SIAM J. SCI. COMPUT.
Vol. 36, No. 4, pp. C335–C352

A PARALLEL DIRECTIONAL FAST MULTIPOLE METHOD*

AUSTIN R. BENSON†, JACK POULSON‡, KENNETH TRAN§, BJÖRN ENGQUIST¶, AND LEXING YING‖

Abstract. This paper introduces a parallel directional fast multipole method (FMM) for solving N-body problems with highly oscillatory kernels, with a focus on the Helmholtz kernel in three dimensions. This class of oscillatory kernels requires a more restrictive low-rank criterion than that of the low-frequency regime, and thus effective parallelizations must adapt to the modified data dependencies. We propose a simple partition at a fixed level of the octree and show that, if the partitions are properly balanced between p processes, the overall runtime is essentially $\mathcal{O}N \log N/p + p$. By the structure of the low-rank criterion, we are able to avoid communication at the top of the octree. We demonstrate the effectiveness of our parallelization on several challenging models.

Key words. parallel, fast multipole methods, N-body problems, scattering problems, Helmholtz equation, oscillatory kernels, directional, multilevel

AMS subject classifications. 65Y05, 65Y20, 78A45

DOI. 10.1137/130945569

1. Introduction.

This paper is concerned with a parallel algorithm for the rapid solution of a class of N-body problems. Let $\{f_i, 1 \le i \le N\}$ be a set of N densities located at points $\{p_i, 1 \le i \le N\}$ in $\mathbb{R}^3$, with $|p_i| \le K/2$, where $|\cdot|$ is the Euclidean norm and K is a fixed constant. Our goal is to compute the potentials $\{u_i, 1 \le i \le N\}$ defined by

$$u_i = \sum_{j=1}^{N} G(p_i, p_j) \cdot f_j, \tag{1.1}$$

where $G(x, y) = e^{2\pi\iota|x-y|}/|x-y|$ is the Green's function of the Helmholtz equation. We have scaled the problem such that the wavelength equals one and thus high frequencies correspond to problems with large computational domains, i.e., large K.

The computation in (1.1) arises in electromagnetic and acoustic scattering, where the usual partial differential equations are transformed into boundary integral equations (BIEs) [16, 25, 40, 42]. The discretized BIEs often result in large, dense linear systems with $N = O(K^2)$ unknowns, for which iterative methods are used. Equation (1.1) represents the matrix-vector multiplication at the core of the iterative methods.

*Submitted to the journal's Software and High-Performance Computing section November 18, 2013; accepted for publication (in revised form) May 5, 2014; published electronically August 14, 2014. This work was partially supported by National Science Foundation under award DMS-0846501 and by the Mathematical Multifaceted Integrated Capability Centers (MMICCs) effort within the Applied Mathematics activity of the U.S. Department of Energy's Advanced Scientific Computing Research program, under Award number(s) DE-SC0009409.

http://www.siam.org/journals/sisc/36-4/94556.html

†ICME, Stanford University, Stanford, CA 94305 (arbenson@stanford.edu). This author's work was supported by an Office of Technology Licensing Stanford Graduate Fellowship.

‡Department of Mathematics, Stanford University, Stanford, CA 94305 (poulson@stanford.edu).

§Microsoft Corporation, Redmond, WA 98052 (ktran@microsoft.com).

¶Department of Mathematics and ICES, University of Texas at Austin, Austin, TX 78712 (engquist@ices.utexas.edu).

‖Department of Mathematics and ICME, Stanford University, Stanford, CA 94305 (lexing@math.stanford.edu).

1.1. Previous work. Direct computation of (1.1) requires $\mathcal{O}(N^2)$ operations, which can be prohibitively expensive for large values of N. A number of methods reduce the computational complexity without compromising accuracy. The approaches include FFT-based methods [2, 3, 6], local Fourier bases [4, 8], Chebyshev interpolation [28], and the high-frequency fast multipole method (HF-FMM) [9, 13, 33]. Another related development is the butterfly algorithm [18, 29, 30]. We focus on the directional FMM [19, 20, 21, 39]. The directional FMM exploits the numerically low-rank interaction satisfying a *directional parabolic separation condition*, which is fundamentally different than the approach in HF-FMM.

There are several methods for parallel FMM on nonoscillatory kernels [17, 24, 32, 45], and recent work provides a scalable parallel butterfly algorithm [31]. The kernel-independent fast multipole method (KI-FMM) [43] is a central tool for our algorithm in the "low-frequency regime," which we will discuss in subsection 2.3. Many efforts have extended KI-FMM to modern, heterogeneous architectures [11, 12, 32, 44, 46], and several authors have paralellized [22, 23, 37, 41] the multilevel fast multipole algorithm [15, 35, 36], which is a variant of the HF-FMM.

1.2. Contributions. This paper provides the first parallel algorithm for computing (1.1) based on the directional FMM algorithm. The top of the octree imposes a well-known bottleneck in parallel FMM algorithms for the low-frequency case [44]. A key advantage of our approach is that the interaction lists of boxes in the high-frequency regime with width greater than or equal to $2\sqrt{K}$ are empty. This alleviates the bottleneck, as no translations are needed at those levels of the octree. However, the directional interactions between points in the high-frequency regime are complicated, and this makes parallelization challenging.

Our parallel algorithm is based on a partition of the boxes at a fixed level in the octree amongst the processes (see subsection 3.1). Due to the structure of directional FMM in the high-frequency regime, we can leverage the partition into a flexible parallel algorithm. The parallel algorithm is described in section 3, and the results of several numerical experiments are in section 4. As will be seen, the algorithm exhibits good strong scaling up to 1024 processes for several challenging models.

2. Preliminaries. In this section, we review the directional low-rank property of the Helmholtz kernel and the sequential directional algorithm for computing (1.1) [19]. Proposition 2.1 in subsection 2.2 shows why the algorithm can avoid computation at the top levels of the octree. We will use this to develop the parallel algorithm in section 3.

2.1. Directional low-rank property. Let Y be a ball of radius $r \geq \sqrt{3}$ centered at a point c, and define the wedge $X = \{x : \theta(x, \ell) \leq 1/r, |x - c| \geq r^2\}$, where ℓ is a unit vector and $\theta(x, \ell)$ is the angle between the vectors x and ℓ. The sets Y and X are said to satisfy the *directional parabolic separation condition*, which is illustrated in Figure 1.

Suppose we have a set of charges $\{f_i\}$ located at points $\{y_i\} \subseteq Y$. Given a threshold ϵ, there exist two sets of points $\{b_q\}_{1 \leq q \leq r_\epsilon}$ and $\{a_p\}_{1 \leq p \leq r_\epsilon}$ and a set of constants $\{d_{pq}\}_{1 \leq p,q \leq r_\epsilon}$ such that, for $x \in X$,

$$\left| \sum_i G(x, y_i) f_i - \sum_q G(x, b_q) \left(\sum_p d_{qp} \sum_i G(a_p, y_i) f_i \right) \right| = \mathcal{O}(\epsilon), \tag{2.1}$$

where r_ϵ is a constant that depends only on ϵ. A random sampling method based on the pivoted QR factorization [21] can be used to determine this approximation. We

