Biomolecular Electrostatics Simulation by an FMM-based BEM on 512 GPUs

Rio Yokota
Department of Mathematics
University of Bristol
Bristol, United Kingdom BS8 1TW
rio.yokota@bristol.ac.uk

Tsyoshi Hamada
Nagasaki University
Advanced Computing Center (NACC)
Nagasaki, Japan
hamada@nacc.nagasaki-u.ac.jp

Jaydeep P. Bardhan
Dept. of Molecular Biophysics and Physiology
Rush University Medical Center
Chicago, IL 60612
jbardhan@alum.mit.edu

Matthew G. Knepley
Computation Institute, University of Chicago
Chicago, IL 60637
knepley@ci.uchicago.edu

L. A. Barba
Mechanical Engineering, Boston University
Boston, MA 02215
labarba@bu.edu

Abstract—We present simulations of biomolecular electrostatics at a scale not reached before, thanks to both algorithmic and hardware acceleration. The algorithmic acceleration is achieved with the fast multipole method (FMM) in conjunction with a boundary element method (BEM) formulation of the continuum electrostatic model. The hardware acceleration is achieved through graphics processors, GPUs. We demonstrate the power of our algorithms and software for the calculation of the electrostatic interactions between biological molecules in solution. Computational experiments are presented simulating the electrostatics of protein–drug binding and several multimillion atom systems consisting of hundreds to thousands of copies of the protein lysozyme. Even for the largest of these problems, which models over 20 million atoms and has more than six billion unknowns, one iteration step requires only a few minutes on 512 GPU nodes. We achieved a sustained performance of 34.6TFlops for the entire BEM calculation. We are currently adapting our solver to model the linearized Poisson–Boltzmann equation for dilute ionic solutions, and our BEM simulator is flexible enough to solve a variety of other integral equation problems as well, ranging from simple Poisson problems to Helmholtz problems in electromagnetics and acoustics to high Reynolds number flow.

I. INTRODUCTION

Electrostatic interactions play an essential role in the structure and function of biomolecules (proteins, DNA, cell membranes, etc.) [1]. One of the most challenging aspects for understanding these interactions is the fact that biologically active molecules are almost always in solution—that is, they are surrounded by water molecules and dissolved ions. These solvent molecules add many thousands or even millions more degrees of freedom to any theoretical study, many of which are of secondary importance for investigations of interest. Classical molecular dynamics (MD) methods, implemented in software such as CHARMM [2] and NAMD [3], use all-atom representations of the biomolecule and solvent, and compute the trajectories of every atom over time by following Newton’s equations of motion. MD methods represent some of the most detailed approaches to studying biomolecular systems, but computing an “average” electrostatic energy in such systems can be extremely expensive, even when one uses efficient linear-scaling methods for the long-range electrostatics between all the atoms, such as particle-mesh Ewald [4]. For many studies a faster, approximate method is a necessity.

In contrast with all-atom MD computations, one can model the electrostatic interactions in solvated molecules using a continuum model. A model for continuum solvation is based on assuming that the molecules and the solvent can be treated as continuous dielectric media with different constants, low and high, respectively [1]. Inside the biomolecule, in addition, point charges are arranged explicitly at the atomic positions. Thus, the electrostatic potential can be described by a Poisson equation, which for general shapes of the molecular surface has to be solved numerically. Accounting for ionic charge distributions in the solvent introduces some extra complications, as the ion locations depend on the combined effect of all charges, dielectric distributions, and the ions themselves. Making the assumption that the average electrostatic potential multiplied by the charge of the ion determines the mean force acting on the ion particle, the Poisson–Boltzmann model for biomolecular systems is obtained [1].

Standard approaches such as finite-difference methods and finite-element methods can be used to solve the Poisson or Poisson–Boltzmann equations [5], [6], [7], but there are several non-trivial challenges that must be overcome. These include the difficulties associated with mapping an irregular molecular surface to a volumetric mesh, representing the source distribution as a set of discrete point charges, and convergence issues associated with dielectric discontinuities. The development of various strategies to mitigate these problems (see, e.g., [8], [9]) and the ready availability of highly scalable, open-source software such as APBS [7] have helped make the continuum model a popular approach for studying molecular electrostatics.

