2018 IEEE International Parallel and Distributed Processing Symposium

A Fast and Massively-Parallel Inverse Solver for Multiple-Scattering Tomographic Image Reconstruction

Mert Hidayetoğlu, Carl Pearson, Izzat El Hajj, Levent Gürel, Weng Cho Chew, and Wen-mei Hwu
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

***Abstract*—We present a massively-parallel solver for large Helmholtz-type inverse scattering problems. The solver employs the distorted Born iterative method for capturing the multiple-scattering phenomena in image reconstructions. This method requires many full-wave forward-scattering solutions in each iteration, constituting the main performance bottleneck with its high computational complexity. As a remedy, we use the multilevel fast multipole algorithm (MLFMA). The solver scales among computing nodes using a two-dimensional parallelization strategy that distributes illuminations in one dimension, and MLFMA sub-trees in the other dimension. Multi-core CPUs and GPUs are used to provide per-node speedup. We demonstrate a 76% efficiency when scaling from 64 GPUs to 4,096 GPUs. The paper provides reconstruction of a $204.8\lambda \times 204.8\lambda$ image ($4M$ unknowns) executed on 4,096 GPUs in near-real time (almost 2 minutes). To the best of our knowledge, this is the largest full-wave inverse scattering solution to date, in terms of both image size and computational resources.**

I. Introduction

Real-life imaging applications in medicine, remote sensing, non-destructive testing, and geophysical exploration involve illuminating unknown objects with propagating waves, measuring the scattered field, and performing tomographic reconstructions based on these measurements. The widespread approach for keeping the computational complexity of such reconstructions low is making the simplifying assumption that a wave is scattered only once by the object. Unfortunately, this single-scattering assumption cannot provide accurate images in many imaging scenarios.

The distorted Born iterative method (DBIM) [1] is a full-wave method that is deliberately designed for solving inverse-scattering problems while accounting for all wave phenomena, including multiple scattering. Extensive research has been done on this method in many application areas, including geophysical applications for finding a buried scatterer [2], [3], tomographic reconstructions with ultrasonic waves [4], [5], [6], and reconstructions with electromagnetic waves of microwave [7] and radar [8] frequencies. Accounting for multiple scattering introduces mathematical nonlinearity to the reconstruction. This nonlinearity requires solving many forward-scattering problems. Each forward solution involves a matrix inversion, where direct approaches such as LU decomposition and singular value decomposition have prohibitive, i.e., $\mathcal{O}(N^3)$, computational and storage complexity, where N is the number of pixels. Because of the high computational burden of DBIM, reported results have so far been limited to reconstructions of only tens of wavelengths in size.

The multilevel fast multipole algorithm (MLFMA) is a fast and efficient algorithm that can solve scattering and radiation problems governed by the Helmholtz equation with reduced-complexity [9], [10]. However, the parallelization of MLFMA is not trivial because of its complicated structure. Moreover, the reduced complexity features of MLFMA make it extremely difficult for a parallel implementation to achieve significant speedup, and the challenges differ across application domains. Therefore, a scalable parallel implementation of MLFMA in general, and for imaging in particular, remains to be a challenge when executing on large supercomputers.

In this paper, we present the detailed implementation of a DBIM-based inverse scattering solver parallelized with MPI, OpenMP, and CUDA. To the best of our knowledge, our solver is the first scalable inverse scattering solver for imaging that is fully enhanced with MLFMA. The complexity of our application is $\mathcal{O}(N)$. This low complexity enables us to solve larger scale problems while capturing multiple-scattering effects for more accurate images. We report the achieved scalability and performance on Blue Waters, a massively parallel supercomputer at the National Center for Supercomputing Applications (NCSA). To the best of our knowledge, the results presented herein include the largest full-wave inverse scattering image reconstruction that has been achieved to date and the largest number of processors scaled to for this problem. The imaging domain is $204.8\lambda \times 204.8\lambda$ ($4M$[1] unknowns) and the solution is performed on 4,096 GPUs in near-real time (in almost 2 minutes).

Publicly available resources can be found under Github repository *FFW-Tomo*.

II. Motivation

Conventional diffraction-tomography imaging relies on the Born approximation [11] which assumes that the incident field is scattered only once from the object. This assumption is only accurate when the imaged object is a few wavelengths in size or has low permittivity contrast. It breaks down

[1]Throughout this paper, uses of k and M for enumerating unknowns refer to 2^{10} and 2^{20}, respectively.

1530-2075/18/$31.00 ©2018 IEEE
DOI 10.1109/IPDPS.2018.00017

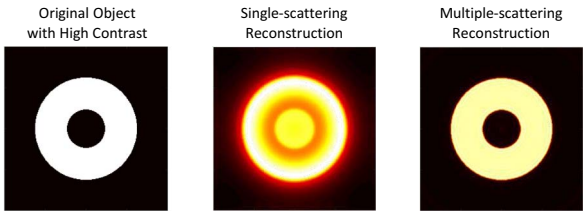

Fig. 1. Reconstruction of high contrast homogeneous annular object using single-scattering (linear) and multiple-scattering (nonlinear) approaches.

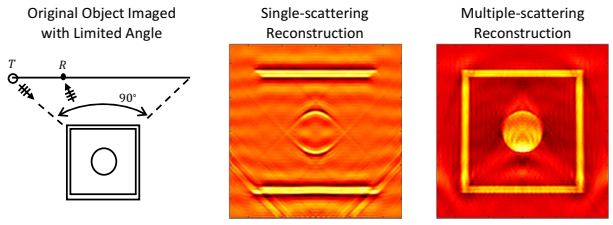

Fig. 2. Reconstruction with transmitters and receivers at a limited angle using single-scattering (linear) and multiple-scattering (nonlinear) approaches. Capturing multiple-scattering waves is critical for detecting the parts of the object whose single-scattering waves move away from the detectors.

when the object is large or has high contrast resulting in strong higher-order scattering effects. These terms can be used to represent the multiple-scattering phenomena. DBIM is a full-wave method that provides a mathematically sound way for capturing these multiple-scattering effects for obtaining accurate images. A simple example is shown in Fig. 1 for how a high contrast image is more accurately reconstructed using nonlinear multiple-scattering reconstruction than using conventional single-scattering reconstruction. Capturing multiple-scattering effects is also useful in cases where transmitters and receivers are exposed to the object at a limited angle, in which case single-scattering waves may be scattered away from the receivers as shown in Fig. 2. More on the limited-angle case can be found in [12].

The main problem with capturing the higher-order nonlinear multiple-scattering terms is the high computational burden they introduce. These terms create the need for a matrix inversion; hence a forward solver, in every iteration which has $\mathcal{O}(N^3)$ complexity with direct approaches such as SVD, QR, or LU decompositions, and $\mathcal{O}(N^2)$ complexity with iterative solutions such as CG or BiCG [13]. Fast multipole methods can solve such problems with $\mathcal{O}(N^{1.4})$ computational complexity [14], for the volumetric problems of interest. MLFMA further decreases the computational complexity from $\mathcal{O}(N^{1.4})$ to $\mathcal{O}(N)$ [9], [10]. Reducing the complexity of multiple-scattering solvers with MLFMA was attempted by Hesford and Chew [15]. However, their approach has limited parallel scalability, solving only a small problem on few computational resources (see Section VII for details).

To the best of our knowledge, our solver is the first scalable inverse scattering solver for imaging that is fully enhanced with MLFMA. In prior work [16], [17], [18], we have given a brief overview of our solver and presented results for a

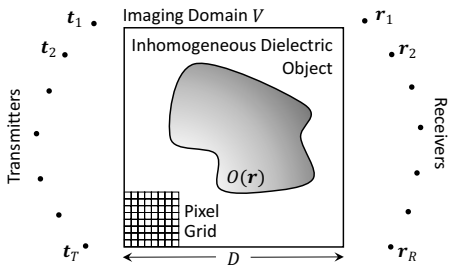

Fig. 3. The application reconstructs the unknown scatterer in the imaging domain. T transmitters illuminate the object. Reconstruction is performed by processing the scattered field collected by R receivers.

