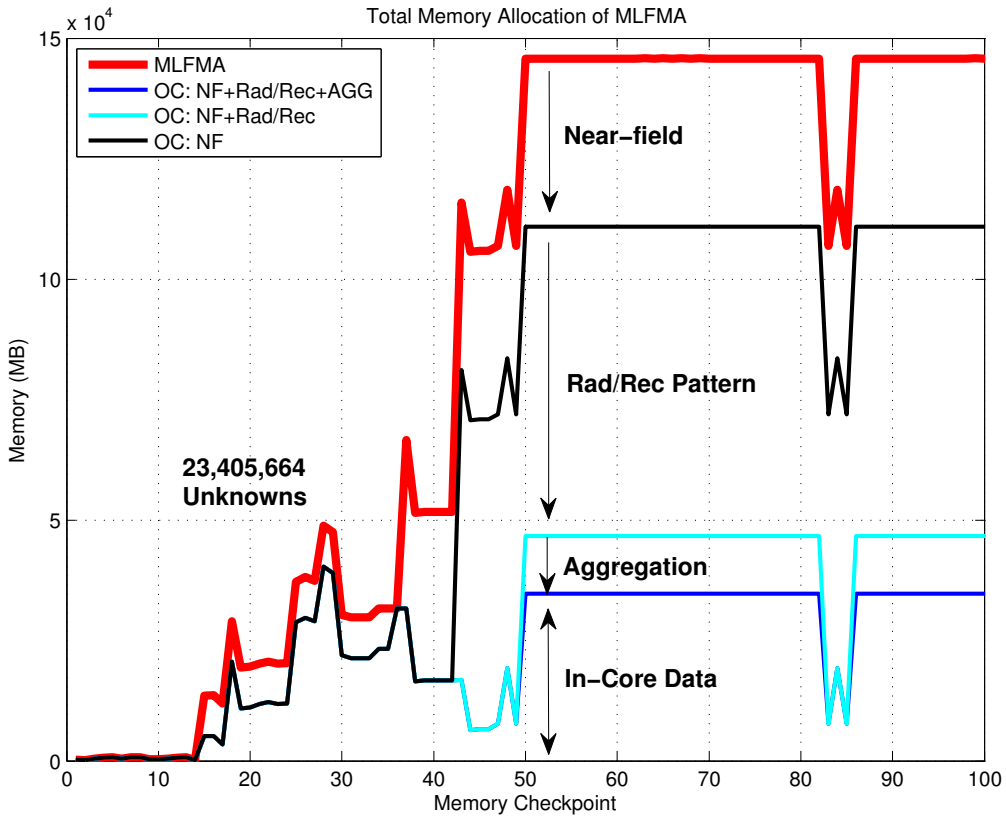


Figure 3.4: Total memory allocation of a 23M-unknowns sphere problem for different steps of MLFMA-OC.

Chapter 4

Experiment Results

In this section, we compare the scalings of the MLFMA-OC with the original MLFMA, then present the optimization results of different parts of the MLFMA-OC. Last, we share extremely large problem solutions using MLFMA-OC.

OC methods and their implementations must compromise between memory and time. Using an OC algorithm will reduce memory usage but increase cpu time. For a 23M-unknowns sphere problem different parts of MLFMA are used out of core; their total iterative solution times are shown in Fig. 4.1. From right to left, we show the results of using no OC memory storage, then using OC, then the radiation/receiving patterns and near-field matrix using OC, and last, near-field matrix, radiation/receiving patterns and aggregation using OC storage.

Observation of the scaling is an another method of determining the efficiency of the OC implementation. Problem-size scaling is performed for the scattering-form-a-sphere problems involving 0.8M, 1.5M, 3M, 23M, 53M, and 93M unknowns. The solutions are handled using 64 processors, comparing MLFMA and MLFMA-OC. In this scaling we obtained total iterative solution times and peak memory per processor. We also obtained problem-size scaling with increasing process number. This scaling test results from the solutions of 23M, 53M, 93M, and 212M unknowns sphere scattering problems using 16, 32, 64, and 128

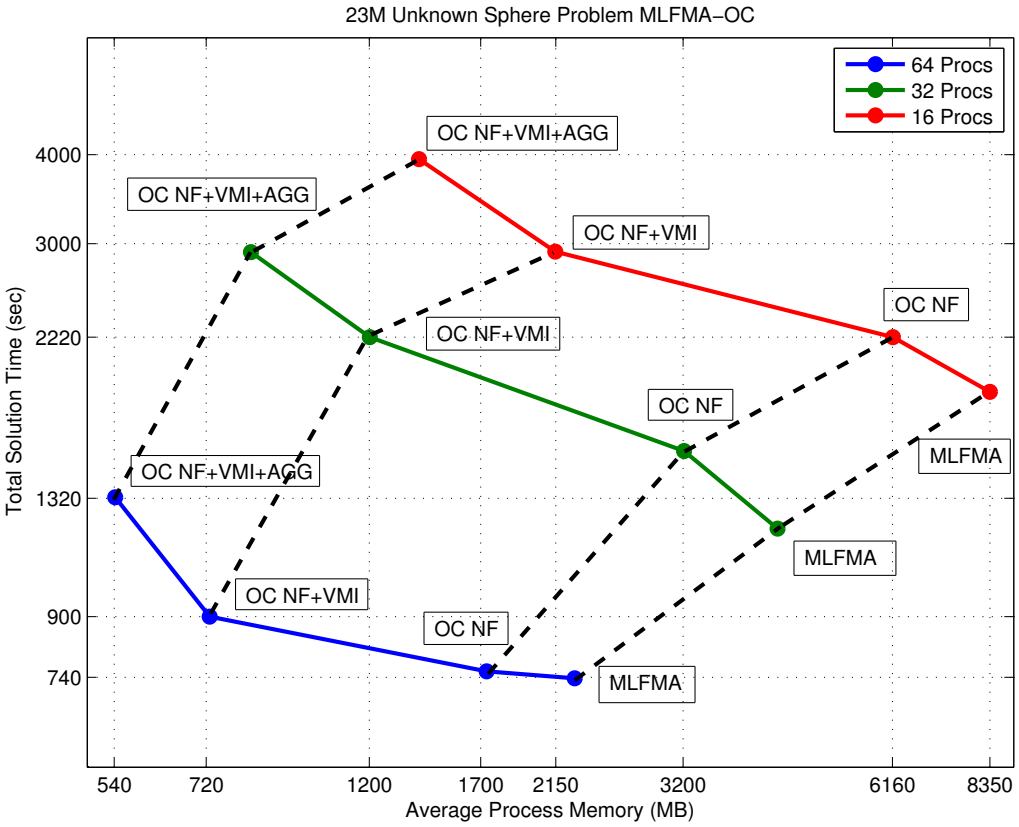


Figure 4.1: A 23M-unknowns sphere scattering problem solution using MLFMA-OC and a time-memory graph of different parts of MLFMA using OC.

processors, respectively. In this scaling we obtained sum of total iterative solution times of each process and total memory required for each problem size. The problem-size scaling and second scaling test results of the MLFMA and MLFMA-OC are quite similar, and this shows that the computational complexity is not changed. The scaling results are shown in Fig. 4.2.