An alternative approach to solving the Poisson equation directly is to determine the induced charge distribution on the molecular surface which accounts for the change in polariza-
tion charge across the dielectric boundary, using a boundary-integral equation and the boundary-element method, BEM. One significant advantage obtained from a BEM formulation is the fact that the discretization is performed over the surface of the biomolecule, rather than the three-dimensional volumetric region occupied by the molecule and solvent. Accurate solution of the linearized form of the Poisson–Boltzmann equation is also possible with BEM using various specialized techniques [10], [11], [12], [13].

The main challenge in the BEM is the computational cost of finding the induced-charge distribution, which is obtained by solving a linear system in which the matrix is dense (in contrast to the sparse linear systems associated with finite-difference and finite-element methods). To greatly reduce the expense of solving such a linear system using iterative methods, the fast multipole method (FMM) [14], [15], [16] can be used to calculate the dense matrix-vector product with the Green’s function in $O(N)$ operations [17]. In this way, the FMM algorithm can enable calculations with many millions of degrees of freedom [17], [18], [12].

In this paper, we demonstrate a fast molecular electrostatics application using a BEM formulation of the continuum model. The application is accelerated both at the algorithmic level and by hardware, achieving an unprecedented capacity for simulating large biomolecular systems. Our largest example models a large collection of proteins, totalling more than 20 million atoms, for which the BEM problem has more than one billion unknowns; on a cluster of 512 GPU-accelerated nodes this simulation requires only a few minutes to run. Large proteins and molecular machines that comprise only a few thousand or tens of thousands of atoms can be simulated in a few minutes on a single GPU or small GPU cluster.

This computational performance transforms the landscape for theoretical investigations of biomolecule electrostatics, such that the new limiting factor for these studies is the generation of suitable meshes for the BEM, rather than fast solver approaches. Our solver is based on the PetFMM library [19] with GPU acceleration, which in turn is derived from the 2009 Gordon-Bell winning FMM code [20].

II. BACKGROUND

A. Electrostatic interactions between biological molecules

The continuum electrostatic model we treat in this paper is a mixed-dielectric Poisson problem (Figure 1). The molecular interior, denoted as region $I$, is treated as a homogeneous dielectric with permittivity $\epsilon_I$ (typically between 2 and 10 [1], [21]), with the molecular charge distribution modeled as a set of $n_c$ discrete point charges located at the atom centers. Denoting the $i$th charge as having value $q_i$ and position $r_i$, the electrostatic potential in this region satisfies a Poisson equation

$$\nabla^2 \varphi_I(r) = -\sum_{i=1}^{n_c} q_i \delta(r - r_i).$$

The water surrounding the molecule, region $II$, is also modeled as a homogeneous dielectric, of permittivity $\epsilon_{II}$ equal to that of bulk water (approximately 80). In this region the Laplace equation $\nabla^2 \varphi_{II}(r) = 0$ holds, because we assume there are no fixed or mobile charges. At the dielectric boundary the potentials and the normal components of the displacement fields are continuous:

$$\epsilon_I \frac{\partial \varphi_I}{\partial n}(r) = \epsilon_{II} \frac{\partial \varphi_{II}}{\partial n}(r).$$

where $n(r)$ denotes the outward unit normal vector at $r \in \Omega$, pointing into region $II$ from region $I$. We define the boundary by rolling a probe sphere (designed to mimic a water molecule) over the union of spheres that represent the solute atoms, a definition known as the solvent-excluded surface (also the molecular surface) [22], [23].