$102.4\lambda \times 102.4\lambda$ ($1M$ unknowns) imaging domain scaled to 1,024 CPUs [16], [17] and 256 GPUs [18]. The GPU parallelization is only performed across illuminations, not MLFMA sub-trees, which limits its scalability. In this paper, we present a detailed description of our solver and demonstrate its power on a larger imaging domain of size $204.8\lambda \times 204.8\lambda$ ($4M$ unknowns). We also simultaneously parallelize illuminations and MLFMA sub-trees among GPU nodes, allowing us to scale to 4,096 GPUs. To the best of our knowledge, this is the largest problem solved to date and the largest number of processors scaled to for this problem.

III. Application Description

A. Inverse Scattering Solver

Fig. 3 depicts the geometry of the imaging setup. We assume an inhomogeneous dielectric object. The object is located in the imaging domain V with equal side lengths of D. The object is illuminated by T transmitters, and the scattered field is collected by R receivers. For numerical solutions, V is discretized with N square pixels. To obtain a good resolution of the field, the size of the pixels is selected to be $\lambda/10$, where λ is the wavelength of the illumination wave in free space.

Fig. 4 summarizes the overall workflow of our application. The inverse scattering solver takes the incident field ϕ^{inc} emitted by the transmitters and the scattered field ϕ^{mea} measured by the receivers and uses them to reconstruct the final object O. It begins with a guessed object O_b and iteratively refines it until convergence using the conjugate-gradient optimization.

Each iteration begins with a scattering solver that uses the given incident field ϕ_t^{inc} of a transmitter t (also called an *illumination*) to compute a scattered field ϕ_t^{sca} based on the guessed object. Illuminations from different transmitters can be processed independently. The scattering solver involves solving a forward problem according to equations (E1) and (E2) in Fig. 4. Next, the computed scattered field along with the actually measured scattered field ϕ_t^{mea} are used to compute a partial gradient $\nabla\Phi_t$. Computing the gradient involves solving another forward problem according to equations (E3) and (E4) in Fig. 4. The gradients are then combined from the different illuminations. After that, the gradient is used to compute a step size δO_t. Computing the step size involves solving yet another forward problem according to equations

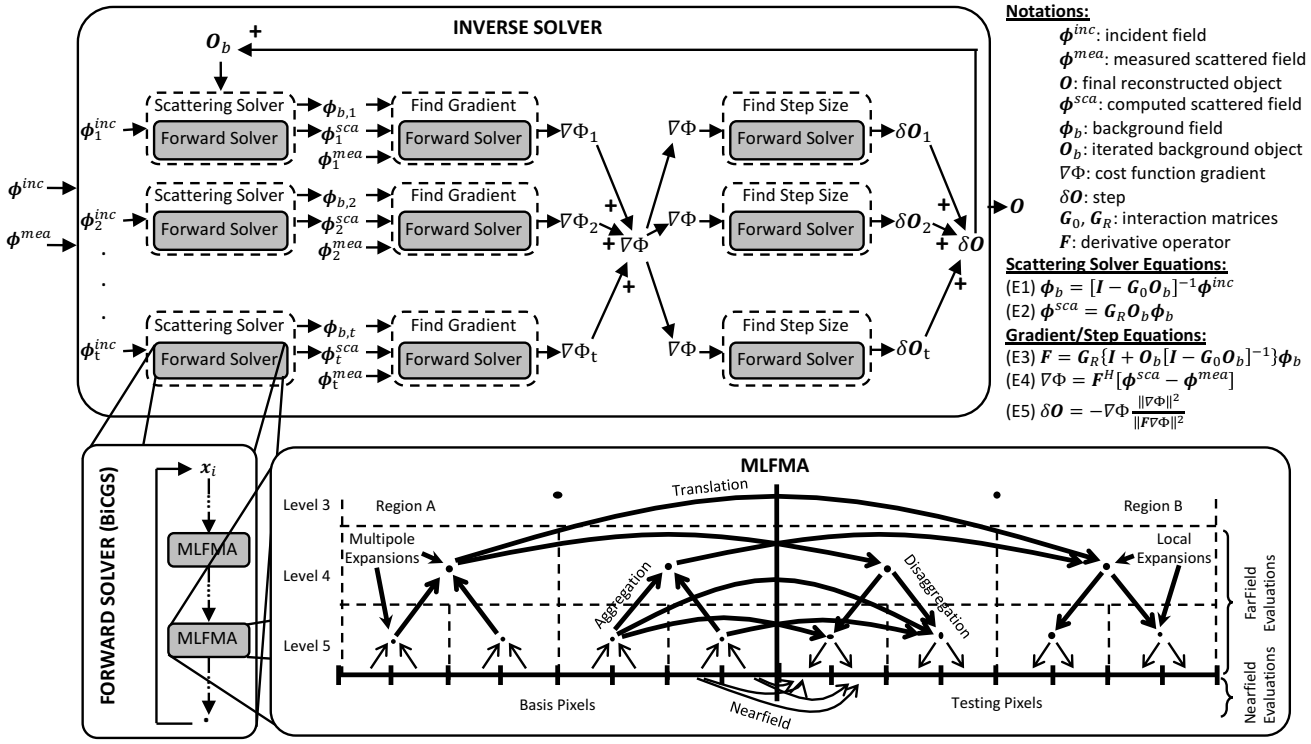

Fig. 4. The overall application workflow. The MLFMA operation dominates performance: it is used twice per iteration of the forward solver, which is itself used three times per transmitter per iteration of the inverse scattering solver.

(E3) and (E5) in Fig. 4. Finally, the steps computed from each illumination are combined and used to update the guessed object for the next inverse scattering solver iteration. Interested readers can refer to Section VI for the detailed formulation of the problem and the derivation of these equations.

The forward solver is the key component of the inverse scattering solver, invoked three times per transmitter per iteration, and dominating the performance of the application. We use the biconjugate gradient stabilized method (BiCGS) [13] for the forward solver, which is itself an iterative method. The dominant operation in BiCGS is a matrix-vector multiplication that occurs twice per iteration. For this operation, we use MLFMA to reduce the computational complexity. Details of BiCGS are omitted in Fig. 4 for brevity.

B. MLFMA

MLFMA provides the foundation for the algorithmic speedup of our solver by enabling matrix-vector multiplications of the pairwise interaction matrix $\boldsymbol{G}_0$ with only $\mathcal{O}(N)$ complexity (see Section III-C). This is achieved by spatially clustering the pixels in the imaging domain in a hierarchical manner, yielding a quad-tree with multiple levels. Groups of pixels are grouped into increasingly large clusters in the quad-tree. Pixel interactions are directly computed only at the lowest level between adjacent clusters. Aggregated cluster interactions are propagated to children and parents.

Cluster interactions may be near field or far field as shown in Fig. 5. The computation of near-field interactions for a cluster involves itself (darkest shade) and the eight adjacent clusters (medium shade) at the same level. Far-field interactions occur between non-near-field children of the parent cluster's near-field clusters. There are eight near-field clusters (light shade thick boxes) for the parent cluster in Fig. 5. Based on this clustering, MLFMA partitions the multiplication as $\overline{\boldsymbol{G}}_0\boldsymbol{x} = \overline{\boldsymbol{G}}_0^{NF}\boldsymbol{x} + \overline{\boldsymbol{G}}_0^{FF}\boldsymbol{x}$. Here, $\overline{\boldsymbol{G}}_0^{NF}$ and $\overline{\boldsymbol{G}}_0^{FF}$ are non-adjacent near-field and far-field matrices, where the former stores the near-field interactions in the lowest-level and the latter represents the rest of the interactions.

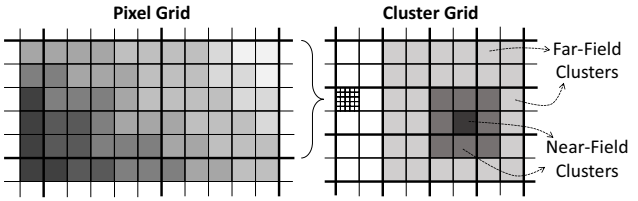

Fig. 5. Left: Pixels (thin boxes) organized in hierarchical clusters (thick boxes). Shades of gray represent changing field intensity discretized over pixels. Right: Thin and thick boxes represent children and parent cluster grids at the lowest two levels of the quad-tree. Every parent cluster has four lower-level child clusters.