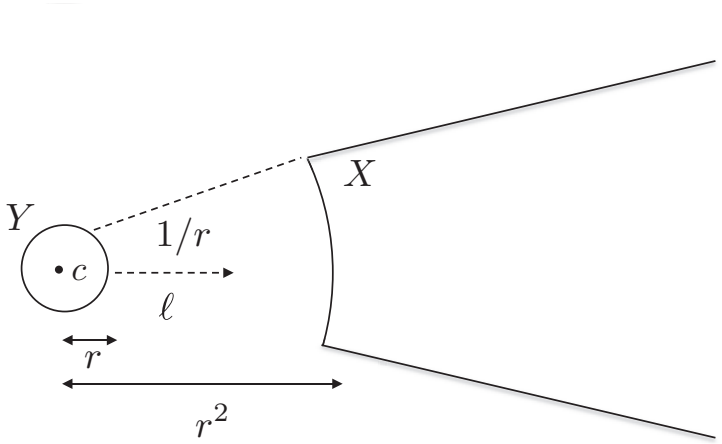

FIG. 1. *Sets Y and X that satisfy the directional parabolic separation condition.*

can, of course, reverse the roles of the ball and the wedge. In (2.1), the quantities $\{\sum_p d_{qp} \sum_i G(a_p, y_i) f_i\}$, $\{b_q\}$, $\{\sum_i G(b_q, x_i) f_i\}$, and $\{a_p\}$ are called the *directional outgoing* equivalent charges, equivalent points, check potentials, and check points of Y in direction ℓ. When the roles of X and Y are reversed, $\{\sum_q d_{qp} \sum_i G(b_q, x_i) f_i\}$, $\{a_p\}$, $\{\sum_i G(b_q, x_i) f_i\}$, $\{b_q\}$ are called the *directional incoming* equivalent charges, equivalent points, check potentials, and check points of Y in direction ℓ. The constants d_{pq} form a matrix $D = (d_{p,q})$ that we call a *translation matrix*.

2.2. Definitions. Without loss of generality, we assume that the domain width is $K = 2^{2L}$ for a positive integer L (recall that $|p_i| \leq K/2$ in (1.1)). The central data structure in the sequential algorithm is an octree. The parallel algorithm, described in section 3, uses the same octree. For the remainder of this paper, let B denote a box in the octree and $w(B)$ the width of B. We say that a box B is in the high-frequency regime if $w(B) \geq 1$ and in the low-frequency regime if $w(B) < 1$. As discussed in [19], the octree is nonadaptive in the nonempty boxes of the high-frequency regime and adaptive in the low-frequency regime. In other words, a box B is partitioned into children boxes if B is nonempty and in the high-frequency regime or if the number of points in B is greater than a fixed constant and in the low-frequency regime.

For each box B in the high-frequency regime and each direction ℓ, we use the following notation for the relevant data:

- $\{y_k^{B,o,\ell}\}$, $\{f_k^{B,o,\ell}\}$, $\{x_k^{B,o,\ell}\}$, and $\{u_k^{B,o,\ell}\}$ are the *outgoing directional* equivalent points, equivalent charges, check points, and check potentials, and
- $\{y_k^{B,i,\ell}\}$, $\{f_k^{B,i,\ell}\}$, $\{x_k^{B,i,\ell}\}$, and $\{u_k^{B,i,\ell}\}$ are the *incoming directional* equivalent points, equivalent charges, check points, and check potentials.

In the low-frequency regime, the algorithm uses the kernel-independent FMM [43]. For each box B in the low-frequency regime, we use the following notation for the relevant data:

- $\{y_k^{B,o}\}$, $\{f_k^{B,o}\}$, $\{x_k^{B,o}\}$, and $\{u_k^{B,o}\}$ are the *outgoing* equivalent points, equivalent charges, check points, and check potentials, and

- $\{y_k^{B,i}\}$, $\{f_k^{B,i}\}$, $\{x_k^{B,i}\}$, and $\{u_k^{B,i}\}$ are the *incoming* equivalent points, equivalent charges, check points, and check potentials.

Due to the translational and rotational invariance of the relevant kernels, the (directional) outgoing and incoming equivalent and check points can be efficiently precomputed since they only depend upon the problem size, K, and the desired accuracy.[1] We discuss the number of equivalent points and check points in subsection 4.1.

Let B be a box with center c and $w(B) \geq 1$. We define the near field of a box B in the high-frequency regime, N^B, as the union of boxes A that contain a point in a ball centered at c with radius $\sqrt{3}w/2$. Such boxes are too close to satisfy the directional parabolic condition described in subsection 2.1. The far field of a box B, F^B, is simply the complement of N^B. The interaction list of B, I^B, is the set of boxes $N^P \backslash N^B$, where P is the parent box of B.

To exploit the low-rank structure of boxes separated by the directional parabolic condition, we need to partition F^B into wedges with a spanning angle of $\mathcal{O}(1/w)$. To form the directional wedges, we use the construction described in [19]. The construction has a hierarchical property: for any directional index ℓ of a box B, there exists a directional index ℓ' for boxes of width $w(B)/2$ such that the ℓth wedge of B is contained in the ℓ'th wedge of B's children. Figure 2 illustrates this idea. In order to avoid boxes on the boundary of two wedges, the wedges are overlapped. Boxes that would be on the boundary of two wedges are assigned to the wedge that comes first in an arbitrary lexicographic ordering.

2.3. Sequential algorithm. In the high-frequency regime, the algorithm uses directional M2M, M2L, and L2L translation operators [19], similar to the standard FMM [14]. The high-frequency directional FMM operators are as follows:

- HF-M2M constructs the outgoing directional equivalent charges of B from the outgoing equivalent charges of B's children,
- HF-L2L constructs the incoming directional check potentials of B's children from the incoming directional check potentials of B, and
- HF-M2L adds to the incoming directional check potentials of B due to outgoing directional charges from each box $A \in I^B$.

The following proposition shows that at the top levels of the octree, the interaction lists are empty.

PROPOSITION 2.1. *Let B be a box with width w. If $w \geq 2\sqrt{K}$, then N^B contains all nonempty boxes in the octree.*

Proof. Let c be the center of B. Then for any point $p_i \in B$, $|p_i - c| \leq \sqrt{3}w/2$. For any point p_j in a box A, $|p_j - c| \leq |p_j| + |p_i| + |p_i - c| \leq K + \sqrt{3}w/2$. N^B contains all boxes A such that there is an $x \in A$ with $|x - c| \leq 3w^2/4$. Thus, N^B will contain all nonempty boxes if $3w^2/4 \geq \sqrt{3}w/2 + K$. This is satisfied for $w \geq \sqrt{4K/3 + 1/3} + \sqrt{1/3}$. Since $K = 2^{2L}$, this simplifies to $w \geq 2\sqrt{K}$. ∎

Thus, if $w(B) \geq 2\sqrt{K}$, no HF-M2L translations are performed ($I^B = \emptyset$). As a consequence, no HF-M2M or HF-M2L translations are needed for those boxes. This property of the algorithm is crucial for the construction of the parallel algorithm that we present in section 3.

We denote the corresponding low-frequency translation operators as LF-M2M, LF-L2L, and LF-M2L. We also denote the interaction list of a box in the low-frequency

[1] Our current hierarchical wedge decomposition does not respect rotational invariance, and so, in some cases, this precomputation can be asymptotically more expensive than the subsequent directional FMM.

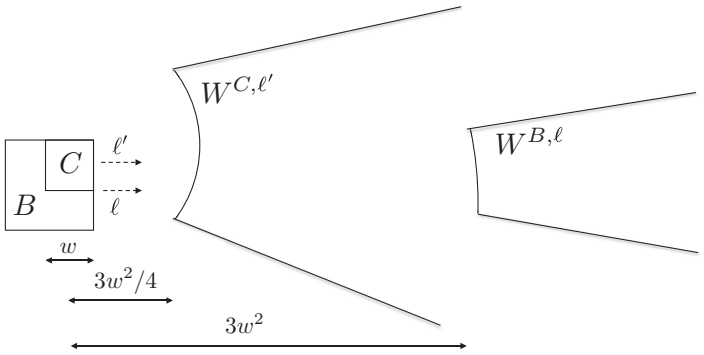

FIG. 2. *A two-dimensional slice of the three-dimensional hierarchical wedges.* $W^{B,\ell}$ *is the wedge for box* B *for the directional index* ℓ*. By the hierarchical construction, we can find an index direction* ℓ' *such that for any child box* C *of* B*,* $W^{B,\ell}$ *is contained in* $W^{C,\ell'}$*, the* ℓ'*th wedge of* C*. Since* C *has a smaller width (*w*) than* B *(*$2w$*), the wedge for* C *is closer and wider than the wedge for* B*. This is a consequence of the directional parabolic separation condition described in subsection* 2.1.

regime by I^B, although the definition of N^B is different [19]. Algorithm 1 presents the sequential algorithm in terms of the high- and low-frequency translation operators.