4.1 Optimization

Out-of-core implementation causes delays and time losses in our MLFMA simulations. Although the scaling will not change, time losses might be reduced by setting a proper size of data partition. We thus observe how a change of solution

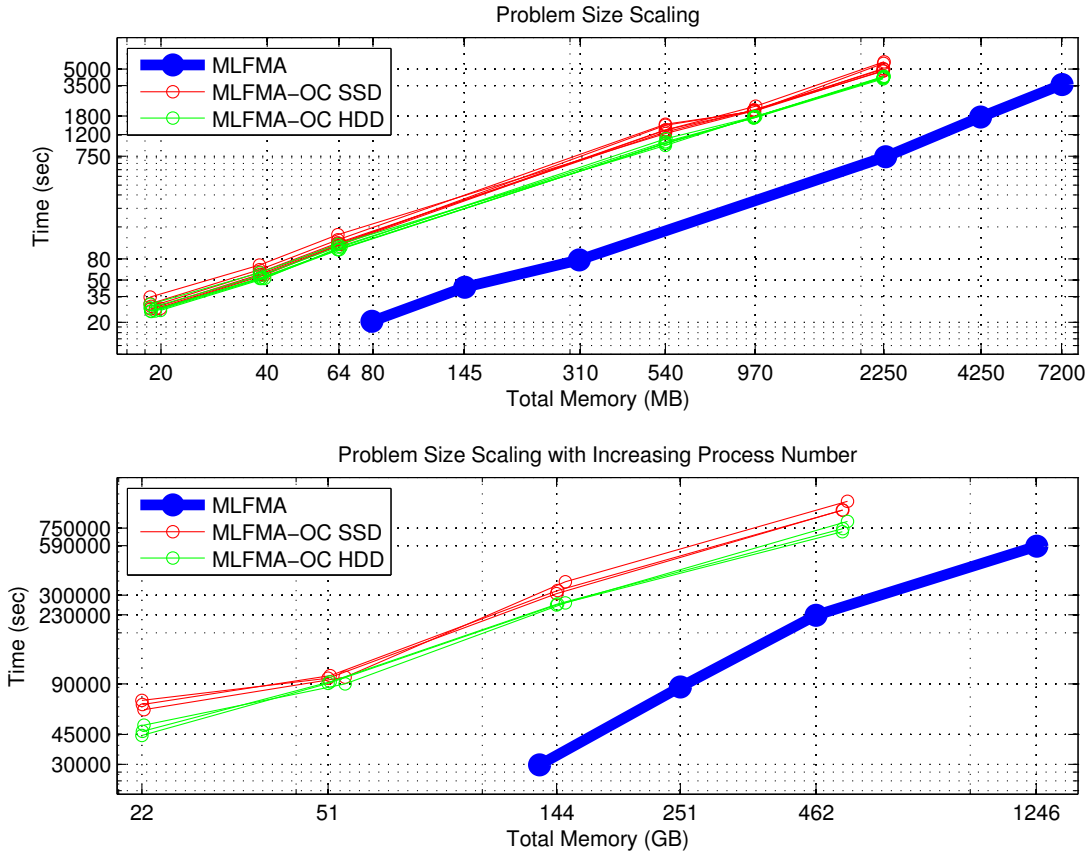


Figure 4.2: Problem size scaling of 0.8M, 1.5M, 3M, 23M, 53M, and 93M unknowns on 64 processes; and problem size scaling with increasing process number of 23M, 53M, 93M, and 212M unknowns on 16, 32, 64, and 128 processes.

time affects partition size. As in the previous simulation, we observe all three parts of the implementation separately. We only measure time for the parts used out of core. We perform several simulations for the HDD and the SSD.

We study the time-memory change for different sizes of sphere scattering problems and solve them with 64 processes, increasing the unknowns of the sphere. Problem sizes are, 0.8M, 1.5M, 3M, 23M, 53M, and 93M, respectively. Because several timing simulations are required for the near-field MVM, we skip the near-field matrix calculation part and therefore do not obtain full solutions. In this section, we explain buffer optimization of the OC near-field matrix, OC aggregation array and OC radiation/receiving patterns and present the results.

4.1.1 Near-field Buffer Optimization

The near-field buffer is kept between 4 KB and 400 MB. For the first four problem sizes, the buffer range is between 4 KB and 40 MB. For the last two problem sizes, the buffers are set between 40 KB and 400 MB. Time-buffer size results of the near-field matrix buffer are given in Fig. 4.3. When the buffer size is too small (between 4 and 40 KB), the number of partitions increases and the processes have too many data partitions to read. When the buffer size is too large (between 40 and 400 MB), the number of partitions decreases, but the reading periods overlaps and overall performance decreases. The range of the buffer size that minimizes the near-field MVM time is between 0.4 and 0.7 MB, which can be declared as the optimal buffer size for the near-field matrix.