MLFMA computes pairwise interactions in four phases: aggregation, translation, disaggregation, and nearfield. These

phases are illustrated at the bottom of Fig. 4. It is important to note that we invoke a Fourier relationship and expand the fields with plane waves, not with multipoles. The plane-wave expansion provides diagonal translation operators, and consequently low complexity. Still, we refer to the expansions as multipole expansions and call the algorithm MLFMA for historical reasons.

The aggregations collect the effect of outgoing fields from children into their parent clusters. The outgoing fields from each cluster are sampled where higher-level (larger) clusters require more samples. Only the lowest level is sampled explicitly. Samples at higher levels are constructed from those in lower levels by interpolating, shifting, and superposing.

The translation step propagates the aggregated outgoing fields to its respective far-field clusters. This is performed by translating the expansion coefficients of each cluster to its respective far-field clusters and superposing the contributions. As a result, each cluster contributes to (and has contributions from) its respective 27 ($6\times6-9$) far-field clusters. The interactions among more distant clusters are translated collectively at higher levels.

The disaggregations propagate the aggregated and translated fields from a cluster to its children. The disaggregation step is the Hermitian transpose of the aggregation step, and like aggregation, higher-level samples are copied, shifted, and anterpolated to produce the lower-level local expansions in its children. The anterpolation operator is a low pass filter followed by downsampling, which corresponds to the adjoint of an interpolation operator. At the lowest level, the local expansions are converted back to incoming fields by taking inverse Fourier transforms.

Finally, the fields from near-field and far-field multiplications are superposed.

C. Computational and Storage Complexity

At the end of the computation described in Section III-B, the field induced by all N pixels will have been evaluated at all N pixels. This computation corresponds to the dense matrix-vector multiplication $\overline{\boldsymbol{G}}_0\boldsymbol{x}$, but with $\mathcal{O}(N)$ computational and storage complexity.

The $\mathcal{O}(N)$ computational complexity is determined as follows. By bounding the computation for each cluster to its 27 far-field clusters, the total work at each level is proportional to the number of samples per cluster times the number of clusters. At the lowest level, the work is $\mathcal{O}(N)$. As the level increases, the number of clusters is divided by four. However, the number of samples per cluster only doubles because it is proportional to the cluster width. The work is thus divided by two at each level. The total work across all levels will therefore remain $\mathcal{O}(N)$ despite there being $\mathcal{O}(\log N)$ levels.

As for storage complexity, the MLFMA tree levels consume memory proportional to the number of samples per cluster times the number of clusters, so the memory usage is similarly $\mathcal{O}(N)$. The dense interaction matrix has actually $\mathcal{O}(N^2)$ storing complexity; for example, for imaging domains with sizes of $102.4\lambda\times102.4\lambda$ (1*M* unknowns) and $409.6\lambda\times409.6\lambda$ (16*M* unkowns) would require interaction matrices of sizes 16TB and 4PB, respectively, with double-precision complex numbers. It is clear that just storing such matrices would constitute a computational bottleneck. Using MLFMA, we multiply $\overline{\boldsymbol{G}}_0$ on-the-fly without storing it explicitly which enables achieving $\mathcal{O}(N)$ storage complexity.

We also make sure that the whole inverse scattering solver has no other step with more than $\mathcal{O}(N)$ computational and storage complexity. Doing so is important to ensure the scalability of the entire application, not just the MLFMA multiplications.

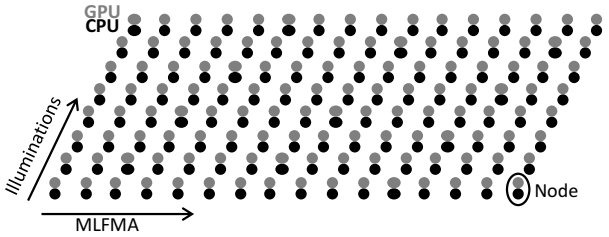

Fig. 6. Two-dimensional parallelization across illuminations and MLFMA sub-trees. Each node employs a GPU for further acceleration.

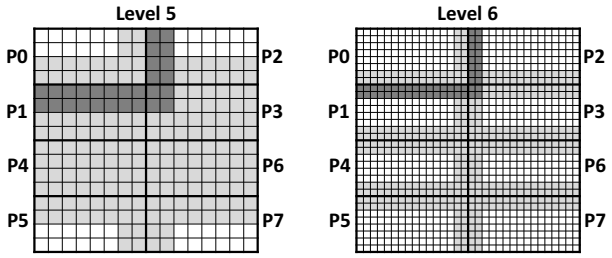

Fig. 7. Parallelization of MLFMA sub-trees across nodes showing cluster grids at two different levels. Translations with inter-process communication are shaded in gray (those of process 0 are shaded in darker gray).

IV. Parallelization and Optimization

A. Parallelization Among Nodes

The parallelization of the application among computational nodes happens in two dimensions as shown in Fig. 6. In the first dimension, we execute different illuminations in parallel on different nodes. Synchronization in this dimension takes place twice per iteration of the inverse scattering solver as shown in Fig. 4. In the second dimension, within each illumination, we parallelize the MLFMA tree structure among nodes. Synchronization in this dimension takes place at each translation and nearfield steps. To the best of our knowledge, our solver is the first to parallelize MLFMA in both dimensions simultaneously while using GPUs.

Efficient parallelization of MLFMA among nodes is achieved by executing the clusters of different sub-trees of the MLFMA tree structure in parallel as shown in Fig. 7. We use Morton indexing to ensure that spatially close clusters are also close in memory. This approach provides good data locality

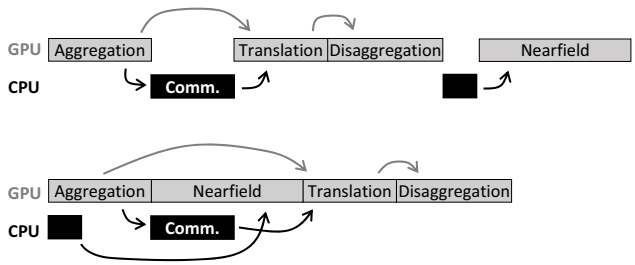

Fig. 8. Overlapping MPI communication with computation. Arrows indicate dependence relationships. While the GPU is handling earlier MLFMA operations, the CPU receives data needed for later operations.

and easy bookkeeping. It also ensures that parent/children clusters across the levels of the tree structure are situated on the same node.

This strategy partitions the MLFMA tree structure to up to 16 processes since there are 16 clusters at the highest computed level. The 16 sub-trees have independent aggregation and disaggregation stages across all levels requiring no MPI communication during these stages. The only MPI communication happens during translation and nearfield steps. Partitioning beyond 16 processes would require splitting aggregation and disaggregation stages at higher levels which would introduce MPI communication to these stages as well.

B. Communication Optimization

When parallelizing a single MLFMA across nodes, MPI communication may be required for the nearfield computation as well as the translation phase of the farfield computation at all levels. For example, Fig. 7 shows for levels 5 and 6 all the far-field clusters involved in communication (light and dark gray), and those involved in communication with P0 specifically (dark gray).

To minimize the number of handshakes, small communication buffers are aggregated into larger ones before communication takes place. We also overlap the communication with computation on each node as shown in Fig. 8. We take advantage of the fact that nearfield and farfield computations are independent to perform the communication for one asynchronously in parallel with the computation of the other.

C. Parallelization Within Nodes

To parallelize an MLFMA sub-tree within nodes, we employ OpenMP threads to process clusters in parallel for levels with many clusters and few samples per cluster, and process samples in parallel for levels with few clusters and many samples per cluster. We extend this parallelization scheme to the GPU which gives us much greater speedup. We apply thread-coarsening, shared-memory tiling, register tiling, and parameter tuning optimization techniques to improve performance. The main factor that enables us to use GPUs effectively is the memory optimizations described in Section IV-D. These optimizations allow interaction matrix operators to fit in GPU memory and formulate these operators as matrix-matrix multiplications which are very suitable for GPU parallelization.