3. Parallelization. The overall structure of our proposed parallel algorithm is similar to the sequential algorithm. A partition of the boxes at a fixed level in the octree governs the parallelism, and this partition is described in subsection 3.1. We analyze the communication, computation, and spatial complexities in subsections 3.4, 3.5, and 3.6. For the remainder of this paper, we use p to denote the total number of processes.

3.1. Partitioning the octree. To parallelize the computation, we partition the octree at a fixed level. Suppose we number the levels of the octree such that the lth level consists of a $2^l \times 2^l \times 2^l$ grid of boxes of width $K \cdot 2^{-l}$. By Proposition 2.1, there are no dependencies between boxes with width strictly greater than $\sqrt{K}$. The partition assigns a process to every box B with $w(B) = \sqrt{K}$, and we denote the partition by $\mathcal{P}$. In other words, for every nonempty box B at level L in the octree (recall that $K = 2^{2L}$), $\mathcal{P}(B) \in \{0, 1, \ldots, p-1\}$. We refer to level L as the *partition level*. Process $\mathcal{P}(B)$ is responsible for all computations associated with box B and the descendent boxes of B in the octree, and so we also define $\mathcal{P}(A) \equiv \mathcal{P}(B)$ for any box $A \subset B$. An example partition is illustrated in Figure 3.

Algorithm 1: Sequential directional FMM for (1.1).

Data: points $\{p_i\}_{1\le i\le N}$, densities $\{f_i\}_{1\le i\le N}$, and octree $\mathcal{T}$
Result: potentials $\{u_i\}_{1\le i\le N}$

foreach *box* B *in* postorder *traversal of* $\mathcal{T}$ *with* $w(B) < 1$ **do**
  foreach *child box* C *of* B **do**
    Transform $\{f_k^{C,o}\}$ to $\{f_k^{B,o}\}$ via LF-M2M

foreach *box* B *in* postorder *traversal of* $\mathcal{T}$ *with* $1 \le w(B) \le \sqrt{K}$ **do**
  foreach *direction* ℓ *of* B **do**
    Let the ℓ'th wedge of B's children contain the ℓth wedge of B
    foreach *child box* C *of* B **do**
      Transform $\{f_k^{C,o,\ell'}\}$ to $\{f_k^{B,o,\ell}\}$ via HF-M2M

foreach *box* B *in* preorder *traversal of* $\mathcal{T}$ *with* $1 \le w(B) \le \sqrt{K}$ **do**
  foreach *direction* ℓ *of* B **do**
    foreach *box* $A \in I^B$ *in direction* ℓ **do**
      Let ℓ' be the direction for which $B \in I^A$
      Transform $\{f_k^{A,o,\ell'}\}$ via HF-M2L and add result to $\{u_k^{B,i,\ell}\}$
    foreach *child box* C *of* B **do**
      Transform $\{u_k^{B,i,\ell}\}$ to $\{u_k^{C,i,\ell}\}$ via HF-L2L

foreach *box* B *in* preorder *traversal of* $\mathcal{T}$ *with* $w(B) < 1$ **do**
  foreach $A \in I^B$ **do**
    Transform $\{f_k^{A,o}\}$ via LF-M2L and add the result to $\{u_k^{B,i}\}$
  Transform $\{u_k^{B,i}\}$ via LF-L2L
  if B *is a leaf box* **then**
    foreach $p_i \in B$ **do**
      Add result to u_i
  else
    foreach *child box* C *of* B **do**
      Add result to $\{u_k^{C,i}\}$

There are two reasons for using this partitioning. First, the partition leads to simple communication patterns (see the following subsection), which results in small communication time for practical problems (see section 4). Second, with minimal communication, running time is dominated by the translation computations. In subsection 3.5, we show that as long as we can partition boxes evenly at the partition level, the work is balanced. There are more sophisticated process distributions in similar tree computations [26], but the communication costs and scalability for directional FMM on nonuniform geometries are unclear.

3.2. Communication. We assume that the ownership of point and densities conforms to the partition $\mathcal{P}$, i.e., each process owns exactly the points and densities needed for its computations. In practice, arbitrary point and density distributions can be handled with a simple preprocessing stage. Our later experiments make use of k-means clustering [27] to distribute the points to processes. This clustering can be

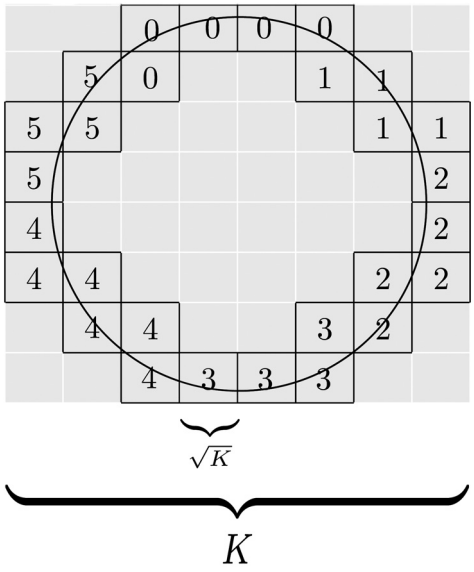

FIG. 3. *Two-dimensional slice of the octree at the partition level for a spherical geometry with each box numbered with the process to which it is assigned. The width of the boxes at the partition level is $\sqrt{K}$, which ensures that the interaction lists of all boxes above this partition level are empty. We note that the number of boxes with surface points (black border) is much smaller than the total number of boxes.*

used as a heuristic for creating $\mathcal{P}$ (see subsection 4.2).

The partition $\mathcal{P}$ dictates the communication in our algorithm. Since $\mathcal{P}(C) = \mathcal{P}(B)$ for any any child box C of B, there is no communication for the L2L or M2M translations in both the low-frequency and high-frequency regimes. These translations are performed in serial on each process. All communication is induced by the M2L translations. In particular, for every box $A \in I^B$ for which $\mathcal{P}(A) \neq \mathcal{P}(B)$, process $\mathcal{P}(B)$ needs outgoing (directional) equivalent densities of box A from process $\mathcal{P}(A)$. We define $\mathcal{I}^B = \{A \in I^B | \mathcal{P}(A) \neq \mathcal{P}(B)\}$, i.e., the set of boxes in B's interaction list that do not belong to the same process as B. The parallel algorithm performs two bulk communication steps:

1. After the HF-M2M translations are complete, all processes exchange outgoing directional equivalent densities. The interactions that induce this communication are illustrated in Figure 4. For every box B in the high-frequency regime and for each box $A \in \mathcal{I}^B$, process $\mathcal{P}(B)$ requests $f^{A,o,\ell}$ from process $\mathcal{P}(A)$, where ℓ is the index of the wedge for which $B \in I^A$.
2. After the HF-L2L translations are complete, all processes exchange outgoing nondirectional equivalent densities. For every box B in the low-frequency regime and for each box $A \in \mathcal{I}^B$, process $\mathcal{P}(B)$ requests $f^{A,o}$ from process $\mathcal{P}(A)$.