The solute charges polarize the solvent, which in turn creates a reaction potential in the solute. In the mixed-dielectric continuum model, the solvent polarization appears as a layer of induced charge $\sigma(r)$ at the dielectric interface, where $\sigma(r)$ satisfies the second-kind Fredholm boundary integral equation [24], [25], [26], [27]:

$$\left(1 - \frac{\epsilon_I}{\epsilon_{II}} \right) \left( \frac{\partial}{\partial n(r)} \sum_{i=1}^{n_c} \frac{q_i}{4\pi ||r - r_i||} + \frac{\partial}{\partial n(r)} \int_{\Omega} \frac{\sigma(r')}{4\pi ||r - r'||} dA' \right) = \sigma(r).$$

The reaction potential in the solute is then just the Coulomb potential induced by $\sigma(r)$, i.e.,

$$\varphi^{\text{REAC}}(r) = \int_{\Omega} \frac{\sigma(r')}{4\pi ||r - r'||} dA',$$

such that the total potential $\varphi(r)$ is the sum of $\varphi^{\text{REAC}}$ and the bare Coulomb potential induced by the point charges. The electrostatic energy associated with the reaction potential represents the change in electrostatic energy associated with transferring the given charge distribution and molecular shape from a uniform low dielectric into the high dielectric medium. For this reason, the energy is called the solute’s electrostatic solvation free energy $\Delta G_{\text{solv}, es}$, and it may be written as

$$\Delta G_{\text{solv}, es} = \frac{1}{2} \sum_{i=1}^{n_c} q_i \varphi^{\text{REAC}}(r_i).$$

Electrostatic solvation free energies and the bare Coulomb energies can be used to estimate quantities such as electrostatic contributions to protein stability [28] or the binding affinity between molecules [29].
One may also investigate the effect of dilute ionic solutions on a solute using either the linearized Poisson–Boltzmann (PB) equation \( \nabla^2 \phi_{11}(r) = \kappa^2 \phi_{11}(r) \) to model the potential in the solvent, or the full nonlinear PB equation [30]. The linearized PB problem can also be solved using boundary integral equations [31], [10], [12]. For simplicity in presenting the FMM, we treat only the mixed-dielectric Poisson problem; however, we have implemented BEM to solve Helmholtz problems and are currently simplifying the method for solving linearized Poisson–Boltzmann BEM problems.

**B. Boundary-element method**

We use the boundary-element method (BEM) [32] to solve Eq. 3 numerically. The first step of the BEM is to divide the boundary \( \Omega \) into \( n_p \) discrete, non-overlapping pieces, called panels or boundary elements. Often, for complicated geometries one approximates the boundary using easily defined boundary elements such as planar triangles, a practice we follow here. Our solver infrastructure supports the use of more accurate representations with curved boundary elements, as in [33], [13], which we intend to explore in future work.

The Galerkin discretization of Eq. 3 when using constant basis functions and planar elements produces a system of linear algebraic equations

\[
Ax = b, \quad \quad \quad (6)
\]

where

\[
A_{ij} = \frac{\epsilon_i + \epsilon_{ij}}{2(\epsilon_i - \epsilon_{ij})} \alpha_i; \quad \quad \quad (7)
\]

\[
A_{ij} = \int_{\Omega_j} \frac{\partial}{\partial n(r)} \left( \int_{\Omega_i} \frac{1}{4\pi ||r - r'||} dA' \right) dA; \quad \quad \quad (8)
\]

\[
b_i = \sum_{k=1}^{n_p} q_k \int_{\Omega_i} \frac{1}{4\pi ||r - r_k||} dA, \quad \quad \quad (9)
\]

and \( \alpha_i \) represents the area of the \( i \)th element. In this paper, we evaluate the Galerkin integrals using the simplest possible quadrature rule — a single point, located at the center of the triangle — although the solver supports more sophisticated approaches. This is a common approach for estimating energies, especially in quantum chemistry [34], and is adequate for the current demonstration of petFMM. For planar elements and polynomial basis functions, one may also compute some of these integrals analytically [35], [36] rather than by numerical quadrature.

Forming the BEM matrix explicitly requires \( O(n_p^2) \) memory and time, and solving the dense matrix equation \( Ax = b \) using LU factorization requires \( O(n_p^3) \) time. These requirements quickly limit the size of computations that can be performed routinely. One could form the dense matrix and use Krylov-subspace iterative methods, e.g. GMRES [37], to solve the equation approximately in only \( O(n_p^2) \) time, where the dominant cost is then the computation of the dense matrix–vector product required to form the Krylov subspace \( (b, Ab, A^2b, \ldots) \).