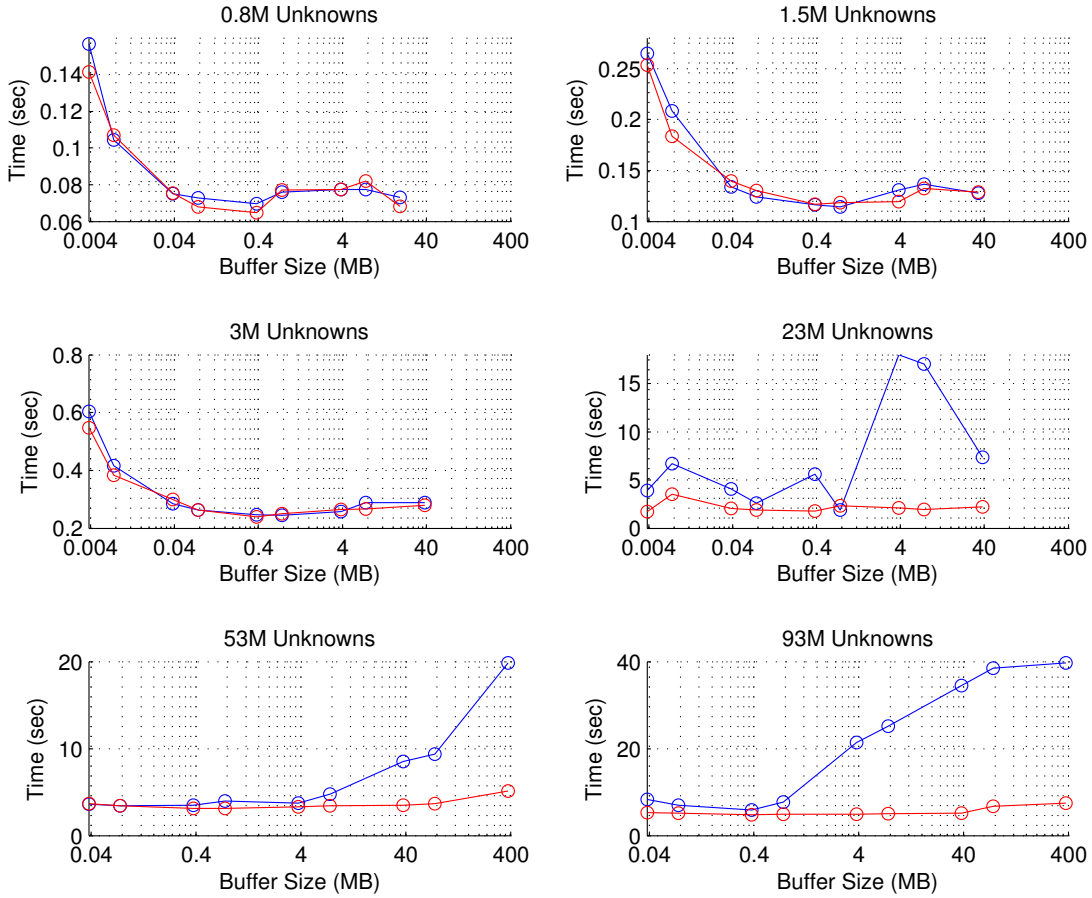


Figure 4.3: The dependence of the CPU time on the size of the near-field buffer. Results are obtained with HDD (red lines) and SDD (blue lines) separately.

4.1.2 Aggregation Buffer Optimization

Similar to the near-field buffer tests, the aggregation buffer test is kept between 4 KB and 400 MB. For the first four problem sizes of the aggregation buffer tests, the buffer size range is between 4 KB and 40 MB. The last two problem sizes are set between 40 KB and 400 MB. Time-buffer size results of the aggregation array buffer are given in Fig. 4.4.

The time responses with respect to buffer sizes are similar to the near-field times. For the small (between 4 and 40 KB) and large buffer sizes (between 40 and 400 MB) the times are higher than for the medium buffer sizes. The lowest times are achieved for buffer sizes between 0.4 and 0.7 MB, hence this size range is optimal for the aggregation array.

4.1.3 Radiation/Receiving Patterns Buffer Optimization

The buffer sizes of the radiation/receiving pattern matrix are determined differently than the near-field matrix (array) and aggregation array. The size of the matrix is related to the number of far-field unknowns and the number of phi and theta samplings on the last level. Because the smallest cluster size changes for the 93M-unknowns problem, the buffer size changes as well.

For this part, buffer sizes are kept between 0.13 and 1373 MB. For the first four problem sizes, the buffer size range is between 0.13 and 1373 MB. For the fifth problem size, the buffer is set between 1.3 and 1620 MB. For the last problem, the buffer size ranges between 0.97 and 2900 MB. Time-buffer size results of the radiation/receiving pattern buffer are given in Fig. 4.5.

The total iterative solution times of the buffers used for the near-field matrix, aggregation array and radiation/receiving patterns are given in Fig. 4.6. Although the buffers are not maximally optimized, lower time values with decreased memory space can be obtained and optimization is improved.

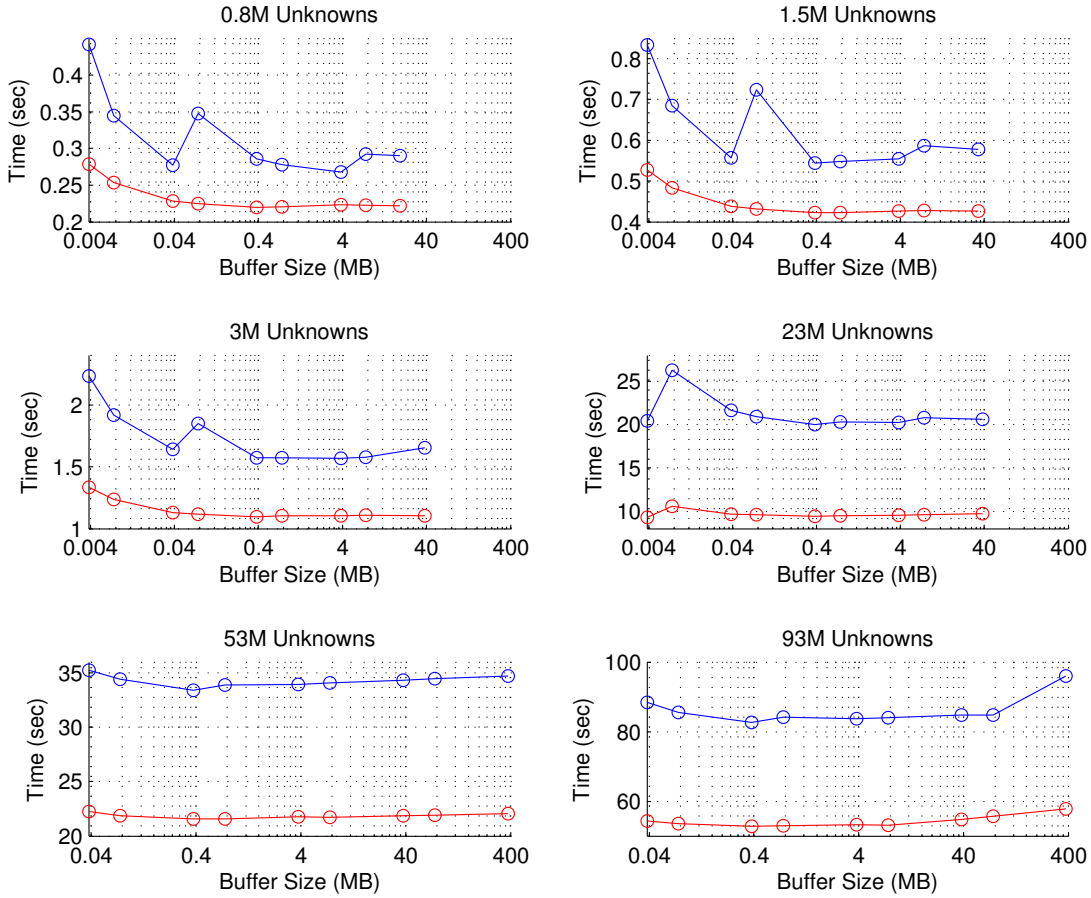


Figure 4.4: The dependence of the CPU time on the size of the aggregation buffer. Results are obtained with HDD (red lines) and SDD (blue lines) separately.