TABLE I
THE KEY MLFMA OPERATORS IN MATRIX FORM

MLFMA Operator	Structure	# Types
Near-Field Interactions	Dense	9
Multipole Expansion	Dense	1
Interpolations	Band-Diagonal	1
Multipole Shiftings	Diagonal	4
Translations	Diagonal	40
Local Shiftings	Diagonal	4
Anterpolations	Band-Diagonal	1
Local Expansions	Dense	1

TABLE II
NCSA BLUE WATERS NODE COMPOSITION

Node	XE6 (CPU)	XK7 (GPU)
Socket 1	AMD Opteron 6276 (8 modules)	AMD Opteron 6276 (8 modules)
Socket 2	AMD Opteron 6276 (8 modules)	NVIDIA Tesla K20x (14 SMX, 6 GB RAM)
RAM	64 GB	32 GB

D. Memory Optimization

The regular grid of pixels and clusters provides opportunities to reuse certain key MLFMA operators listed in Table I. Matrices for these operators are generated ahead of time in the setup stage of the imaging algorithm and stored as lookup tables. This small additional memory cost enables a large reduction in the overall memory demands of MLFMA.

First, the near-field matrix is sparse with $\mathcal{O}(N)$ elements. However, we do not fully store this matrix. Instead, we take advantage of the symmetry in the regular pixel grid to store nine types of key interaction matrices and use them as needed during near-field multiplications.

Second, the far-field matrix is dense with $\mathcal{O}(N^2)$ elements, but it is neither stored nor multiplied explicitly. MLFMA multiplies it on-the-fly using the pre-computed operator matrices through aggregation, translation, and disaggregation steps as explained in Section III-B. The multipole and local expansions are implemented as matrix-matrix multiplications, which achieves better data reuse than matrix-vector multiplications. The interpolation and anterpolation operators, which are used for aggregation and disaggregation, respectively, are realized with band-diagonal matrices. The band-diagonal structure comes from local interpolations of the band-limited expansion samples. More accuracy yields a thicker band (more non-zero elements in the interpolation matrix). The multipole and local shifting operators, which are used for aggregation and disaggregation, respectively, are realized with diagonal matrices. There are four types of these matrices at each level. For the translation step, there are 40 unique types of translation operators at each level which are reused as needed. Translation operators are implemented as diagonal matrices.

V. PERFORMANCE RESULTS

A. Computational Environment

The Blue Waters Supercomputer of NCSA [19] consists of 22,500 XE6 CPU compute nodes and 4,224 XK7 GPU-accelerated nodes. The specifications of the XE6 and XK7

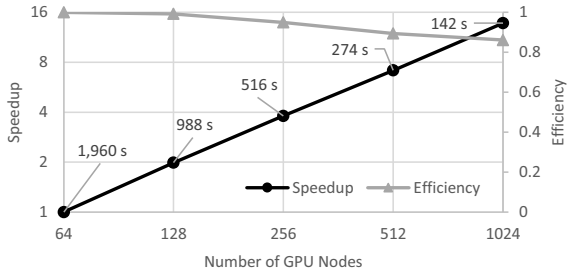

Fig. 9. Strong scaling when illuminations distributed across additional nodes. Each MLFMA solver is employed on a single node.

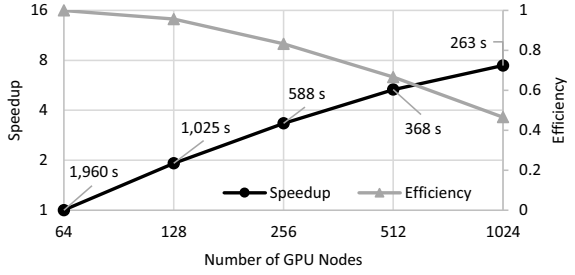

Fig. 10. Strong scaling when MLFMA sub-trees are distributed across additional nodes. Each MLFMA solver is parallelized on multiple nodes.

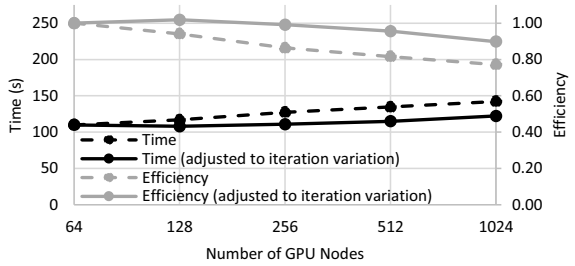

Fig. 11. Weak scaling when number of illuminations are increased with the number of nodes. Each node handles a single illumination.

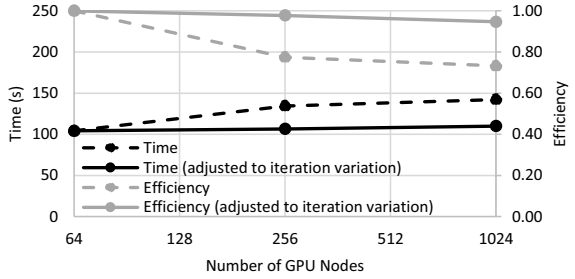

Fig. 12. Weak scaling when solution domain is enlarged with the number of nodes. Each node handles a constant amount of MLFMA sub-tree.

nodes are summarized in Table II. We will refer XE6 nodes as CPU nodes and XK7 nodes as GPU nodes. The CPU and GPU runs employ 16 CPU cores, and 8 CPU cores and a GPU, respectively. The software environment is a 64-bit linux OS. For the numerical results, we use GNU GCC 4.9.3 and Nvidia NVCC 7.5.17 compilers, both with fast-math and O3 optimizations enabled.

B. Solver Parameters

For the numerical results, the MLFMA parameters are chosen such that each matrix-vector multiplication has at most 10^{-5} error, relative to naive direct $\mathcal{O}(N^2)$ multiplication. The forward solutions are terminated when the BiCGS solver converges under 10^{-4} relative residual error norm. The DBIM reconstructions perform 50 nonlinear conjugate-gradient steps. The results use no regularization in the reconstructions except early termination. We terminate heuristically after evaluating the iterative behaviour based on trial and error. All computations use double-precision.

C. Strong Scaling

Since we parallelize the application across two dimensions, illuminations and MLFMA sub-trees, we present two scaling experiments, one in each dimension, for both strong and weak scaling. All strong scaling tests reconstruct the same numerical phantom placed inside a $102.4\lambda \times 102.4\lambda$ imaging domain. The discretization of the domain yields $1M$ unknown pixel properties to be solved. The smallest cluster size is chosen to be $0.8\lambda \times 0.8\lambda$, i.e., each lowest-level cluster involves 64 pixels, yielding a quad-tree structure with eight levels. The phantom is illuminated by 1,024 transmitters and the scattered field is collected by 1,024 receivers.

1) Strong Scaling across Illuminations: In this analysis, the number of GPU nodes is varied from 64 to 1,024 while the problem size is kept the same. Each node is equipped with a single MLFMA solver and 1,024 illuminations are distributed equally across the nodes. The total reconstruction times are shown in Fig. 9.

The reconstruction on 64 GPU nodes, where each node handles 16 illuminations, takes 1,096 s (32.7 min). When scaling to 1,024 nodes, where each node handles a single illumination, the reconstruction takes 142 s (2.4 min). The run with 1,024 GPU nodes is 13.8× faster than than with 64 GPU nodes, yielding 86.1% parallelization efficiency. The efficiency is high due to the mostly-independent handling of illuminations. A notable source of inefficiency is the small difference between the number of iterations in forward solutions, which is averaged-out when the solutions are serialized on nodes.

2) Strong Scaling across MLFMA Sub-trees: In this analysis, we take the 64-node GPU run as a baseline, and distribute the sub-trees of MLFMA across the additional nodes while keeping the problem to be solved the same. For example, when 1,024 nodes are used, there are 64 forward solvers, each parallelized on 16 GPU nodes (see Fig. 6 for depiction), where each solver handles 16 illuminations. The total reconstruction times are shown in Fig. 10.

The reconstruction on 1,024 GPU nodes takes 263 s (4.4 min) which is 7.45× faster than that on 64 GPU nodes, yielding 46.6% parallelization efficiency. The efficiency in this case is lower than that of the previous case. One reason is the

degradation in GPU efficiency due to smaller chunks of work per kernel of the MLFMA kernels. Nevertheless, the GPU runs are still faster than the CPU runs (see Section V-E).