Alternatively, the algorithm could communicate equivalent densities for the LF-M2L and HF-M2L translations at each level in the preorder traversal of the octree. In theory, this provides an opportunity to overlap computation and communication to improve performance. However, in the numerical experiments of section 4, we show that the communication time is minimal. Using the bulk communication keeps the

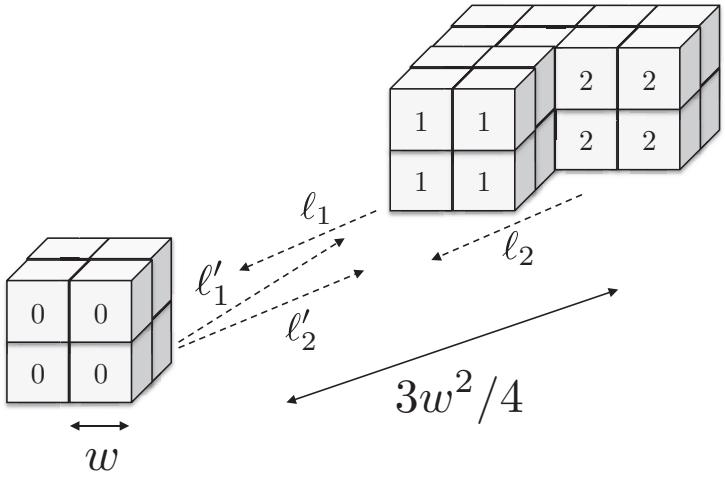

FIG. 4. *Interaction in the high-frequency regime that induces communication in the parallel algorithm. In this case, process* 0 *needs outgoing directional equivalent densities in direction* ℓ_1 *from process* 1 *and direction* ℓ_2 *from process* 2*. The equivalent densities are translated to incoming directional potentials in the directions* ℓ'_1 *and* ℓ'_2 *via HF-M2L. Similarly, processes* 1 *and* 2 *need outgoing directional equivalent densities in directions* ℓ'_1 *and* ℓ'_2 *from process* 0*.*

algorithm simple without sacrificing performance.

3.3. Parallel algorithm. We finally arrive at our proposed parallel algorithm. Communication for the HF-M2L translations occurs after the HF-M2M translations are complete, and communication for the LF-M2L translations occurs after the HF-L2L translations are complete. The parallel algorithm from the view of process q is in Algorithm 2. We assume that communication is not overlapped with computation. Although there is opportunity to hide communication by overlapping, the overall communication time is small in practice, and the more pressing concern is proper load balancing (see section 4).

3.4. Communication complexity. We will now discuss the communication complexity of our algorithm assuming that the N points are quasi-uniformly sampled from a two-dimensional manifold with $N = \mathcal{O}(K^2)$, which is generally satisfied for many applications which involve boundary integral formulations. In the following proposition, we consider the setting of a constant number of points sampled per wavelength (λ) for a fixed geometry. We note, however, that the implementation of the algorithm does not depend on this sampling rate. Instead of analyzing smaller λ, we fix $\lambda = 1$ and scale the geometry by $1/\lambda = K$.

PROPOSITION 3.1. *Let* $\mathbb{S}$ *be a surface in* $B(0, 1/2)$*. Suppose that for a fixed* K*, the points* $\{p_i, 1 \le i \le N\}$ *are samples of* $K\mathbb{S}$*, where* $N = \mathcal{O}(K^2)$ *and* $K\mathbb{S} = \{K \cdot p | p \in \mathbb{S}\}$ *(the surface obtained by magnifying* $\mathbb{S}$ *by a factor of* K*). Then, for any prescribed accuracy, the proposed algorithm sends at most* $\mathcal{O}(K^2 \log K) = \mathcal{O}(N \log N)$ *data in aggregate.*

Outline of the proof. We analyze the amount of data communicated at each of two steps.

- We claim that at most $\mathcal{O}(N \log N)$ data is communicated for HF-M2L. We know that, for a box B of width w, the boxes in its high-frequency interaction list are approximately located in a ball centered at B with radius

Algorithm 2: Parallel directional FMM for (1.1) from the view of process q.

Data: points $\{p_i\}_{1\le i\le N}$, densities $\{f_i\}_{1\le i\le N}$, octree $\mathcal{T}$, partition $\mathcal{P}$
Result: potentials $\{u_i\}_{1\le i\le N}$

foreach *box* B *in* postorder *traversal of* $\mathcal{T}$ *with* $w(B) < 1, \mathcal{P}(B) = q$ **do**
  foreach *child box* C *of* B **do**
    `//` $\mathcal{P}(C) = \mathcal{P}(B) = q$`, so child data is available locally`
    Transform $\{f_k^{C,o}\}$ to $\{f_k^{B,o}\}$ via LF-M2M

foreach *box* B *in* postorder *traversal of* $\mathcal{T}$ *with* $1 \le w(B) \le \sqrt{K}, \mathcal{P}(B) = q$ **do**
  foreach *direction* ℓ *of* B **do**
    Let the ℓ'th wedge of B's children contain the ℓth wedge of B
    foreach *child box* C *of* B **do**
      `//` $\mathcal{P}(C) = \mathcal{P}(B) = q$`, so child data is available locally`
      Transform $\{f_k^{C,o,\ell'}\}$ to $\{f_k^{B,o,\ell}\}$ via HF-M2M

`// High-frequency M2L data communication`
foreach *box* B *with* $w(B) > 1$ *and* $\mathcal{P}(B) = q$ **do**
  foreach *box* $A \in \mathcal{I}^B$ **do**
    Let ℓ be the direction for which $B \in \mathcal{I}^A$
    Request $f^{A,o,\ell}$ from process $\mathcal{P}(A)$

foreach *box* B *in* preorder *traversal of* $\mathcal{T}$ *with* $1 \le w(B) \le \sqrt{K}, \mathcal{P}(B) = q$ **do**
  foreach *direction* ℓ *of* B **do**
    foreach *box* $A \in I^B$ *in direction* ℓ **do**
      Let ℓ' be the direction for which $B \in I^A$
      Transform $\{f_k^{A,o,\ell'}\}$ via HF-M2L and add result to $\{u_k^{B,i,\ell}\}$
    foreach *child box* C *of* B **do**
      `//` $\mathcal{P}(C) = \mathcal{P}(B) = q$`, so child data is available locally`
      Transform $\{u_k^{B,i,\ell}\}$ to $\{u_k^{C,i,\ell}\}$ via HF-L2L.

`// Low-frequency M2L data communication`
foreach *box* B *with* $w(B) \le 1$ *and* $\mathcal{P}(B) = q$ **do**
  foreach *box* $A \in \mathcal{I}^B$ **do**
    Request $f^{A,o}$ from process $\mathcal{P}(A)$

foreach *box* B *in* preorder *traversal of* $\mathcal{T}$ *with* $w(B) < 1, \mathcal{P}(B) = q$ **do**
  foreach $A \in I^B$ **do**
    Transform $\{f_k^{A,o}\}$ via LF-M2L and add the result to $\{u_k^{B,i}\}$
  Transform $\{u_k^{B,i}\}$ via LF-L2L
  if B *is a leaf box* **then**
    foreach $p_i \in B$ **do**
      Add result to u_i
  else
    `//` $\mathcal{P}(C) = \mathcal{P}(B) = q$`, so child data is available locally`
    foreach *child box* C *of* B **do**
      Add result to $\{u_k^{C,i}\}$

$3w^2/4$ and that there are $\mathcal{O}(w^2)$ directions $\{\ell\}$. The fact that our points are samples from a two-dimensional manifold implies that there are at most $\mathcal{O}(w^4/w^2) = \mathcal{O}(w^2)$ boxes in B's interaction list. Each box in the interaction list contributes at most a constant amount of communication. Since the points are sampled from a two-dimensional manifold, there are $\mathcal{O}(K^2/w^2)$ boxes of width w. Thus, each level in the high-frequency regime contributes at most $\mathcal{O}(K^2)$ communication. We have $\mathcal{O}(\log K)$ levels in the high-frequency regime ($w = 1, 2, 4, \ldots, \sqrt{K}$), so the total communication volume for HF-M2L is at most $\mathcal{O}(K^2 \log K) = \mathcal{O}(N \log N)$.