4.1.4 212M-Unknowns Sphere Problem Buffer Optimization

After the optimization tests, a 212M-unknowns problem is solved with 128 processes. We use only SSDs for this test because the total capacity of HDDs in the computation cluster are not sufficient. We skip the near-field matrix calculations and end the tests after the tenth iteration. The buffer size-timing benchmarks and memory-total solution benchmarks are given in Fig. 4.7. The results are similar to previous tests, with the optimal buffer range of the near-field matrix and aggregation array between 0.4 and 0.7 MB and the radiation/receiving pattern buffer between 0.97 and 1.9 MB. Although the optimum of the latter is

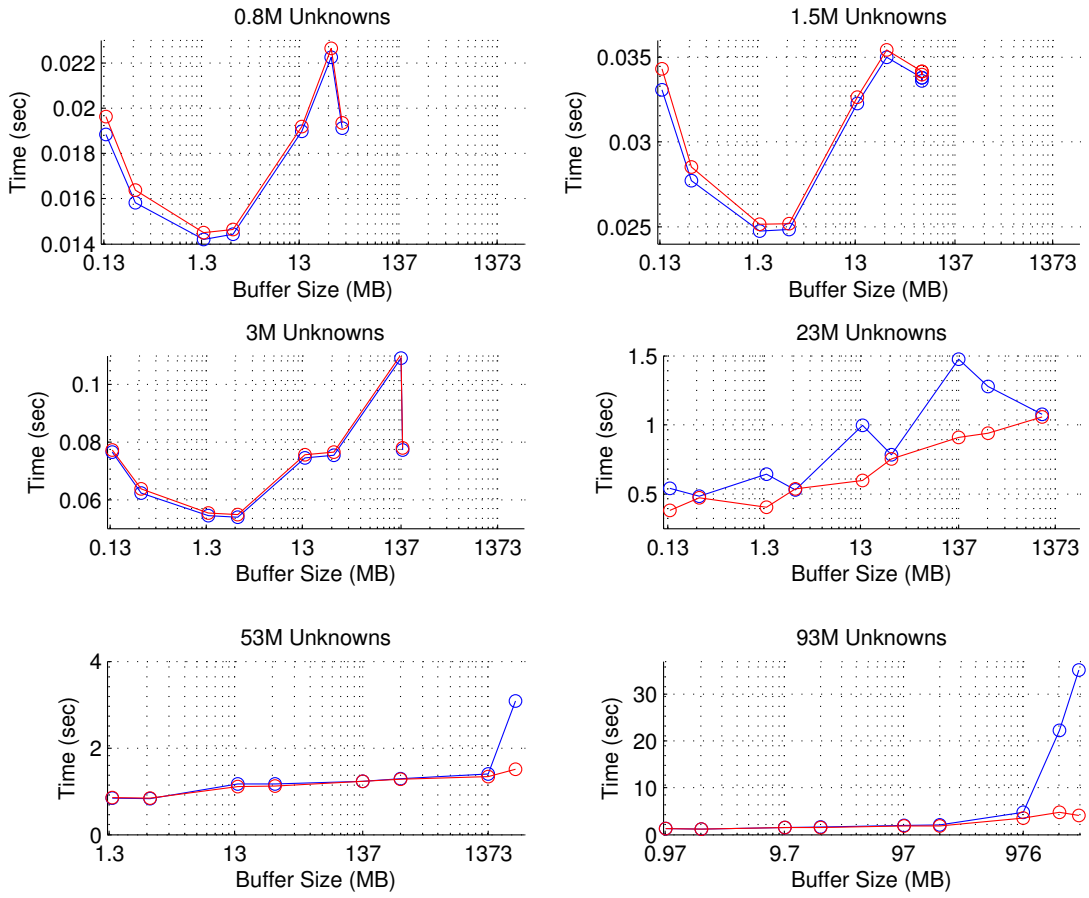


Figure 4.5: The dependence of the CPU time on the size of the radiation/receiving pattern buffer. Results are obtained with HDD (red lines) and SDD (blue lines) separately.

determined to be 1.9 MB, the first three buffer sizes of MLFMA-OC (0.97 MB, 1.9 MB and 9.7 MB) take between 2.3 and 2.7 seconds, whereas MLFMA takes 1.3 seconds. Thus, the third radiation/receiving pattern buffer size still gives a good result.

The optimal buffers (near-field buffer: 0.4 MB; aggregation buffer: 0.4 MB; radiation/receiving pattern buffer: 9.7 MB) for the MLFMA-OC solution takes 7493 seconds using 4072 MB memory. The MLFMA solution takes 4578 seconds using 9738 MB memory. Thus, MLFMA-OC requires 63% more time using 58% less memory than MLFMA to solve a scattering problem involving a sphere with 212M unknowns. This result also shows that the optimal buffer size obtained for