As a strategy for maximizing the overall parallelization efficiency of the application, the illuminations should be partitioned first. If there are more nodes than illuminations, then the MLFMA sub-trees can be partitioned to employ remaining nodes.

D. Weak Scaling

One challenge with the weak scaling analysis for this application is that as the problem size is scaled with the number of processors, the number of BiCGS iterations in forward problems changes creating a disproportional scaling of the problem size. To account for this effect, we show both real execution time/efficiency as well as "adjusted" execution time/efficiency. The latter adjusts the fraction of time spent in the forward solvers to factor out the variation in the number of forward solver iterations. This adjustment is done by dividing the total time spent in BiCGS by the number of BiCGS iterations, then multiplying by the number of BiCGS iterations performed at the smallest problem size with 64 nodes.

1) Weak Scaling across Illuminations: In this analysis, as we increase the number of nodes, we increase the number of illuminations proportionally such that the new nodes are assigned to compute the new illuminations and the number of illuminations per node is held constant. The results of this experiment are shown in Fig. 11. Despite the 77.2% in weak scaling efficiency when real time is considered, we note that after adjusting for iteration variation, the efficiency is 89.9%. This shows that part of the reduction in weak scaling efficiency is largely due to forward solver iteration variation which is a property of the algorithm, and not due to inefficiency in the parallelization of the problem.

2) Weak Scaling across MLFMA Sub-trees: In this analysis, as we increase the number of nodes, we increase the size of the MLFMA tree structure proportionally by increasing the size of the imaging domain, hence the number of pixels. The new nodes are used to process the larger tree structure such that the size of the sub-tree per node is held constant. Note that the scaling factor in this experiment must be a factor of four because of the square dimension of the imaging domain. The results of this experiment are shown in Fig. 12. The real weak scaling efficiency is 73.3% while the adjusted is 94.7%. This result shows that, as with weak scaling of illuminations, the degradation in efficiency is caused largely by the forward solver iteration variation as opposed to inefficient parallelization of the problem.

E. GPU Performance

1) MLFMA Speedup: We evaluate the benefit of GPU acceleration for a single MLFMA multiplication on a large $409.6\lambda \times 409.6\lambda$ ($16M$ unknowns) imaging domain. Table III shows the speedup of individual MLFMA operations and overall speedup. One observation is that 16 GPU nodes are 15.36× faster than 1 GPU node, while 16 CPU nodes are 14.54× faster than 1 CPU node. This better efficiency across GPU nodes is due to overlapping communication by the CPU with computation by the GPU (see Section IV-B). Another observation is that the speedup of the translation step is not as great as that of the other steps because of the low data reuse of the diagonal translation operations (see Table I). In contrast, the multipole and local expansions have the highest speedup because they are implemented as dense matrix-matrix multiplications which are well-suited for GPUs.

TABLE III
INDIVIDUAL MLFMA OPERATIONS GPU SPEEDUPS

MLFMA Operation	1 Node		16 Nodes	
	CPU	GPU	CPU	GPU
Multipole Expansion	1.0×	5.05×	16.30×	79.95×
Aggregation	1.0×	5.92×	15.42×	78.71×
Translation	1.0×	2.90×	12.86×	44.80×
Disaggregation	1.0×	2.82×	13.77×	38.22×
Local Expansion	1.0×	5.48×	15.55×	86.51×
Near-Field Interactions	1.0×	3.92×	15.75×	62.76×
Overall	1.0×	3.91×	14.54×	60.08×

TABLE IV
WHOLE APPLICATION GPU SPEEDUP

Number of Nodes	64 Nodes	256 Nodes	1,024 Nodes	4,096 Nodes
CPU Time	8,216 s	2,107 s	558 s	151 s
GPU Time	1,960 s	516 s	142 s	40.2 s
GPU Speedup	4.19×	4.08×	3.92×	3.77×

2) Whole Application Speedup: We evaluate the benefit of GPU acceleration for the whole application using the same parameters from the strong scaling study in Section V-C. Table IV shows the whole application speedup from using GPU acceleration. We scale up to 1,024 nodes by parallelizing across illuminations, and from 1,024 to 4,096 nodes by parallelizing across MLFMA sub-trees. For the largest scale run on 4,096 GPU nodes, the reconstruction time obtained is 40.2 s (less than a minute!) which corresponds to 76.2% strong scaling efficiency with respect to 64 GPU nodes. We do not scale further than 4,096 nodes because Blue Waters has only 4,224 GPU nodes.

The values computed by the CPU and GPU are similar enough so that the solver in both cases takes the same steps and performs the same number of iterations. The final images in the two cases have a relative difference norm of 7.15×10^{-13}.

F. A Large Reconstruction

We solve a very large inverse-scattering problem using the synthetic Shepp-Logan phantom [20] which represents a head section and is a well-known benchmark geometry for imaging applications. The results for a monochromatic (single-frequency) imaging of the phantom are shown in Fig. 13. The phantom is illuminated by 1,024 distinct transmitters and the scattered field is collected by 2,048 receivers. The image size is $204.8\lambda \times 204.8\lambda$ ($4M$ pixels). The problem is partitioned across 4,096 GPU nodes (1,024 illuminations × 4 MLFMA sub-trees per node). After 50 DBIM iterations, the residual error norm drops down to 2.89×10^{-3}. The total solution time is 126.9 s. The total number of forward scattering problems solved is 153,600 with a total number of 2,054,312 MLFMA multiplications (13.4 MLFMA multiplications per forward solution on average). To the best of our knowledge,

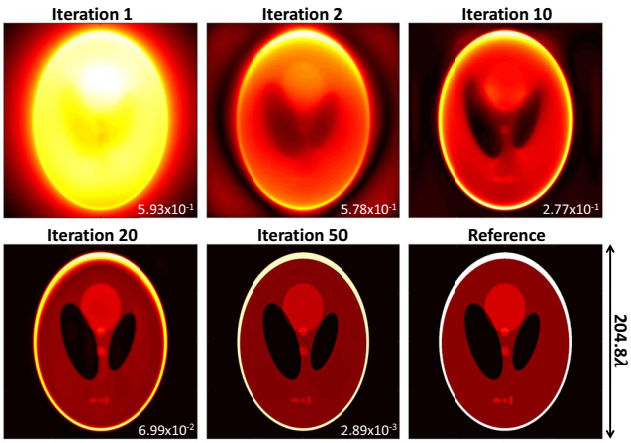

Fig. 13. Reconstruction of a large Shepp-Logan phantom with 0.02 maximum contrast using 1,024 transmitters and 2,048 receivers. The image size is 204.8λ×204.8λ (4M unknowns). We demonstrate that the relative residual norm (of the right-hand-side) goes from 59.3% to 0.03% after 50 iterations. Correspondingly, the image details can be perceived very well.

this is the largest solution achieved to date in terms of object size, number of pixels, and number of processors.

VI. Mathematical Formulation

This section is optional and intended for expert readers.

A. Formulation of Scattering Problems

The Helmholtz equation for inhomogeneous media can be written as $[\nabla^2 + k^2(\boldsymbol{r})]\phi(\boldsymbol{r}) = -q(\boldsymbol{r})$, where $q(\boldsymbol{r})$ is the known illuminating source, $\phi(\boldsymbol{r})$ is the unknown field, and $k(\boldsymbol{r})$ represents the inhomogeneous complex medium. The Helmholtz equation can be reorganized as

$$[\nabla^2 + k_0^2(\boldsymbol{r})]\phi(\boldsymbol{r}) = -q(\boldsymbol{r}) - O(\boldsymbol{r})\phi(\boldsymbol{r}), \tag{1}$$

where $O(\boldsymbol{r}) = k_0^2 \Delta\epsilon_r(\boldsymbol{r})$, k_0 is the background medium, and $\Delta\epsilon_r(\boldsymbol{r})$ is the dielectric permittivity contrast to the background medium. Invoking the free space Green's function satisfying $[\nabla^2 + k_0^2]g_0(\boldsymbol{r}, \boldsymbol{r}') = -\delta(\boldsymbol{r} - \boldsymbol{r}')$, we can convert the differential equation in (1) into an integral equation:

$$\phi(\boldsymbol{r}) = \int_V d\boldsymbol{r}' g_0(\boldsymbol{r}, \boldsymbol{r}')q(\boldsymbol{r}') + \int_V d\boldsymbol{r}' g_0(\boldsymbol{r}, \boldsymbol{r}')O(\boldsymbol{r}')\phi(\boldsymbol{r}') \tag{2}$$

where the first term in the right hand side corresponds to the incident field and the second term corresponds to the scattered field in the imaging domain V (see Fig. 3).