To minimize the number of expensive matrix–vector products computed, one often solves the preconditioned system \( PAx = Pb \) for nonsingular \( P \), where \( P \) reduces the number of iterations required by clustering the eigenvalues of \( A \) (that is, it acts in some sense like \( A^{-1} \)) [38]. Then, recognizing that multiplying a vector \( y \) by \( A \) represents the computation of a set of potentials (or fields) at \( n_p \) points due to \( n_p \) sources, we may turn to matrix sparsification/fast summation methods such as the fast multipole method (FMM) to rapidly apply the BEM matrix [14], [15]. The combination of preconditioned Krylov methods with linear- or near-linear time algorithms for computing the dense matrix–vector product leads to \( O(n_p) \) or nearly linear methods in time and memory for solving BEM problems [16], [39].

**C. Fast multipole method**

The fast multipole method is an algorithm that accelerates the computations required in \( N \)-body problems, which are expressed as a sum of the form

\[
f(y_j) = \sum_{i=1}^{N} c_i \mathbb{K}(y_j, x_i). \quad \quad \quad (10)
\]

Here, \( f(y_j) \) represents a field value evaluated at a point \( y_j \), where the field is generated by the influence of sources located at the set of centers \( \{x_i\} \). The evaluation of the field at the centers themselves corresponds to the well-known \( N \)-body problem. Thus \( \{x_i\} \) is the set of source points with weights given by \( c_i \), \( \{y_j\} \) the set of evaluation points, and \( \mathbb{K}(y, x) \) is the kernel that governs the interactions between evaluation and source particles. Obtaining the field \( f \) at all the evaluation points requires in principle \( O(N^2) \) operations, if both sets of points have \( N \) elements each. Fast algorithms aim at obtaining \( f \) approximately with a reduced operation count, ideally \( O(N) \). In our problem, for example, we use two types of fast summation: in the first, we replace the source points with a discretization of the induced charge on the molecular surface \( \sigma \), evaluate the field at the same surface points, and use as the kernel the boundary integral operator in Equation 3. In the second type of fast summation, the source points are the point charges in the molecule and the field is evaluated at points on the surface \( \Omega \).

In the FMM algorithm, the influence of a cluster of particles is approximately represented by a single collective expression, which is then used to evaluate far-away interactions with controllable accuracy. To accomplish this, the computational domain is hierarchically decomposed to increasing levels of refinement, and near and far sub-domains can then be identified at each level. Note that accuracy can only be maintained for kernels whose influence diminishes at long distances, so-called Calderón-Zygmund operators [40], [41]; the inverse of the Laplacian is such an operator. Using the hierarchical spatial decomposition, the total work done to calculate the interaction between charges will be in \( O(N) \).

We will illustrate the phases of the algorithm using a diagram of the tree structure which provides the spatial division (Figure 2), thereby directly relating the algorithm to the data structure used by the FMM. This approach proves useful when
we discuss parallelization of the algorithm. We will also need to introduce terminology for the approximation of the kernel action at long and short distances.

- **Multipole Expansion (ME):** a series expansion truncated after \( p \) terms that represents the influence of a cluster of particles, and is valid at distances large with respect to the cluster radius.

- **Local Expansion (LE):** a truncated series expansion, valid only inside a sub-domain, that is used to efficiently evaluate a group of MEs locally to a cluster of evaluation points.