- In the low-frequency regime, there are $\mathcal{O}(N)$ boxes, and the interaction list of each box is of a constant size. Thus, at most $\mathcal{O}(N)$ data is communicated for LF-M2L. ∎

The constants in the above analysis strongly depend on the geometry of the problem. Specifically, the number of boxes in the octree that contain surface points varies by geometry. This has implications for scalability, which we explore theoretically in subsection 3.7 and empirically in subsection 4.2.

To analyze the cost of communication, we use a common model: each process can simultaneously send and receive a single message at a time, and the cost of sending a message with n units of data is $\alpha + \beta n$ [1, 10, 38]. The constant α is the start-up cost, or latency, and β is the inverse bandwidth. At each of the two communication steps, each process needs to communicate with every other process. If the communication is well-balanced at each level of the octree, i.e., the amount of data communicated between any two processes is roughly the same, then the communication is similar to an "all-to-all" or "index" pattern. The communication time is then $\mathcal{O}(p\alpha + \frac{1}{p}K^2 \log K\beta)$ or $\mathcal{O}(\log p\alpha + \frac{\log p}{p}K^2 \log K\beta)$ using standard algorithms [5, 38]. We emphasize that, while we have not shown that such balanced partitions even always exist, if the partition $\mathcal{P}$ clusters the assignment of boxes to processes, then many boxes in the interaction lists are already owned by the correct process. This tends to reduce communication, and in practice, the actual communication time is small (see subsection 4.2).

3.5. Computational complexity. Our algorithm parallelizes all of the translation operators, which constitute nearly all of the computations. This is a direct result from the assignment of boxes to processes by $\mathcal{P}$. However, the computational complexity analysis depends heavily on the partition. We will make idealizing assumptions on the partitions to show how the complexity changes with the number of processes, p.

Recall that we assume that communication and computation are not overlapped. Thus, the translations are computed simultaneously on all processes.

PROPOSITION 3.2. *Suppose that each process has the same number boxes in the low-frequency regime. Then the LF-M2M, LF-M2L, and LF-L2L translations have computational complexity* $\mathcal{O}(N/p)$.

Proof. Each nonempty box requires a constant amount of work for each of the low-frequency translation operators since the interaction lists are of constant size. There are $\mathcal{O}(N)$ boxes in the low-frequency regime, so the computational complexity is $\mathcal{O}(N/p)$ if each process has the same number of boxes. ∎

We note that Proposition 3.2 makes no assumption on the assignment of the computation to process at any fixed level in the octree: we need only a load-balanced partition throughout *all* of the boxes in the low-frequency regime. It is important to note that empty boxes (i.e., boxes that do not contain any of the $\{p_i\}$) require no

computation. When dealing with two-dimensional surfaces in three dimensions, this poses a scalability issue (see subsection 3.7).

For the high-frequency translation operators, instead of a load balancing the total number of boxes, we need to load balance the number of directions. We emphasize that $\mathcal{P}$ partitions boxes. Thus, if $\mathcal{P}(B) = q$, process q is assigned all directions for the box B.

PROPOSITION 3.3. *Suppose that each process is assigned the same number of directions and that the assumptions of Proposition* 3.1 *are satisfied. Then the HF-M2M, HF-M2L, and HF-L2L translations have computational complexity* $\mathcal{O}(N \log N/p)$.

Proof. For HF-M2M, each box of width w contains $\mathcal{O}(w^2)$ directions $\{\ell\}$. Each direction takes $\mathcal{O}(1)$ computations. There are $\mathcal{O}(\log K)$ levels in the high-frequency regime. Thus, in parallel, HF-M2M translations take $\mathcal{O}(N \log N/p)$ operations provided that each process has the same number of directions. The analysis for HF-L2L is analogous.

Now consider HF-M2L. Following the proof of Proposition 3.1, there are $\mathcal{O}(w^2)$ directions $\{\ell\}$ for a box B of width w. For each direction, there are $\mathcal{O}(1)$ boxes in B's interaction list, and each interaction is $\mathcal{O}(1)$ work. Thus, each direction ℓ induces $\mathcal{O}(1)$ computations. There are $\mathcal{O}(K^2 \log K) = \mathcal{O}(N \log N)$ directions. Therefore, in parallel, HF-M2L translations take $\mathcal{O}(N \log N/p)$ operations provided that each process has the same number of directions. ∎

The asymptotic analysis absorbs several constants that are important for understanding the algorithm. In particular, the size of the translation matrix D in (2.1) drives the running time of the costly HF-M2L translations. The size of D depends on the box width w, the direction ℓ, and the target accuracy ϵ. In subsection 4.1, we discuss the size of D for problems used in our numerical experiments. The constants also depend on the geometry of the problem. For example, the number of directions per box can vary at any level for a nonuniform geometry. In addition, the geometry affects load balancing and scalability. We discuss these issues for practical problems in subsection 4.2.

Finally, we note that the critical path length is small. Algorithm 2 consists of one postorder and one preorder traversal of a tree. The tree has depth $\mathcal{O}(\log \sqrt{K})$ in the high-frequency regime and depth $\mathcal{O}(\log N)$ in the low-frequency regime, for a total depth of $\mathcal{O}(\log N)$. Therefore, the critical path length is only $\mathcal{O}(\log N)$.

3.6. Spatial complexity. The memory requirements of our algorithm are directly linked with the boxes in the low-frequency regime and the directions in the high-frequency regime. Propositions 3.4 and 3.5 show that only $\mathcal{O}(1)$ data is stored for each box and each direction. No additional storage requirements impact the spatial complexity analysis.

PROPOSITION 3.4. *Suppose that each process has the same number boxes in the low-frequency regime. Then Algorithm* 2 *uses* $\mathcal{O}(N/p)$ *memory in the low-frequency regime.*

Proof. The algorithm stores the low-frequency translation matrices and the outgoing and incoming equivalent points, equivalent charges, check points, and check potentials (see section 2 for definitions of the data). Each type of data requires $\mathcal{O}(1)$ memory for each box, and there are $\mathcal{O}(N)$ boxes in the low-frequency regime. ∎

PROPOSITION 3.5. *Suppose that each process is assigned the same number of directions and that the assumptions of Proposition* 3.1 *are satisfied. Then Algorithm* 2 *uses* $\mathcal{O}(N \log N/p)$ *memory in the high-frequency regime.*

TABLE 1
Maximum dimension of the translation matrix D from (2.1) for each target accuracy ϵ and box width w. Given ϵ, the maximum dimension is approximately the same for each box width.

	$w = 1$	$w = 2$	$w = 4$	$w = 8$	$w = 16$
ϵ =1e-4	60	58	60	63	67
ϵ =1e-6	108	93	96	95	103
ϵ =1e-8	176	142	141	136	144

Proof. The algorithm stores the high-frequency translation matrices and the outgoing and incoming directional equivalent points, equivalent charges, check points, and check potentials. Each type of data requires $\mathcal{O}(1)$ memory for each box and direction ℓ, and there are $\mathcal{O}(N \log N)$ directions in the high-frequency regime. ∎

Since we need $\mathcal{O}(N/p)$ memory to store the solution, the spatial complexity is within a log factor of optimal. The equivalent points in the high-frequency regime are needed for the duration of the algorithm, so the log factor is necessary.

There are several compression techniques for storing the low-frequency translation matrices [43]. The number of high-frequency translation matrices is asymptotically much smaller than $\mathcal{O}(N \log N)$. Since the Green's function of the Helmholtz equation is translation invariant, we store one matrix for each direction for each box width $w \geq 1$. There are $\mathcal{O}(w^2)$ directions $\{\ell\}$ for a box width w, $w = 1, 2, \ldots, \mathcal{O}(\sqrt{K})$, so there are only $\mathcal{O}(K)$ translation matrices used in the high-frequency regime.