With discretization, we can write (2) in matrix form $\boldsymbol{\phi} = \overline{\boldsymbol{G}}_T\boldsymbol{q} + \overline{\boldsymbol{G}}_0\overline{\boldsymbol{O}}\boldsymbol{\phi}$, where $\overline{\boldsymbol{G}}_0$ and $\overline{\boldsymbol{G}}_T$ are $N\times N$ and $N\times T$ dense matrices, respectively, representing the corresponding free space Green's operator. The incident field on the pixels is defined as $\boldsymbol{\phi}^{inc} = \overline{\boldsymbol{G}}_T\boldsymbol{s}$. The object matrix $\overline{\boldsymbol{O}} = \text{diag}\{\boldsymbol{O}\}$ is diagonal. With a known object, the unknown field can be solved with a matrix inversion

$$\boldsymbol{\phi} = [\overline{\boldsymbol{I}} - \overline{\boldsymbol{G}}_0\overline{\boldsymbol{O}}]^{-1}\boldsymbol{\phi}^{inc}. \tag{3}$$

This corresponds to solving a *forward problem*, where the field satisfying (1) is solved with the known object. Then the scattered field at the transmitters can be found by a $R \times N$ Green's operator $\overline{\boldsymbol{G}}_R$ defined such that $\boldsymbol{\phi}^{sca} = \overline{\boldsymbol{G}}_R\overline{\boldsymbol{O}}\boldsymbol{\phi}$.

For the discretization of (2), we choose the basis and testing (or weighting) functions to be unity on the corresponding pixels, within a Galerkin scheme, and zero elsewhere. The transmitters and receivers are modeled with Dirac delta functions for simplicity. Since each matrix element of $\overline{\boldsymbol{G}}_0$ represents the interaction between a basis and testing function pair, the matrix is called the *interaction matrix*. The matrix elements of the Green's operators can be written explicitly as:

$$\begin{aligned} G_{0\,m,n} &= \int_{V_m} d\boldsymbol{r}' \int_{V_n} d\boldsymbol{r}\, g_0(\boldsymbol{r}', \boldsymbol{r}) \\ G_{R\,r,n} &= \int_{V_n} d\boldsymbol{r}\, g_0(\boldsymbol{r}_r, \boldsymbol{r}) \\ G_{T\,m,t} &= \int_{V_m} d\boldsymbol{r}' g_0(\boldsymbol{r}', \boldsymbol{r}_t) \end{aligned} \tag{4}$$

where V_m and V_n are the domains of the m^{th} and n^{th} pixels, and $\boldsymbol{r}_r$ and $\boldsymbol{r}_t$ are the locations of the r^{th} receiver and t^{th} transmitter, respectively. Furhermore, the diagonal operators have elements $O_n = k_0^2\Delta\epsilon_r(\boldsymbol{r}_n)$ and $I_n = V_n^2$, where $\boldsymbol{r}_n$ is the center of the n^{th} pixel. In two dimensions, the free space Green's function is $g_0(\boldsymbol{r}, \boldsymbol{r}') = i/4H_0^{(1)}(k|\boldsymbol{r} - \boldsymbol{r}'|)$, and an analytical singularity extraction is invoked appropriately for accurate evaluations in (4) with numerical integrations.

B. Iterative Optimization for Inverse Scattering Solutions

The *inverse problem* minimizes the cost function in the form of $\Phi(\boldsymbol{O}) = \|\boldsymbol{\phi}^{sca}(\boldsymbol{O}) - \boldsymbol{\phi}^m\|^2$, where $\boldsymbol{\phi}^{sca}$ and $\boldsymbol{\phi}^m$ are the scattered and measured fields at the receivers, respectively. It is clear that the scattered field is a nonlinear function of the object because $\boldsymbol{\phi}^{sca} = \overline{\boldsymbol{G}}_R\overline{\boldsymbol{O}}[\overline{\boldsymbol{I}} - \overline{\boldsymbol{G}}_0\overline{\boldsymbol{O}}]^{-1}\boldsymbol{\phi}^{inc}$. Therefore we use a gradient-search algorithm to perform this N-dimensional nonlinear optimization. For deriving the gradient (an N-dimensional complex vector) of the cost function, we perturb the object at $\boldsymbol{O}_b$ as $\boldsymbol{O} = \boldsymbol{O}_b + \delta\boldsymbol{O}$ and approximate $\Phi(\boldsymbol{O}_b + \delta\boldsymbol{O}) \approx \|\boldsymbol{\phi}^{sca}(\boldsymbol{O}_b) + \overline{\boldsymbol{F}}\delta\boldsymbol{O} - \boldsymbol{\phi}^m\|^2$, where $\overline{\boldsymbol{F}}$ represents the functional derivative operator. This approximation is second-order accurate. In this case, it is easy to show that the gradient is $\nabla\Phi = \overline{\boldsymbol{F}}^H\boldsymbol{b}$, where $\boldsymbol{b} = \boldsymbol{\phi}^{sca}(\boldsymbol{O}_b) - \boldsymbol{\phi}^m$ is the residual field at the receivers and ^{H} stands for Hermitian transpose. We the exact $\overline{\boldsymbol{F}}$ semi-analytically with the distorted Born approximation, yielding a forward scattering problem.

The derivative operator $\overline{\boldsymbol{F}}$ is updated with the background object in each iteration. For avoiding line search, we analytically fit a quadratic curve to the cost function (in the direction of the negative gradient), of which we take the step at its bottom. With a little math, this step can be shown as

$$\boldsymbol{O} = \boldsymbol{O}_b - \nabla\Phi\frac{\|\nabla\Phi\|^2}{\|\overline{\boldsymbol{F}}\nabla\Phi\|^2}. \tag{5}$$

This step can be implemented with two forward solutions: one for finding the gradient, and one for taking the step. Evaluating the new residual requires an additional solution, yielding three forward solutions per iteration.

The steepest-descent iterations with (5) are naive. To improve convergence, we take steps in conjugate-gradient directions (with a trivial modification). We prefer nonlinear conjugate-gradient iterations because they take fewer total matrix-vector multiplications than Newton-type optimization.

C. Finding Derivative with Distorted-Born Approximation

To find the derivative operator $\overline{\boldsymbol{F}}$, we define a Hilbert space scattering operator $\mathcal{G}$, which solves a scattering problem involving an object $\mathcal{O}$ and a source q such that $\phi = \mathcal{G}q$. Then a scattering equation can be written as $\mathcal{G} = \mathcal{G}_0 + \mathcal{G}_0\mathcal{O}\mathcal{G}$, consistent with (2), where $\mathcal{G}_0$ is the free space Green's operator. Under an object perturbation $\mathcal{O} = \mathcal{O}_b + \delta\mathcal{O}$, the variational equation can be written as $\delta\mathcal{G} = \mathcal{G}_b\delta\mathcal{O}\mathcal{G}_b + \mathcal{G}_b\delta\mathcal{O}\delta\mathcal{G}$, where its first-order term is $\delta\mathcal{G}^{(1)} = \mathcal{G}_b\delta\mathcal{O}\mathcal{G}_b$. Here, $\mathcal{G}_b$ solves the scattering problem involving the background object $\mathcal{O}_b$. When the operators operate on a source, the field variation is $\delta\phi \approx \delta\phi^{(1)} = \mathcal{G}_b\delta\mathcal{O}_b\phi_b$. Since $\delta\mathcal{O}$ is diagonal, we can rearrange and write $\delta\phi^{(1)} = \mathcal{G}_b\phi_b\delta\mathcal{O} = \mathcal{F}\delta\mathcal{O}$, where $\mathcal{F}$ is the functional derivative (Fréchet) operator. The discretization of this yields $\delta\boldsymbol{\phi}^{(1)} = \overline{\boldsymbol{F}}\delta\boldsymbol{O}$, where

$$\overline{\boldsymbol{F}} = \overline{\boldsymbol{G}}_R\{\overline{\boldsymbol{I}} + \overline{\boldsymbol{O}}_b[\overline{\boldsymbol{I}} - \overline{\boldsymbol{G}}_0\overline{\boldsymbol{O}}_b]^{-1}\}\overline{\boldsymbol{\phi}}_b. \tag{6}$$

Multiplications with $\overline{\boldsymbol{F}}$ are costly because of the involved inversions.