The computation of the action of the kernel using FMM proceeds in three stages: upward sweep, downward sweep, and evaluation stage. In the upward sweep, the objective is to build the MEs for each node of the tree. The MEs are built first at the tree leaves, level \( L \), and then translated to the center of the parent cells. This is illustrated in Figure 2 by the black arrows pointing up from the nodes on the left side of the tree. Notice that at each level above the leaves, MEs are computed by shifting and then combining the MEs of the child cells, which results in a reduction of the number of expansions by a factor \( 2^L \). In the downward sweep of the tree, the MEs are first transformed into LEs for all the boxes in the interaction list—a process represented by the dashed red-colored arrows in Figure 2. For a given cell, the interaction list corresponds to the cells whose parents are neighbors of the given cell’s parent, and yet not directly adjacent to the given cell. Each LE is then translated to the centers of its child cells, and combined with the transformed MEs from the child level to obtain the complete far field influence for each box. This process is represented by the dashed blue-colored arrows going down the right side of the tree in Figure 2. At the end of the downward sweep, each box will have an LE that represents the complete far-field for the box. Finally, at the evaluation stage, in each leaf cell, the total field is evaluated for every particle it contains by adding the near-field and far-field contributions. The near field contribution comes from directly computing the interactions between all the particles in the adjacent cells. The far field contribution comes from evaluating the LE of the cell at each particle location.

For the current experiments, an expansion order of \( p = 10 \) was used since it achieves sufficient accuracy when measured against very finely discretized models. The FMM operator application has also been validated against the direct \( O(N^2) \) operator application. The present FMM implementation, PeiFMM, uses spherical harmonic rotation before each translation, resulting in an \( O(p^3) \) cost. While this is asymptotically larger than the \( O(p^2) \) achievable with plane wave expansions, in our experiments with \( p = 10 \), this method outperformed plane wave expansions.

Note that the entire tree construction, including calculation of particle assignments, interaction lists, and communication structures, remain the same during the BEM solve, so they may be created once at the beginning of the calculation. In addition, we optimize storage by saving only the expansion coefficients for tree cells which contain particles. This saves an order of magnitude of storage in the present BEM calculation compared to the case where the full (but local) cells are stored.
III. Parallel GPU implementation of the Fast Multipole Method

A. Parallel FMM

In order to apply the boundary integral operator using FMM, we make use of the PETFMM library [19]. Construction of the distributed octree is done by distributing the Local Essential Trees (LET) [42], the subtrees of the global tree that each process needs to evaluate its interaction list. The division is based on work load balancing, with no communication load balancing, by equally distributing the Morton sorted boxes at leaf-level.

Communication occurs only between boxes present in each others interaction lists, since calculations to the root proceed independently on each process. We use non-blocking communication between neighboring nodes in the tree, but we eliminate communication for boxes which do not contain particles. For the case of the M2L transformation, a single rendezvous is necessary to determine how much communication is necessary, and this communication involves only box identifiers. After this, only coefficients for occupied boxes are communicated, again using non-blocking primitives. Our implementation is able to overlap the direct calculation of neighboring particle interactions with the M2L communication, which results in considerable reduction in latency. Finally, with this distribution, memory scaling of our implementation is completely linear in the number of processes.

The P2M, M2M, M2L, L2L, L2P, and P2P kernels were all ported to the GPU architecture using CUDA. The multipole expansions, carried out in spherical harmonics, can be calculated at runtime for each transform. However, we store the intermediate result, the Wigner D matrix, used for the rotation calculations during translation. All kernels are calculated in single precision, expect for the large final summation. For this we use a technique from Narumi [20] to achieve double precision accuracy using only single precision variables. In our tests, this calculation is only 10% slower than a pure single precision calculation, yet has the same accuracy as the CPU kernel calculated completely in double precision. In fact, all GPU kernels were validated against CPU kernels, and against direct operator application as well.

A small modification of our original CPU iteration allows efficient queueing of GPU tasks. Instead of executing a given kernel, e.g. M2L, when using the GPU, the same call queues an equivalent task for the GPU and buffers the input and output data, making memory access on the GPU contiguous. This coalesced read of source points is essential for good performance. In the GPU buffer, output memory is also made contiguous. In addition, the memory for evaluation points is padded to match the size of a thread block. For example, if the number of multipole coefficients is 55 and the thread block size is 64, the code will insert zero padding for indices 56–64. This coalesced write to evaluation points was found to be much more efficient than using a compressed buffer. In all, this queueing time is much less than 1% of the total runtime for our calculations. Moreover, if the task buffer exceeds GPU main memory, it is split and the kernels are executed in a series of calls.