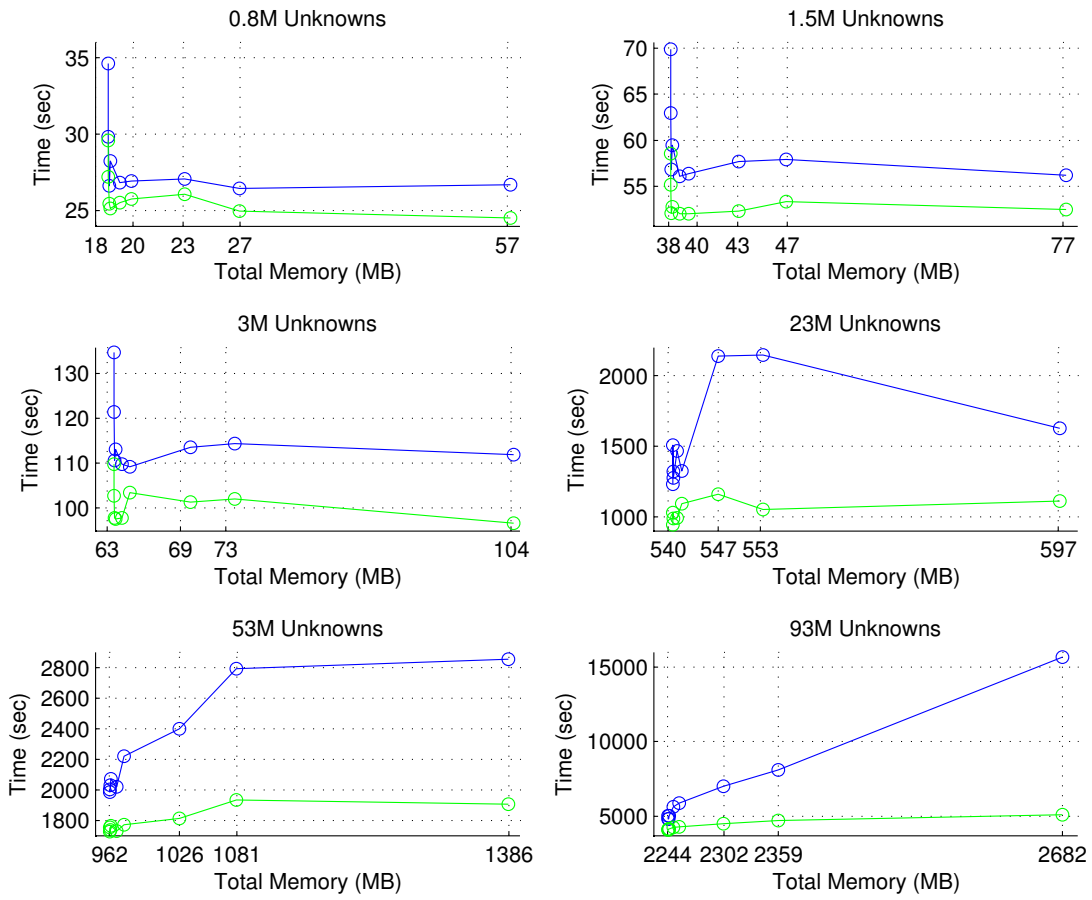


Figure 4.6: The dependence of total iterative solution times on the total memory change for buffer size. Results are obtained with HDD (red lines) and SDD (blue lines) separately.

128 processors is similar with the optimal buffers obtained for 64 processors.

Buffer sizes are determined by modifying the array size and predetermined array sizes used for the optimization benchmark. Therefore, different optimization ranges might be included more-optimal data sizes. We use a 3M-unknowns sphere for this test; the total solution time is given in Fig. 4.6, where it is evident that all optimal buffer sizes are not overlapping. Starting with the original buffer sizes, we change each buffer type to 0.5 and 1.5 times of its original size. The optimization plot is given in Fig. 4.8, where the yellow point is the original test. The near-field buffer is 0.4 MB, the aggregation buffer is 0.4 MB and the radiation/receiving pattern buffer is 13.7 MB. The red points are the near-field

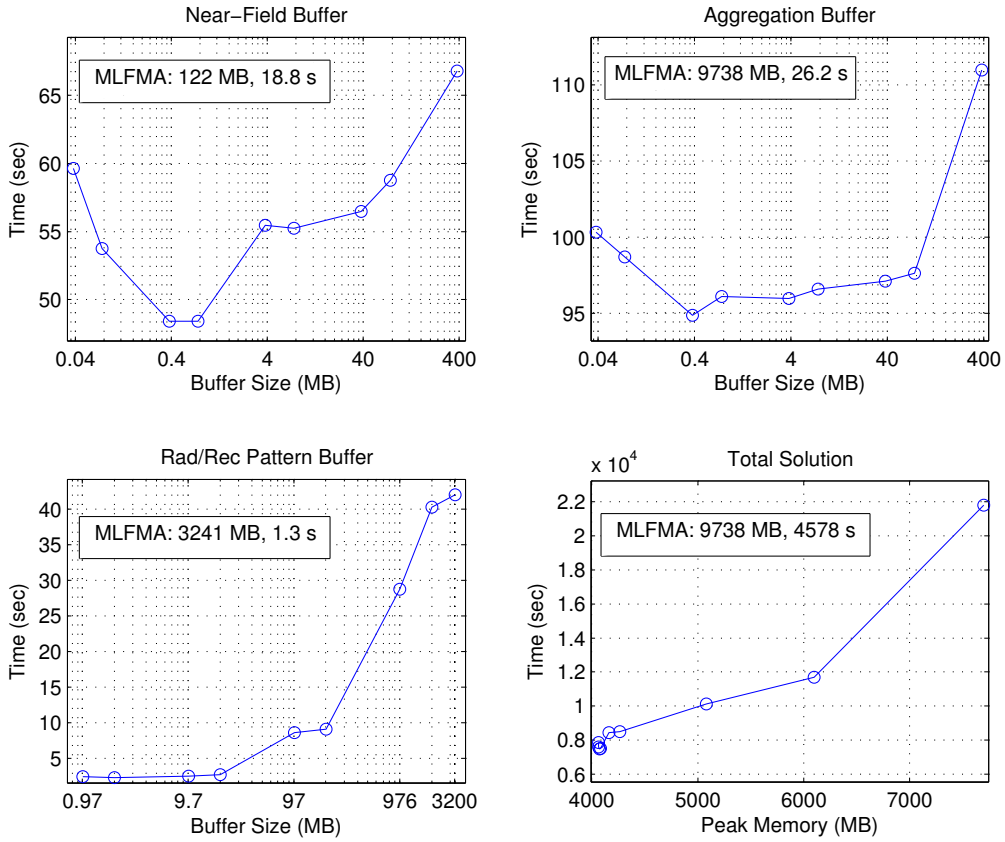


Figure 4.7: OC buffer optimization benchmarks of a 212M-unknowns sphere scattering problem for 128 processes with an SSD.

buffer trials; because this buffer already had a low time, these changes resulted in a time increase. Blue points are the aggregation buffer trials, where the SSD resulted in a slightly lower time for the 0.6 MB buffer size and the HDD resulted in a slightly lower time for the 0.2 MB buffer. We see a major change in the radiation/receiving pattern buffer trials (cyan points). The half-sized buffer (6.8 MB) took less time, and the 1.5 buffer size (20.5 MB) took more time. At this level, we change the aggregation buffers and the radiation/receiving pattern buffers and perform a second test (round black points), which results in improved times. For the third test (square black points), we set the radiation/receiving pattern buffers to 0.65 MB, 1.3 MB and 2.7 MB; the optimal size turned out to be 2.7 MB.