3.7. Scalability. Recall that a single process performs the computations associated with all descendent boxes of a given box at the partition level. Thus, the scalability of the parallel algorithm is limited by the number of nonempty boxes in the octree at the partition level. If there are n boxes at the partition level that contain points, then the maximum speedup is n. Since the boxes at the partition level can have width as small as $\mathcal{O}(\sqrt{K})$, there are $\mathcal{O}((K/\sqrt{K})^3) = \mathcal{O}(K^{3/2})$ total boxes at the partition level. However, in the scattering problems of interest, the points are sampled from a two-dimensional surface and are highly nonuniform. In this setting, the number of nonempty boxes of width w is $\mathcal{O}(K^2/w^2)$. The boxes at the partition level have width $\mathcal{O}(\sqrt{K})$, so the number of nonempty boxes is $\mathcal{O}(K)$. If $p = \mathcal{O}(K)$, then the communication complexity is $\mathcal{O}(\alpha K + K \log K \beta) = \mathcal{O}(\alpha\sqrt{N} + \sqrt{N} \log N \beta)$, and the computational complexity is $\mathcal{O}(\sqrt{N} \log N)$.

The scalability of our algorithm is limited by the geometry of our objects, which we observe in subsection 4.2. However, we are still able to achieve significant strong scaling with this limitation.

4. Numerical results. We now illustrate how the algorithm scales through a number of numerical experiments. Our algorithm is implemented with C++, and the communication is implemented with the message passing interface (MPI). All experiments were conducted on the Lonestar supercomputer at the Texas Advanced Computing Center. Each node has two Xeon Intel Hexa-Core 3.3 GHz processes (i.e., 12 total cores) and 24 GB of DRAM, and the nodes are interconnected with InfiniBand. Each experiment used four MPI processes per node. The implementation of our algorithm is available at https://github.com/arbenson/ddfmm.

4.1. Precomputation and separation rank. The precomputation for Algorithm 2 is the computation of the matrices $D = (d_{pq})_{1 \leq p,q \leq r_\epsilon}$ used in (2.1). For each box width w, there are approximately $2w^2$ directions ℓ that cover the directional interactions. Each direction at each box width requires the high-frequency translation

TABLE 2
Size of files that store precomputed data for the parallel algorithm. The data are suitable for $K < 1024$. For larger values of K, translation matrices for more boxes are needed.

	ϵ = 1e-4	ϵ =1e-6	ϵ =1e-8
High-frequency	89 MB	218 MB	440 MB
Low-frequency	19 MB	64 MB	158 MB

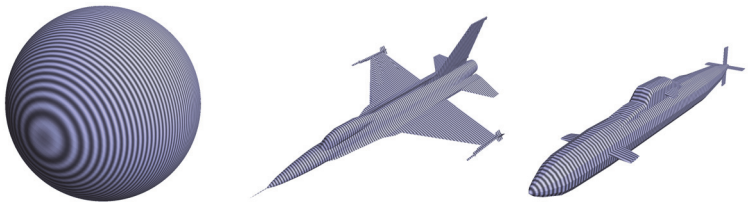

FIG. 5. *Sphere (left), F16 (middle), and submarine (right) geometries.*

matrix D. The dimension of D is called the *separation rank*. The maximum separation rank for each box width w and target accuracy ϵ is in Table 1. We see that the separation rank is approximately constant when ϵ is fixed. As ϵ increases, our algorithm is more accurate and needs larger translation matrices.

In theory, each process needs only a subset of the translation matrices (subsection 3.6). However, in practice, the storage requirements are small enough that we let each process store all translation matrices. Table 2 lists the size of the data files used in our experiments that store the high- and low-frequency translation matrices.

4.2. Strong scaling on different geometries. We test our algorithm on three geometries: a sphere, an F16 fighter aircraft, and a submarine. The geometries are in Figure 5. The surface of each geometry is represented by a triangular mesh, and the point set $\{p_i\}$ is generated by sampling the triangular mesh randomly with 10 points per wavelength on average. The densities $\{f_i\}$ are generated by independently sampling from a standard normal distribution.

In our experiments, we use simple clustering techniques to partition the data and create the partition $\mathcal{P}$. To partition the point set $\{p_i, 1 \le i \le N\}$, we use k-means clustering with p clusters. The points (and corresponding densities) in the qth cluster are assigned to process q. This ignores the octree boxes where the points live, but tends to place many points in a single box on the same process. To create $\mathcal{P}$, we use an algorithm that greedily assigns a box to the process that owns the most points in the box while also balancing the number of boxes on each process. Algorithm 3 formally describes the construction of $\mathcal{P}$. Because the points are partitioned with k-means clustering, Algorithm 3 outputs a map $\mathcal{P}$ that tends to cluster the assignment of process to box as well.

Figure 6 shows strong scaling results for the algorithm on the geometries with target accuracy ϵ =1e-4, 1e-6, and 1e-8. The smallest number of processes used for each geometry is chosen as the smallest power of two such that the algorithm does not run out of memory when ϵ = 1e-8. Our algorithm scales best for the sphere geometry. This makes sense because the symmetry of the geometry balances the number of directions assigned to each process. While we use a power-of-two number of processes in the scaling experiments, our implementation works for an arbitrary number of processes.

Scaling levels off at 512 processes for the F16 and 256 processes for the submarine.

Algorithm 3: Construction of $\mathcal{P}$ used in numerical experiments.

Data: number of processes p, partition level L
Result: partition $\mathcal{P}$

Run k-means on point set $\{p_i, 1 \leq i \leq N\}$ with p clusters.
Assign the points in the qth cluster to process q.
Compute $B_{max} = \lceil \#\left(\text{nonempty boxes at level } L\right)/p \rceil$
$C = \{0, 1, \ldots, p-1\}$
for $i = 0$ *to* $p-1$ **do**
  $B_i = 0$
foreach *nonempty box* B *at level* L **do**
  $i = \arg\max_{j \in C} |\{p_k \in B | p_k \text{ on process } j\}|$
  $\mathcal{P}(B) = i$
  $B_i = B_i + 1$
  if $B_i == B_{max}$ **then**
    $C = C \backslash \{i\}$

TABLE 3
Number of nonempty boxes at the partition level L. As discussed in subsection 3.7, this limits scalability. In all numerical experiments for all geometries, $L = 5$.

Geometry	#(nonempty boxes)	#(boxes)
Sphere	2,136	32,768
F16	342	32,768
Submarine	176	32,768

This is a direct consequence of the number of nonempty boxes in the octree at the partition level, which was discussed in subsection 3.7. Table 3 lists the number of nonempty boxes for the three geometries.

Table 4 provides a breakdown of the running times of the different components of the algorithm. We see the following trends in the data:

- The HF-M2L and HF-L2L (the downwards pass in the high-frequency regime) constitute the bulk of the running time. M2L is typically the most expensive translation operator [44], and the interaction lists are larger in the high-frequency regime than in the low-frequency regime.
- Communication time is small. The high-frequency and low-frequency communication combined are about 1% of the running time. For the number of processes in these experiments, this problem is expected to be compute bound. This agrees with studies on communication times in KI-FMM [11].
- The more accurate solutions do not scale as well as less accurate solutions. Higher accuracy solutions are more memory constrained, and we speculate that lower accuracy solutions benefit relatively more from cache effects when the number of processes increases.

The data show superlinear speedups for the sphere and F16 geometries at some accuracies. This is partly caused by the suboptimality of the partitioning algorithm (Algorithm 3). However, the speedups are still encouraging.