B. Parallel BEM

The system of linear algebraic equations arising from the BEM discretization of Equation 3 is solved using the PETSc
The present system consists of 288 NVIDIA GTX 295 cards, each with two GPUs. There are two cards per host PC, which amounts to 144 CPUs and 576 GPUs. Figure 4 shows the configuration of the interconnect between the nodes. There are 36 nodes connected to each of the 6 QDR switches. The bandwidth of SDR is 10 Gbps and for the QDR it is 40 Gbps. We use 4 QDR networks to achieve a total bandwidth of 160 Gbps between the switches. Each circle represents one compute node equipped with 1 CPU and 4 GPUs. An actual photo of the system in use is shown in Figure 5.

The costs of the constituent elements of our GPU cluster are summarized in Table I. The total price of the system was 38,386,918 JPY, which is equivalent to $ 411,921 (93.19 JPY = $ 1 on April 11th 2010). All prices are inclusive of a sales tax of 5%. The sustained performance of 34.6 TFlops results in a cost performance of 80.0 MFlops/$.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Quantity</th>
<th>Price (JPY)</th>
<th>Price ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPUs</td>
<td>288</td>
<td>14,227,783</td>
<td>$ 152,675</td>
</tr>
<tr>
<td>Host PCs</td>
<td>144</td>
<td>14,535,683</td>
<td>$ 155,979</td>
</tr>
<tr>
<td>Infiniband SQR card</td>
<td>144</td>
<td>2,880,036</td>
<td>$ 30,905</td>
</tr>
<tr>
<td>Infiniband Cables</td>
<td>164</td>
<td>1,924,000</td>
<td>$ 20,646</td>
</tr>
<tr>
<td>Infiniband QDR switches</td>
<td>6</td>
<td>4,819,507</td>
<td>$ 51,717</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>38,386,918</strong></td>
<td><strong>$ 411,921</strong></td>
</tr>
</tbody>
</table>

IV. Computational results

The calculations we report here demonstrate the speed and scalability of petFMM on realistic biomolecular problems, without an attempt at this point of offering biological insights into our chosen examples. Much more detailed simulation and analysis is required to obtain meaningful understanding of these complex systems. Furthermore, it bears repeating that techniques such as explicit-solvent molecular dynamics offer much more detail and are to be preferred in some circumstances, but are not always practical for some types of problems.

We first demonstrate the convergence and accuracy of the petFMM method using as a model problem a clinically relevant protein implicated in cancer and a small-molecule inhibitor.

Finally, we calculate the electrostatics of several multi-million atom systems by creating arrays of molecules of the protein lysozyme. This example is inspired by the pioneering work of McGuffee and Elcock [44], who were among the first to conduct atomistic-level simulations of concentrated protein solutions. Proteins in biological systems, especially inside cells, operate in crowded environments that can strongly influence protein behavior [45]. Computational expense has long been a severe constraint on scientists’ ability to study such systems theoretically. Most investigations use implicit-solvent models in conjunction with Brownian dynamics (see, for example, [46]) to maximize the time scales that can be simulated at reasonable computational cost. Still, however, most studies have of necessity employed reduced models of the proteins (e.g. spheres) despite the evidence that protein shape plays a significant role [47], [48]. Atomistic-level treatments have also been forced to use some approximations to the physics (approximating the electrostatics) to make the computations feasible [49]. To our knowledge, our ability to compute these interactions rigorously in seconds is unprecedented, and may enable more accurate studies than have been possible previously.

The Appendix provides details regarding the preparation of the protein structures as well as the surface discretizations. All calculations were performed with $\epsilon_I = 4$ and $\epsilon_{II} = 80$.