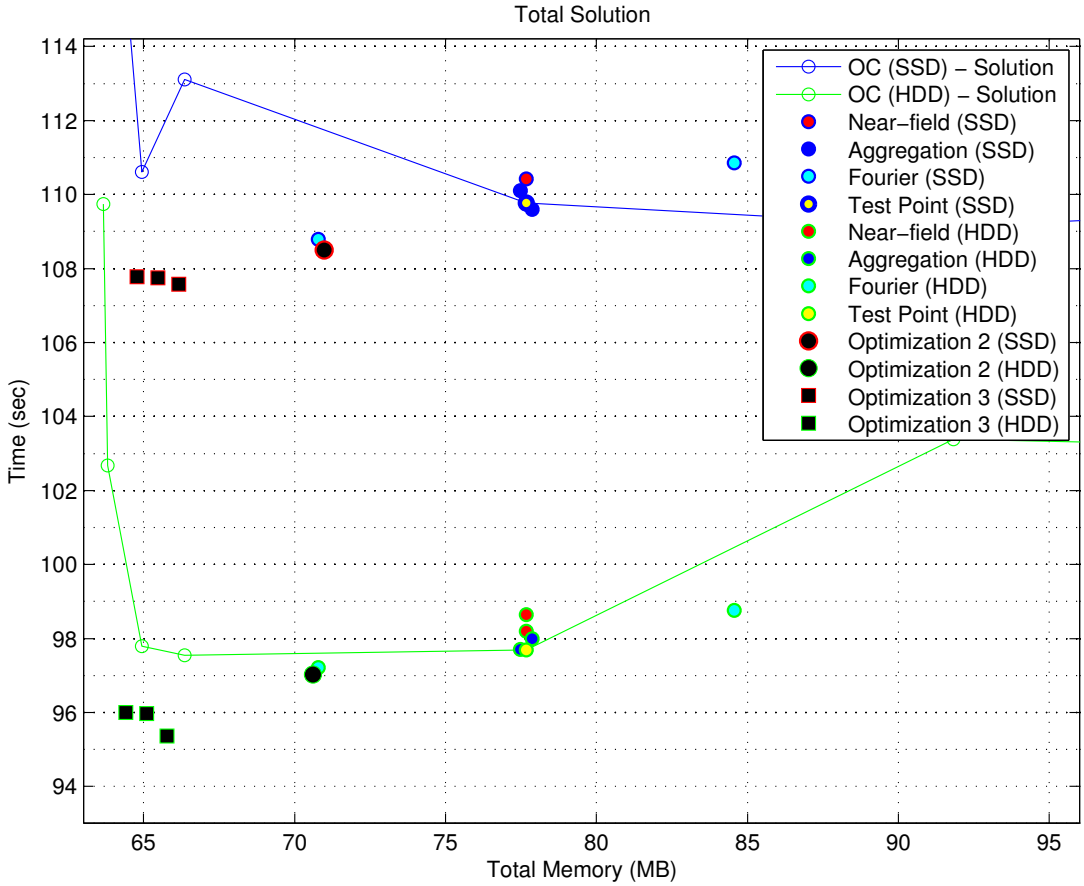


Figure 4.8: Optimization of "total iterative solution times versus total memory" results for a 3M-unknowns sphere scattering problem. Testing HDD (green lines) and SSD (blue lines) separately. For comparison purposes, MLFMA solution requires 308 MB memory and 77.5 seconds total iterative solution time.

4.2 Solution of Sphere and Almond Geometries with Optimized MLFMA-OC

In this section, we solve different sizes of sphere and NASA Almond scattering problems, comparing MLFMA and MLFMA-OC using SSDs. We use the optimal buffer sizes given in the previous section.

The sharp end of the NASA Almond geometry lies on the x-y plane and its sharp edge points in the $+x$ direction. The geometry is illuminated 180° from the x axis on the x-y plane. Thus, the round face of the geometry has been

illuminated. The solutions are handled using 128 processors. The in-core solution represents MLFMA and the OC solution represents MLFMA-OC using multiple SSDs. For each solution, we give the per iteration time and peak memory per processor in Table 4.1 for each problem size.

Table 4.1: Iteration Time and Peak Memory for the NASA Almond Problem

# of Unknowns	Geometry Size (λ)	Time (sec)		Memory (MB)	
		IC	OC SSD	IC	OC SSD
1.5M	84.18	2.0	4.2	128	76
6M	168.36	9.6	14.7	338	180
24M	336.73	35.8	48.9	1252	647
97M	673.46	133.8	173.8	4927	2434

The NASA Almond geometry solutions result in OC method timings of 110%, 53%, 36%, and 30% time delays compared to the MLFMA solutions of spheres with 1.5M, 6M, 24M, and 97M unknowns, respectively. On the other hand, memory reductions for increasing problem sizes are 40%, 47%, 49%, and 51%. Thus, for a large-scale problem, a memory reduction of 50% would only cause a time increase of 30%. Figure 4.9 illustrates bistatic radar cross section (RCS) results of 1.5M and 6M-unknowns NASA Almond geometries. The corresponding sizes of geometries are 84.18λ and 168.36λ, respectively.

We obtain scattering solutions for various sizes of sphere geometries. The problems are solved using 64 processes, comparing MLFMA and MLMFA-OC. For the OC solution, SSDs are used as the secondary storage devices. For each solution, per iteration time and peak memory per processor is given in Table 4.2 for each problem size.

Table 4.2: Iteration Time and Peak Memory for Sphere Problem

# of Unknowns	Geometry Size (λ)	Time (sec)		Memory (MB)	
		IC	OC SSD	IC	OC SSD
0.8M	15	2.0	2.6	79	18
1.5M	20	4.2	5.6	145	38
3M	30	7.7	11.0	308	63
23M	80	73.8	132.1	2278	540
53M	120	176.2	203.0	4243	962
93M	160	357.1	480.9	7223	2244

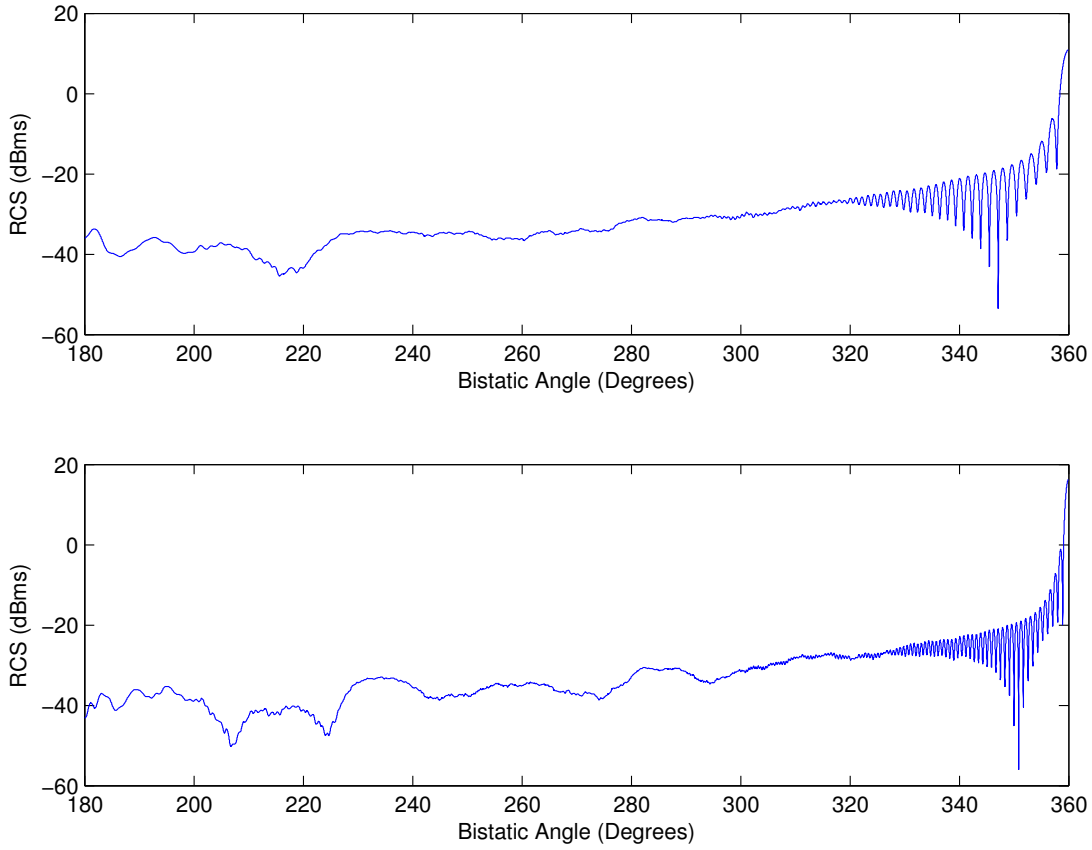


Figure 4.9: RCS on the azimuth plane of a 1.5M and 6M-unknown NASA Almond geometry.