VII. Related Work

There have been many works implementing DBIM-based inverse scattering solvers for multiple-scattering reconstruction [1], [2], [3], [4], [5], [6], [7], [8], [21], [22]. These works suffer from high computational complexity which limits their scalability. Our work uses MLFMA for reduced complexity.

Reduced-complexity solutions for DBIM-based inverse solvers for multiple-scattering include CG-FFT [23] and BCGS-FFT [24]. FFT-based solvers have $\mathcal{O}(N \log N)$ complexity. Our work uses MLFMA which has $\mathcal{O}(N)$ complexity.

Hesford and Chew [15] also use MLFMA to implement a reduced complexity DBIM-based inverse scattering solver. Their work is demonstrated on 3-D problems while our work is demonstrated on 2-D problems. However, their approach has limited scalability compared to ours. It performs shared-memory parallelization across CPU cores within a single node while our approach performs MPI parallelization across many nodes, leverages MLFMA sub-tree parallelism for additional scaling to more nodes, and uses GPUs for acceleration within nodes, scaling to 4,096 GPU nodes. The largest full problem they solve is $12.8\lambda\times12.8\lambda\times12.8\lambda$ ($256k$ unknowns) with 72 illuminations while our approach solves $204.8\lambda\times204.8\lambda$ ($4M$ unkowns) with 1,024 illuminations. They show scaling results for $2M$ unknowns without solving the full problem, but these results are obtained using just 2 illuminations and by overdiscretizing a $4\lambda\times4\lambda\times4\lambda$ imaging domain. Note that scaling the number of unknowns by overdiscretizing a small imaging domain only increases work and storage at the lowest level of the MLFMA tree, whereas scaling by enlarging the imaging domain as we do increases work and storage across the entire tree. Although both approaches have the same computational and storage complexity, overdiscretization has a much smaller constant.

Many works use MLFMA to reduce complexity of forward solvers [25], [26], [27], [28], [29], [30]. One work parallelizes the problem on a single GPU [28]. We use many GPUs to parallelize our forward solver and deploy it in a full-fledged inverse solver that performs thousands of forward solutions.

MLFMA has also been used to reduce the complexity of solvers for Helmholtz-type surface problems. There have been many forward solvers proposed [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [28], [46], [47], [48]. A full MLFMA-based inverse solver for surface problems has been proposed by Etminan and Gürel [49]. These solvers have applied various algorithmic optimizations such as exploiting symmetries to save memory and setup time [31], buffering to tune the MLFMA tree structure [32], low-frequency formulations to prevent the MLFMA-breakdown phenomenon [33], [34], and adoption/improvement of numerical interpolation and integration [35], [36]. Parallelization of MLFMA-based forward solvers for surface problems have included MPI-based partitioning of clusters [31], hybrid partitioning of clusters in lower levels and field samples in higher levels [37], partitioning both clusters and samples in middle levels [38], [39], OpenMP parallelization to prevent data duplication via shared memory [40], [41], single GPU parallelization [28], [46], multi-GPU single-node parallelization [47], and multi-GPU multi-node parallelization [48]. We reuse some of these algorithmic optimizations and parallelization strategies while applying MLFMA to a different problem, namely volume problems.

MLFMA has also been used to reduce the complexity of non-Helmholtz problems [50], [51], [52]. Our solver targets Helmholtz-type volume problems.

VIII. Conclusion and Future Work

In this paper, we present the implementation of the first scalable inverse scattering solver for imaging that is fully enhanced with MLFMA, to the best of our knowledge. Employing MLFMA enables us to reduce the complexity of the solver in order to capture multiple-scattering effects producing higher quality images as well as to scale to large problems. Our solver achieves the largest full-wave tomographic image reconstructed to date, which involves an image size of $204.8\lambda\times204.8\lambda$ ($4M$ unknowns), scaled to 4,096 GPU nodes.

Our next steps are to extend this technique to solve three-dimensional imaging domains for which scalability is ever more important for reconstructing meaningfully sized objects. We also plan to apply resonance-free integral formulations and preconditioning of the system to address situations where the problem goes into resonance and near-resonance frequencies which is more likely to happen when problem size is larger.

Acknowledgment

This work was supported by NSF grants OAC-0725070 and EECS-1609195.

REFERENCES

[1] W. C. Chew and Y.-M. Wang, "Reconstruction of two-dimensional permittivity distribution using the distorted Born iterative method," IEEE Transactions on Medical Imaging, vol. 9, no. 2, pp. 218–225, 1990.

[2] T. J. Cui, W. C. Chew, A. A. Aydiner, and S. Chen, "Inverse scattering of two-dimensional dielectric objects buried in a lossy earth using the distorted Born iterative method," IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 2, pp. 339–346, 2001.

[3] F. Li, Q. H. Liu, and L.-P. Song, "Three-dimensional reconstruction of objects buried in layered media using Born and distorted Born iterative methods," IEEE Geoscience and Remote Sensing Letters, vol. 1, no. 2, pp. 107–111, 2004.

[4] R. Lavarello and M. Oelze, "A study on the reconstruction of moderate contrast targets using the distorted Born iterative method," IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 55, no. 1, pp. 112–124, 2008.

[5] R. J. Lavarello and M. L. Oelze, "Tomographic reconstruction of three-dimensional volumes using the distorted Born iterative method," IEEE Transactions on Medical Imaging, vol. 28, no. 10, pp. 1643–1653, 2009.

[6] R. Lavarello and M. Oelze, "Density imaging using a multiple-frequency DBIM approach," IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 57, no. 11, pp. 2471–2479, 2010.

[7] W. H. Weedon, J. E. Mast, W. C. Chew, H. Lee, and J. P. Murtha, "Inversion of real transient radar data using the distorted-Born iterative algorithm," in Antennas and Propagation Society International Symposium, pp. 217–220, 1992.

[8] J. Lin, C. Lu, Y. Wang, W. Chew, J. Mallorqui, A. Broquetas, C. Pichot, and J.-C. Bolomey, "Processing microwave experimental data with the distorted born iterative method of nonlinear inverse scattering," in Antennas and Propagation Society International Symposium, 1993. AP-S. Digest, pp. 500–503, IEEE, 1993.

[9] J. Song, C.-C. Lu, and W. C. Chew, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," IEEE Transactions on Antennas and Propagation, vol. 45, no. 10, pp. 1488–1493, 1997.

[10] J. Song, C. Lu, W. C. Chew, and S. Lee, "Fast illinois solver code (FISC)," IEEE Antennas and Propagation Magazine, vol. 40, no. 3, pp. 27–34, 1998.

[11] M. Born and E. Wolf, Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light. Elsevier, 1980.

[12] M. Hidayetoglu, W.-M. Hwu, and W. C. Chew, "Seeing the invisible: Limited-view imaging with multiple-scattering reconstruction," URSI National Radio Science Meeting (USNC-URSI NRSM), Jan. 2018.

[13] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, vol. 43. SIAM, 1994.

[14] V. Rokhlin, "Diagonal forms of translation operators for the Helmholtz equation in three dimensions," Applied and Computational Harmonic Analysis, vol. 1, no. 1, pp. 82–93, 1993.

[15] A. J. Hesford and W. C. Chew, "Fast inverse scattering solutions using the distorted Born iterative method and the multilevel fast multipole algorithm," The Journal of the Acoustical Society of America, vol. 128, no. 2, pp. 679–690, 2010.