A. Protein–inhibitor binding calculations

Understanding the interactions between inhibitors and their proteins can help lead to the design of more potent drugs with fewer side effects, and computational modeling can be a valuable approach to study inhibitor–protein binding [50]. The first model problem is a protein known as cyclin-dependent kinase 2 (CDK) and a small-molecule inhibitor. Cyclin-dependent kinases are involved in the control of the cell cycle and implicated in the growth of cancers [51], and it is thought that developing tight-binding inhibitors of CDK proteins may be a valuable therapeutic approach to treating some types of tumors. The atomic structure of a CDK2 protein bound to a novel small-molecule inhibitor was solved using X-ray crystallography and deposited in the Protein Data Bank [52] (PDB accession code 1OIT [51]). Figure 6 shows the drug in the binding site.
As described in §II-B, the integral equation 3 is solved using BEM to calculate the contribution to binding affinity that is due to the polarization of the solvent around the protein and drug. We use a widely used, but somewhat simplistic, approach to estimating binding affinity, in which the protein and inhibitor bind rigidly in the conformations shown in the crystal structure. It has been shown that more realistic models, which account for conformational changes on binding, require highly accurate electrostatic simulations for convergence [13]. Thus, PetFMM/GPU may be a valuable tool to improve drug modeling by reducing the computational cost of accounting for flexibility. For these types of calculations, it would be ideal to be able to demonstrate that the computed energies are correct to within 0.1 kcal/mol, which is roughly the accuracy limit of many experimental binding assays.

Figure 7(a), (b), and (c) contain plots of the computed electrostatic solvation free energies for the unbound inhibitor, unbound protein, and inhibitor–protein complex, as functions of the density of vertices on the discretized surface (in vertices per square Angstrom). Figure 7(d) plots the solvation free energy of the complex minus the solvation free energies of the unbound species,

$$
\Delta G_{\text{solv,es,bind}} = \Delta G_{\text{solv,es,complex}} - \Delta G_{\text{solv,es,protein}} - \Delta G_{\text{solv,es,inhibitor}}.
$$

Meshing becomes problematic for the protein and inhibitor–protein complex beyond a vertex density of 10/square Angstrom. It can be seen from Figure 7(d) that much finer discretizations than this would be required to converge the estimated contribution to the binding free energy. Finer discretizations of the inhibitor surface can be generated and we have simulated meshes up to 36 vertices/square Angstrom (at which point the surface is discretized into 19,998 elements), with convergence approaching the desired 0.1 kcal/mol level (Figure 7(a)). This is consistent with earlier work assessing the accuracy of planar elements versus curved elements [53]; however, we note that the point-charge model employed here is bound to be somewhat less accurate than higher-order Galerkin approximations.

These problems are sufficiently small that a single GPU solves the largest protein and complex problems in under 30 seconds. Figure 8 contains plots of the average matrix–vector product time required for the inhibitor–protein complex problems at different vertex densities. As expected, the direct method scales quadratically with problem size and the FMM is approximately linear. Figure 9 is a plot of the surface charge density for the complex, in electrons per square Angstrom.

B. A model calculation for protein solutions

To demonstrate the scalability on large problems, we use collections of randomly oriented lysozyme molecules arranged
Fig. 7: (a)–(c) Convergence of calculated free energies as functions of the density of vertices on the dielectric boundaries for the components indicated. (d) Estimated electrostatic solvation free energy contribution to the inhibitor–protein binding affinity. All energies are in kcal/mol.

Fig. 8: Average times required to compute the dense matrix–vector product using a direct method or the FMM on a single GPU, for the inhibitor–protein complex, as a function of the vertex density.

Fig. 9: The induced charge distribution at the dielectric boundary for the inhibitor–protein complex. Results are in electrons per square Angstrom.
on a regular Cartesian grid, as a mimic of the Brownian dynamics calculations performed by McGuffee and Elcock [44] at each time step. One such collection, of 1000 proteins, is shown in Figure 10. The surface charge density for an isolated lysozyme molecule is plotted in Figure 11. Of course, actually implementing a Brownian-dynamics method requires some special adaptation of BEM [54], [55], which we have not implemented. The calculation did not employ periodic boundary conditions, which would also require further adaptation of the solver.

Figure 12 is a plot of the convergence of GMRES as a function of iteration count for unpreconditioned and preconditioned linear solves, illustrating the effectiveness of the preconditioning scheme, as seen elsewhere [56], [57].