Sphere geometry solutions result in OC method timings of 30%, 33%, 43%, 80%, 15%, and 34% time delays compared to MLFMA solutions of 0.8M, 1.5M, 3M, 23M, 53M, and 93M unknowns, respectively. On the other hand, memory reductions for increasing problem sizes are 78%, 74%, 80%, 77%, 78%, and 69%. Thus, for a large-scale problem, a memory reduction of 70% would only cause a time increase of at most 80%. The results from the sphere and the NASA Almond show that it is possible to reduce the peak memory by half in less than double the solution time.

Last, we test a sphere scattering problem involving 670 million unknowns. The diameter of the sphere is 680λ. We obtain a solution using 128 processes

with 1% residual in 30 iterations. The MLFMA-OC buffers are 0.4 MB for near-field and aggregation, and 0.5 MB for the radiation and receiving patterns. Peak memory usage per processor is 7376 MB and the total solution takes 27.8 hours. The RCS result is given in Fig. 4.10.

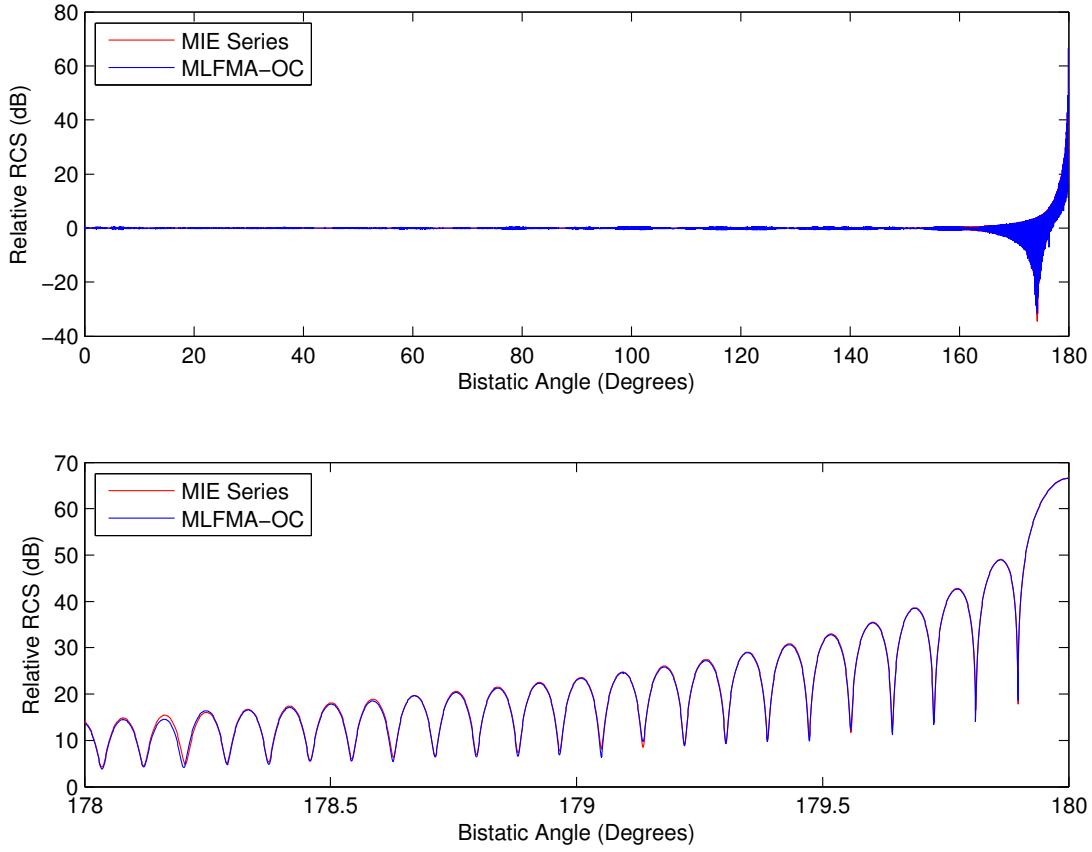


Figure 4.10: RCS of a 670M-unknowns sphere with 680λ diameter.

We also solve this problem with the same residual and input parameters but different buffer sizes. The buffer size for the near-field is 7.6 MB, for aggregation 54 MB and for radiation/receiving patterns 195 MB. We obtain the same results, with a total solution time of 30.1 hours. Compared to the previous solution, more than two hours is saved by selecting more-optimal buffer sizes.

We solve a NASA Almond scattering problem involving 610 million of unknowns. The length of the geometry is 1704λ. We obtain a solution using 128