[16] M. Hidayetoğlu, C. Pearson, L. Gürel, W.-m. Hwu, and W. C. Chew, "Scalable parallel DBIM solutions of inverse-scattering problems," in Computing and Electromagnetics International Workshop (CEM), 2017, pp. 65–66, IEEE, 2017.

[17] M. Hidayetoğlu, C. Pearson, I. El Hajj, W. C. Chew, L. Gürel, and W.-m. Hwu, "Scaling analysis of a hierarchical parallelization of large inverse multiple-scattering solutions," SC17: International Conference for High Performance Computing, Networking, Storage and Analysis (poster), 2017.

[18] M. Hidayetoğlu, C. Pearson, W. C. Chew, L. Gürel, and W.-M. Hwu, "Large inverse-scattering solutions with DBIM on GPU-enabled supercomputers," in Applied Computational Electromagnetics Society Symposium-Italy (ACES), 2017 International, pp. 1–2, IEEE, 2017.

[19] NCSA, "Blue Waters user portal — user guide," 2012.

[20] L. A. Shepp and B. F. Logan, "The Fourier reconstruction of a head section," IEEE Transactions Nuclear Science, vol. 21, pp. 21–43, 1974.

[21] F. Gao, B. D. Van Veen, and S. C. Hagness, "Sensitivity of the distorted Born iterative method to the initial guess in microwave breast imaging," IEEE Transactions on Antennas and Propagation, vol. 63, no. 8, pp. 3540–3547, 2015.

[22] D. W. Winters, B. D. Van Veen, and S. C. Hagness, "A sparsity regularization approach to the electromagnetic inverse scattering problem," IEEE transactions on antennas and propagation, vol. 58, no. 1, pp. 145–154, 2010.

[23] A. J. Hesford and W. C. Chew, "A frequency-domain formulation of the Fréchet derivative to exploit the inherent parallelism of the distorted Born iterative method," Waves in Random and Complex Media, vol. 16, pp. 495–508, 2006.

[24] C. Yu, M. Yuan, and Q. H. Liu, "Reconstruction of 3D objects from multi-frequency experimental data with a fast DBIM-BCGS method," Inverse Problems, vol. 25, no. 2, p. 024007, 2009.

[25] C.-C. Lu, "A fast algorithm based on volume integral equation for analysis of arbitrarily shaped dielectric radomes," IEEE transactions on antennas and propagation, vol. 51, no. 3, pp. 606–612, 2003.

[26] K. Sertel and J. L. Volakis, "Multilevel fast multipole method solution of volume integral equations using parametric geometry modeling," IEEE Transactions on Antennas and Propagation, vol. 52, no. 7, pp. 1686–1692, 2004.

[27] H. WallEN, S. Järvenpää, and P. Ylä-Oijala, "Broadband multilevel fast multipole algorithm for acoustic scattering problems," Journal of Computational Acoustics, vol. 14, no. 04, pp. 507–526, 2006.

[28] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency mlfma on graphics processors," IEEE Antennas and Wireless Propagation Letters, vol. 9, pp. 8–11, 2010.

[29] S. Järvenpää, J. Markkanen, and P. Ylä-Oijala, "Broadband multilevel fast multipole algorithm for electric-magnetic current volume integral equation," IEEE Transactions on Antennas and Propagation, vol. 61, no. 8, pp. 4393–4397, 2013.

[30] B. Engquist and L. Ying, "Fast directional multilevel algorithms for oscillatory kernels," SIAM Journal on Scientific Computing, vol. 29, no. 4, pp. 1710–1737, 2007.

[31] S. Velamparambil, W. C. Chew, and J. Song, "10 million unknowns: Is it that big?," IEEE Antennas and Propagation Magazine, vol. 45, no. 2, pp. 43–58, 2003.

[32] S. Ohnuki and W. C. Chew, "Numerical accuracy of multipole expansion for 2D mlfma," IEEE Transactions on Antennas and Propagation, vol. 51, no. 8, pp. 1883–1890, 2003.

[33] H. Wallen and J. Sarvas, "Translation procedures for broadband mlfma," Progress in Electromagnetics Research, vol. 55, pp. 47–78, 2005.

[34] I. Bogaert, J. Peeters, and F. Olyslager, "A nondirective plane wave MLFMA stable at low frequencies," IEEE Transactions on Antennas and Propagation, vol. 56, no. 12, pp. 3752–3767, 2008.

[35] F. P. Andriulli, K. Cools, H. Bagci, F. Olyslager, A. Buffa, S. Christiansen, and E. Michielssen, "A multiplicative Calderon preconditioner for the electric field integral equation," IEEE Transactions on Antennas and Propagation, vol. 56, no. 8, pp. 2398–2412, 2008.

[36] L. Gurel and O. Ergul, "Singularity of the magnetic-field integral equation and its extraction," IEEE Antennas and Wireless Propagation Letters, vol. 4, no. 1, pp. 229–232, 2005.

[37] 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, 2005.

[38] O. Ergul and L. Gurel, "A hierarchical partitioning strategy for an efficient parallelization of the multilevel fast multipole algorithm," IEEE Transactions on Antennas and Propagation, vol. 57, no. 6, pp. 1740–1750, 2009.

[39] L. Gurel and O. Ergul, "Hierarchical parallelization of the multilevel fast multipole algorithm (MLFMA)," Proceedings of the IEEE, vol. 101, no. 2, pp. 332–341, 2013.

[40] X.-M. Pan, W.-C. Pi, M.-L. Yang, Z. Peng, and X.-Q. Sheng, "Solving problems with over one billion unknowns by the MLFMA," IEEE Transactions on Antennas and Propagation, vol. 60, no. 5, pp. 2571–2574, 2012.

[41] M. Hidayetoglu and L. Gurel, "An MPIxOpenMP implementation of the hierarchical parallelization of MLFMA," in Computational Electromagnetics International Workshop (CEM), 2015, IEEE, 2015.

[42] B. Michiels, J. Fostier, I. Bogaert, and D. De Zutter, "Full-wave simulations of electromagnetic scattering problems with billions of unknowns," IEEE Transactions on Antennas and Propagation, vol. 63, no. 2, pp. 796–799, 2015.

[43] L. Gurel, O. Ergul, A. Unal, and T. Malas, "Fast and accurate analysis of large metamaterial structures using the multilevel fast multipole

algorithm," Progress in Electromagnetics Research, vol. 95, pp. 179–198, 2009.

[44] B. MacKie-Mason, A. Greenwood, and Z. Peng, "Adaptive and parallel surface integral equation solvers for very large-scale electromagnetic modeling and simulation," Progress in Electromagnetics Research, vol. 154, pp. 143–162, 2015.

[45] M. Hidayetoglu and L. Gurel, "Parallel out-of-core MLFMA on distributed-memory computer architectures," in Computational Electromagnetics International Workshop (CEM), 2015, IEEE, 2015.

[46] K. Xu, D. Z. Ding, Z. H. Fan, and R. S. Chen, "Multilevel fast multipole algorithm enhanced by GPU parallel technique for electromagnetic scattering problems," Microwave and Optical Technology Letters, vol. 52, no. 3, pp. 502–507, 2010.

[47] 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, 2013.

[48] V. Dang, Q. M. Nguyen, and O. Kilic, "GPU cluster implementation of fmm-fft for large-scale electromagnetic problems," IEEE Antennas and Wireless Propagation Letters, vol. 13, pp. 1259–1262, 2014.

[49] A. Etminan and L. Gürel, "Shape reconstruction of three-dimensional conducting objects via near-field measurements," in Antennas and Propagation Society International Symposium (APSURSI), 2014 IEEE, pp. 153–154, IEEE, 2014.

[50] N. A. Gumerov and R. Duraiswami, "Fast multipole methods on graphics processors," Journal of Computational Physics, vol. 227, no. 18, pp. 8290–8313, 2008.

[51] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji, "42 TFLOPS hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence," in High Performance Computing Networking, Storage and Analysis, pp. 1–12, IEEE, 2009.

[52] R. Yokota, L. A. Barba, T. Narumi, and K. Yasuoka, "Petascale turbulence simulation using a highly parallel fast multipole method on GPUs," Computer Physics Communications, vol. 184, no. 3, pp. 445–455, 2013.