5. Conclusion and future work. We have presented the first parallel algorithm for the directional FMM to solve N-body problems with the Helmholtz kernel. Under proper load balancing, the computational complexity of the algorithm is

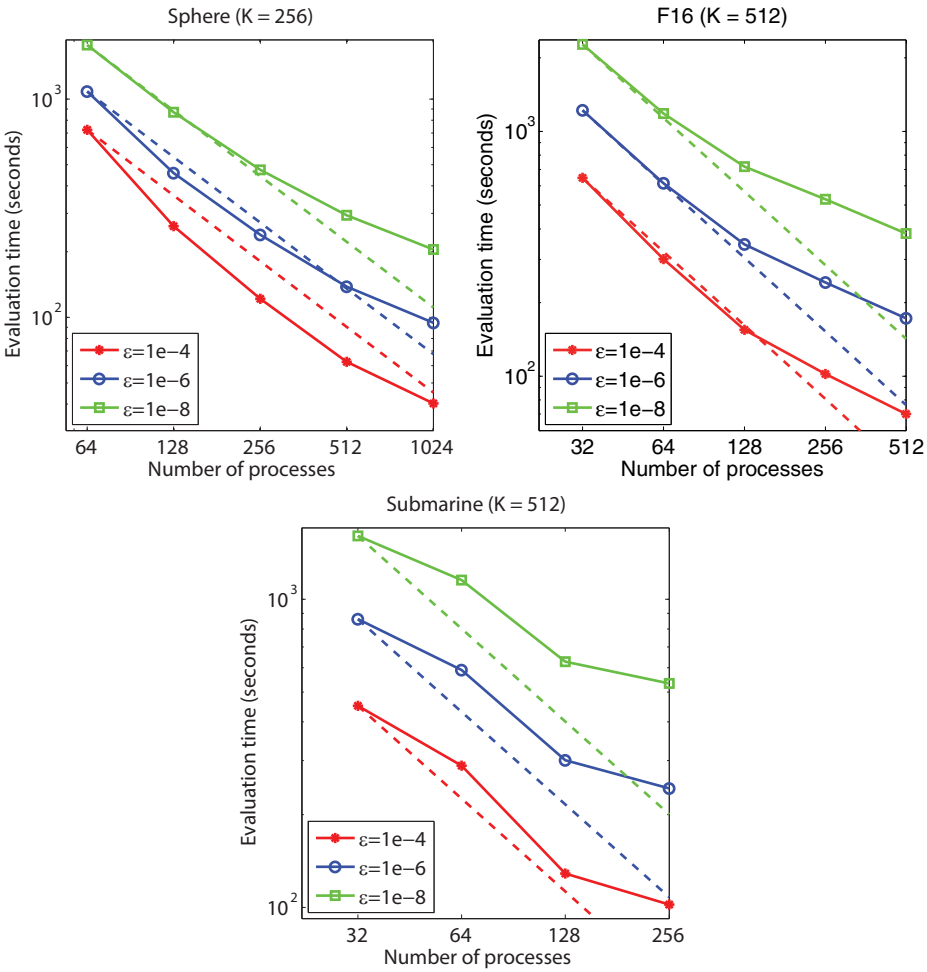

FIG. 6. *Strong scaling experiments for sphere ($K = 256$), F16 ($K = 512$), and submarine ($K = 512$). For each geometry, we test with target accuracy $\epsilon = 1e-4, 1e-6, 1e-8$. Dashed lines are linear scaling from the running time with the smallest number of processes.*

$\mathcal{O}(N \log N/p)$, and the communication time is $\mathcal{O}(p\alpha + \frac{1}{p} N \log N \beta)$. Communication time is minimal in our experiments. The number of processes is limited to $\mathcal{O}(\sqrt{N})$, so the running time can essentially be reduced to $\mathcal{O}(\sqrt{N} \log N)$. The algorithm exhibits good strong scaling for a variety of geometries and target accuracies.

In future work, we plan to address the following issues:

- Partitioning the computation at a fixed level in the octree avoids communication at high levels in the octree but limits the parallelism. In future work, we plan to divide the work at lower levels of the octree and introduce more communication. Since communication is such a small part of the running time in our algorithm, it is reasonable to expect further improvements with these changes.
- Our algorithm uses $\mathcal{O}(w^2)$ wedges for boxes of width w. This makes precomputation expensive. We would like to reduce the total number of wedges at each box width.

TABLE 4

Breakdown of running times for the parallel algorithm. All listed times are in seconds. For all geometries, the more accurate solutions produce worse scalings.

Geometry	K	N	ϵ	p	HF-M2M	HF-M2L + HF-L2L	LF-M2L + LF-L2L	HF+LF Comm.	Total	Efficiency
Sphere	256	1.57e7	1e-4	64	29.44	525.69	159.83	5.53	722.73	100.0%
				128	17.76	187.88	50.85	2.77	261.55	138.16%
				256	12.1	88.7	17.54	1.96	121.32	148.93%
				512	8.83	45.75	6.19	0.97	62.47	144.61%
				1024	7.28	29.12	2.8	0.71	40.25	112.22%
Sphere	256	1.57e7	1e-6	64	70.4	773.39	229.38	8.18	1084.9	100.0%
				128	43.21	334.31	73.89	4.39	458.11	118.41%
				256	30.57	177.5	26.6	2.16	238.92	113.52%
				512	23.0	102.91	10.1	1.31	138.3	98.05%
				1024	18.71	69.02	4.86	1.04	94.36	71.86%
Sphere	256	1.57e7	1e-8	64	167.73	1293.32	284.51	21.24	1773.42	100.0%
				128	101.58	657.19	101.22	8.16	872.38	101.64%
				256	70.41	359.07	39.89	2.7	474.19	93.5%
				512	52.31	220.45	17.93	1.52	293.57	75.51%
				1024	42.28	150.04	9.33	1.2	204.67	54.15%
F16	512	1.19e7	1e-4	32	33.22	470.32	134.85	4.49	646.01	100.0%
				64	19.32	230.77	46.92	2.48	301.27	107.21%
				128	12.66	122.01	17.67	1.43	154.88	104.27%
				256	8.93	80.78	9.67	1.68	102.23	78.99%
				512	7.14	56.32	5.11	0.73	70.0	57.68%
F16	512	1.19e7	1e-6	32	84.26	895.07	229.44	5.25	1219.35	100.0%
				64	49.46	480.55	79.28	3.13	615.39	99.07%
				128	32.72	276.15	33.25	1.97	345.99	88.11%
				256	23.73	197.55	17.26	1.41	241.43	63.13%
				512	19.18	140.59	10.48	0.99	172.41	44.2%
F16	512	1.19e7	1e-8	32	195.51	1769.14	290.37	7.04	2271.94	100.0%
				64	118.07	944.96	112.72	4.25	1185.44	95.83%
				128	76.21	585.09	50.34	2.76	717.81	79.13%
				256	57.51	435.65	31.25	2.16	529.37	53.65%
				512	46.1	314.68	18.73	1.41	383.06	37.07%
Submarine	512	9.36e6	1e-4	32	23.42	349.93	71.66	4.07	451.51	100.0%
				64	17.18	226.32	40.21	2.65	288.43	78.27%
				128	10.2	104.03	12.36	1.43	128.99	87.51%
				256	9.01	82.88	8.36	1.07	102.14	55.26%
Submarine	512	9.36e6	1e-6	32	57.6	667.59	128.79	5.35	863.21	100.0%
				64	42.15	475.55	65.86	3.35	589.66	73.2%
				128	26.37	245.83	24.66	1.91	300.38	71.84%
				256	23.56	201.56	15.27	1.51	243.27	44.35%
Submarine	512	9.36e6	1e-8	32	136.74	1281.87	172.27	8.09	1606.38	100.0%
				64	101.14	948.74	94.44	4.7	1154.14	69.59%
				128	61.57	523.28	37.73	2.66	628.12	63.94%
				256	55.03	447.74	25.8	2.01	532.93	37.68%

- Our implementation uses a simple octree construction. We would like to incorporate the more sophisticated approaches of software such as `p4est` [7] and `Dendro` [34] in order to improve our parallel octree manipulations. This would alleviate memory constraints, since octree information is currently redundantly stored on each process.