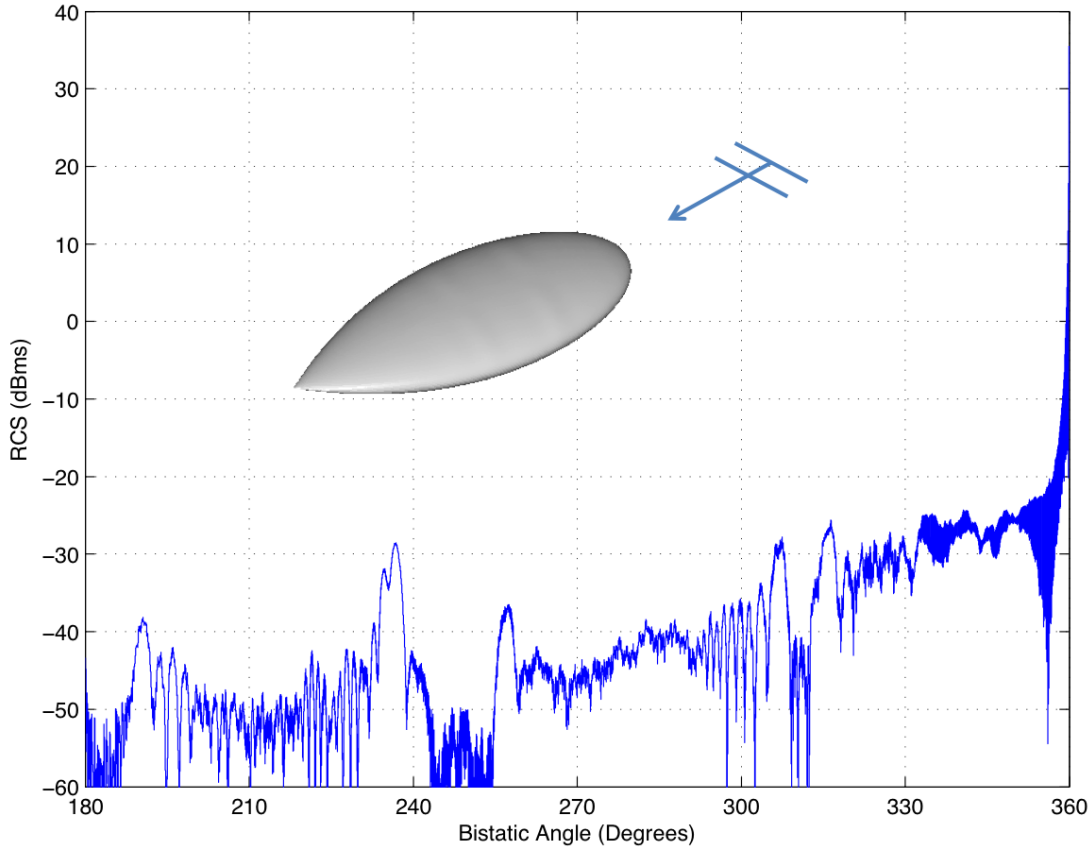


Figure 4.11: RCS on the azimuth plane of a 610M-unknown NASA Almond geometry.

processes with 0.5% residual error in 120 iterations. The sizes of the MLFMA-OC buffers are 0.4 MB for the near-field and aggregation, and 0.5 MB for the radiation and receiving patterns. Peak memory usage per processor is 11886 MB and the total solution takes 92.7 hours. The RCS result of the geometry is given in Fig. 4.11.

Chapter 5

Conclusions

Using a low-complexity algorithm is necessary to solve extremely large-scale scattering problems. MLFMA can handle such problems with computational and memory complexity of $\mathcal{O}(N \log N)$. However, even a low-complexity algorithm may require large amount of memory for challenging real-life problems. One way to reduce memory is to incorporate out-of-core methods into the main algorithm. In this research, we implement an OC method into the parallel MLFMA, and achieve memory reduction without increasing computational complexity. This implementation succeeds through the following steps: investigation of best data type for OC, memory peak detection, OC implementation and OC buffer size tests.

Data types are tested on HDDs and SSDs. Disk types did not result in a significant time difference. However, there was a major time difference between using ASCII formatted and binary data. It was observed that transferring binary data required much less time than ASCII-formatted data. Also, binary data requires less space in the drive. Thus, binary data type is used for the OC implementation.

After selecting the data type, memory profiling is performed. Memory peaks are carefully detected and elements with major sizes are selected: near-field interactions matrix, aggregation array, and radiation/receiving patterns. The first two

elements are fully calculated and used in an OC fashion, while the aggregation array is used partially out-of-core. After we finish the OC implementation, sizes of OC buffers are investigated. We found the optimal buffer-size interval for each OC element, and minimized the OC solution time.

We performed various tests for different sizes of spheres and NASA Almond geometries. Out-of-core implementation reduces the memory usage by almost 50% and the per-iteration solution time approximately becomes 1.5 times the original MLFMA solution. Finally, full-wave solutions of scattering problems are obtained for large sphere and NASA Almond geometries. Solving a sphere scattering problem including 670 million unknowns is achieved using only 966 GB memory and within 30 hours. Solving a NASA Almond scattering problem including 610 million unknowns is achieved using only 1.5 TB memory and within 93 hours.

Future work will include the streamlining of the data read-in and write-out operations in order to increase the performance of the OC implementation of the parallel MLFMA. For example, first-in, first-out strategy may be used for this purpose. The goal will be to protect the processors from getting affected from the data traffic during the disk-read and disk-write operations.

Bibliography

[1] W. C. Reiley and R. A. van de Geijn, "Pooclapack: Parallel out-of-core linear algebra package," Austin, TX, USA, Tech. Rep., 1999.

[2] L. Nyland, M. Harris, and J. Prins, *Fast N−Body Simulation with CUDA*, GPU Gems, 3rd ed.

[3] M. Yuan, T. K. Sarkar, and B. Kolundzija, "Solution of large complex problems in computational electromagnetics using higher-order basis in MoM with out-of-core solvers," *IEEE Trans. Antennas Propag.*, vol. 48, no. 2, pp. 55–62, 2006.

[4] X.-W. Zhao, Y. Zhang, H.-W. Zhang, D. Garcia-Donoro, S.-W. Ting, T. K. Sarkar, and C.-H. Liang, "Parallel MoM-PO method with out-of-core technique for analysis of complex arrays on electrically large platforms," *Prog. Electromagn. Res.*, vol. 108, pp. 1–21, 2010.

[5] G. Sylvand, "Performance of a parallel implementation of the FMM for electromagnetics applications," *Int. J. Numer. Methods Fluids*, vol. 43, no. 8, pp. 865–879, 2003.

[6] J. M. Song and W. C. Chew, "Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering," *Microwave Opt. Tech. Lett.*, vol. 10, no. 1, pp. 14–19, 1995.

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

[8] X.-Q. Sheng, J. M. Jin, J. M. Song, W. C. Chew, and C.-C. Lu, "Solution of combined-field integral equation using multilevel fast multipole algorithm for scattering by homogeneous bodies," *IEEE Trans. Antennas Propag.*, vol. 46, no. 11, pp. 1718–1726, 1998.

[9] Ö. Ergül and L. Gürel, "Improving the accuracy of the magnetic field integral equation with the linear-linear basis functions," *Radio Sci.*, vol. 41, no. 4, 2006.

[10] L. Gürel and Ö. Ergül, "Singularity of the magnetic-field integral equation and its extraction," *IEEE Antennas Wireless Propag. Lett.*, vol. 4, pp. 229–232, 2005.

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

[12] Ö. Ergül and L. Gürel, "Enhancing the accuracy of the interpolations and anterpolations in MLFMA," *IEEE Antennas Wireless Propag. Lett.*, vol. 5, no. 1, pp. 467–470, 2006.

[13] S. Koc, J. Song, and W. C. Chew, "Error analysis for the numerical evaluation of the diagonal forms of the scalar spherical addition theorem," *SIAM J.*, vol. 36, no. 3, pp. 906–921, 1999.

[14] R. Coifman, V. Rokhlin, and S. Wandzura, "The fast multipole method for the wave equation: A pedestrian prescription," *IEEE Antennas Propag. Mag.*, vol. 35, no. 3, pp. 7–12, 1993.