REFERENCES

[1] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in numerical linear algebra*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 866–901.

[2] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, *AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems*, Radio Science, 31 (1996), pp. 1225–1251.

[3] N. N. Bojarski, *K-space Formulation of the Electromagnetic Scattering Problem*, Technical report, DTIC Document, Fort Belvoir, VA, 1972.

[4] B. Bradie, R. Coifman, and A. Grossmann, *Fast numerical computations of oscillatory integrals related to acoustic scattering,* I, Appl. Comput. Harmon. Anal., 1 (1993), pp. 94–99.

[5] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby, *Efficient algorithms for all-to-all communications in multiport message-passing systems*, IEEE Trans. Parallel Distrib. Systems, 8 (1997), pp. 1143–1156.

[6] O. P. Bruno and L. A. Kunyansky, *A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications*, J. Comput. Phys., 169 (2001), pp. 80–110.

[7] C. Burstedde, L. C. Wilcox, and O. Ghattas, *p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees*, SIAM J. Sci. Comput., 33 (2011), pp. 1103–1133.

[8] F. X. Canning, *Sparse approximation for solving integral equations with oscillatory kernels*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 71–87.

[9] C. Cecka and E. Darve, *Fourier-based fast multipole method for the Helmholtz equation*, SIAM J. Sci. Comput., 35 (2013), pp. A79–A103.

[10] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn, *Collective communication: Theory, practice, and experience*, Concurrency and Computation: Practice and Experience, 19 (2007), pp. 1749–1783.

[11] A. Chandramowlishwaran, J. W. Choi, K. Madduri, and R. W. Vuduc, *Brief announcement: Towards a communication optimal fast multipole method and its implications at exascale*, in Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures, ACM, New York, 2012, pp. 182–184.

[12] A. Chandramowlishwaran, S. Williams, L. Oliker, I. Lashuk, G. Biros, and R. Vuduc, *Optimizing and tuning the fast multipole method for state-of-the-art multicore architectures*, in Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), IEEE, Piscataway, NJ, 2010, pp. 1–12.

[13] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, *A wideband fast multipole method for the Helmholtz equation in three dimensions*, J. Comput. Phys., 216 (2006), pp. 300–325.

[14] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys., 155 (1999), pp. 468–498.

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

[16] D. L. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, 3rd ed., Appl. Math. Sci. 93, Springer, New York, 2013.

[17] E. Darve, C. Cecka, and T. Takahashi, *The fast multipole method on parallel clusters, multicore processors, and graphics processing units*, C. R. Méch., 339 (2011), pp. 185–193.

[18] L. Demanet, M. Ferrara, N. Maxwell, J. Poulson, and L. Ying, *A butterfly algorithm for synthetic aperture radar imaging*, SIAM J. Imaging Sci., 5 (2012), pp. 203–243.

[19] B. Engquist and L. Ying, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM Journal on Scientific Computing, 29 (2007), pp. 1710–1737.

[20] B. Engquist and L. Ying, *A fast directional algorithm for high frequency acoustic scattering in two dimensions*, Commun. Math. Sci., 7 (2009), pp. 327–345.

[21] B. Engquist and L. Ying, *Fast directional algorithms for the Helmholtz kernel*, J. Comput. Appl. Math., 234 (2010), pp. 1851–1859.

[22] O. Ergül and L. Gürel, *Efficient parallelization of the multilevel fast multipole algorithm for the solution of large-scale scattering problems*, IEEE Trans. Antennas and Propagation, 56 (2008), pp. 2335–2345.

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

[24] L. Greengard and W. D. Gropp, *A parallel version of the fast multipole method*, Comput. Math. Appl., 20 (1990), pp. 63–71.

[25] J. G. Harris, *Linear Elastic Waves*, Cambridge Texts Appl. Math. 26, Cambridge University Press, Cambridge, UK, 2001.

[26] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros, *A massively parallel adaptive fast multipole method on heterogeneous architectures*, Commun. ACM, 55 (2012), pp. 101–109.

[27] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1: Statistics, University of California Press, Berkeley, CA, 1967, pp. 281–297.

[28] M. Messner, M. Schanz, and E. Darve, *Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation*, J. Comput. Phys., 231 (2012), pp. 1175–1196.

[29] E. Michielssen and A. Boag, *Multilevel evaluation of electromagnetic fields for the rapid solution of scattering problems*, Microwave and Optical Technology Letters, 7 (1994), pp. 790–795.

[30] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering from large structures*, IEEE Trans. Antennas and Propagation, 44 (1996), pp. 1086–1093.

[31] J. Poulson, L. Demanet, N. Maxwell, and L. Ying, *A parallel butterfly algorithm*, SIAM J. Sci. Comput., 36 (2014), pp. C49–C65.

[32] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, et al., *Petascale direct numerical simulation of blood flow on* 200*k cores and heterogeneous architectures*, in Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Computer Society, 2010, pp. 1–11.

[33] V. Rokhlin, *Diagonal Forms of Translation Operators For Helmholtz Equation in Three Dimensions*, Technical report, DTIC Document, Fort Belvoir, VA, 1992.

[34] R. S. Sampath, S. S. Adavani, H. Sundar, I. Lashuk, and G. Biros, *Dendro: Parallel algorithms for multigrid and AMR methods on* 2:1 *balanced octrees*, in Proceedings of the 2008 ACM/IEEE conference on Supercomputing, IEEE Press, New Brunswick, NJ, 2008, pp. 1–12.

[35] J. M. Song and W. C. Chew, *Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering*, Microwave and Optical Technology Letters, 10 (1995), pp. 14–19.

[36] J. Song, C.-C. Lu, and W. C. Chew, *Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 1488–1493.

[37] J. M. Taboada, M. G. Araujo, J. M. Bertolo, L. Landesa, F. Obelleiro, and J. L. Rodriguez, *MLFMA-FFT parallel algorithm for the solution of large-scale problems in electromagnetics*, Progress In Electromagnetics Research, 105 (2010), pp. 15–30.

[38] R. Thakur, R. Rabenseifner, and W. Gropp, *Optimization of collective communication operations in MPICH*, International Journal of High Performance Computing Applications, 19 (2005), pp. 49–66.

[39] K. D. Tran, *Fast Numerical Methods for High Frequency Wave Scattering*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 2012.

[40] P. Tsuji and L. Ying, *A fast directional algorithm for high-frequency electromagnetic scattering*, J. Comput. Phys., 230 (2011), pp. 5471–5487.

[41] C. Waltz, K. Sertel, M. A. Carr, B. C. Usner, and J. L. Volakis, *Massively parallel fast multipole method solutions of large electromagnetic scattering problems*, IEEE Trans. Antennas and Propagation, 55 (2007), pp. 1810–1816.

[42] L. Ying, *Fast Algorithms for Boundary Integral Equations*, Multiscale Modeling and Simulation in Science, Lect. Notes Comput. Sci. Eng. 66, Springer, Berlin, 2009, pp. 139–193.

[43] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

[44] L. Ying, G. Biros, D. Zorin, and H. Langston, *A new parallel kernel-independent fast multipole method*, in Proceedings of the 2003 ACM/IEEE Conference Supercomputing, ACM, New York, 2003, p. 14.

[45] R. Yokota and L. Barba, *Hierarchical n-body simulations with autotuning for heterogeneous systems*, Computing in Science & Engineering, 14 (2012), pp. 30–39.

[46] R. Yokota and L. A. Barba, *A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems*, International Journal of High Performance Computing Applications, 26 (2012), pp. 337–346.