The largest simulation we have conducted consists of 10,648 molecules, where each surface was discretized into 102,486 elements. This calculation, which models more than 20 million atoms and possesses over one billion unknowns, required only approximately one minute per iteration on 512 nodes. The results of a more detailed scalability study of our code is shown in Figure 13. The present code can calculate the matrix-

Fig. 10: Rendering of the discretized surfaces of a collection of 1000 randomly oriented lysozyme molecules spaced on a regular Cartesian grid.

Fig. 11: The induced charge distribution for an isolated lysozyme molecule. The surface has been discretized into 102,486 planar triangles.

Fig. 12: Convergence of GMRES using either no preconditioning or a diagonal preconditioner, for an isolated lysozyme molecule whose surface has been discretized into 102,486 planar triangles.

Fig. 13: Parallel scaling of the FMM implementation for solving large BEM problems. The matrix–vector product times are plotted as a function of the number of points in each problem; the given problems are arrays of randomly oriented points in a cube.
vector product of 6.4 billion elements in 57 seconds. This amounts to a performance of **34.6 TFlops**, as shown in Table II

Our new ability to rigorously simulate such large systems enables the assessment of the heuristics that were previously unavoidable for computational reasons; we emphasize that such assessments should not be perceived as criticism of earlier work but as attempts guide future modeling developments. For example, the electrostatic model of McGuffee and Elcock relies on the assumption that the electrostatic potential generated by each protein does not depend on the presence of surrounding proteins [44]. In the boundary-integral equation formulation, that assumption takes the form of computing the induced surface charge for a single isolated molecule and then employing that surface charge for each molecule in the solution, without solving the BEM problem self-consistently. We can then compare that energy against the actual energy computed by solving the BEM problem.

**V. DISCUSSION**

We have built a boundary-element method (BEM) solver on top of our fast-multipole method (FMM) code with hardware acceleration using GPUs. To demonstrate the solver’s speed and scalability, we have simulated several problems in electrostatics which model the interactions between biological molecules. The combination of optimal algorithms and modern hardware acceleration can lead to a qualitative shift in the methodology of computational protein science, and improve our ability to compute meaningfully converged quantities to be compared with laboratory experiments.

As a first step beyond the results shown above, we will simplify our Helmholtz FMM kernel in order to treat linearized Poisson–Boltzmann problems [10], [12]. The BEM solver could also be used to address numerous other problems, including problems in electromagnetics and the vortex particle formulation fluid dynamics [58]. In addition, we will incorporate higher order discretizations, using either collocation or Galerkin formulations, of the potential in order to improve accuracy. Since the code scales linearly with the number of quadrature points, this addition should result in improved scaling due to the increase in accuracy.

PetFMM with GPU acceleration enables three broad classes of molecular studies that were formerly intractable for routine investigations. First, the speed of PetFMM allows the simulation of much larger assemblies of proteins, or proteins and nucleic acids, than could previously be achieved. Until now, computational constraints necessitated the use of coarse-grained, heuristic models with similarly heuristic physics. Such approximations should no longer be a necessity for studying macromolecular complexes, and the cellular processes in which they participate, at atomic resolution, using the continuum electrostatic model. For example, because simulating a problem with a million unknowns requires much less than a second per iteration on a single GPU-accelerated node (i.e., a computer that as of April 2010 would cost well under $1,000), it is now entirely feasible to conduct detailed electrostatic component analysis of the protein–protein interactions that contribute to ribosome stability or enable specific functions such as translocation of the newly formed polypeptide chain through it [59]. Component analysis studies [60] of such large systems, which require hundreds or thousands of Poisson–Boltzmann simulations, have been prohibitively expensive until now. In fact, to a large degree, PetFMM eliminates actual computation as a limiting factor; the generation of suitable surface meshes, a topic that has received much attention recently [61], [62], appears to be one of the primary remaining challenges.

The acceleration afforded by PetFMM also enables scientists to study molecular flexibility more thoroughly than ever before. Proteins and nucleic acids are not rigid bodies; indeed, most of their properties, such as binding, catalysis, and energy transduction, rest at least partly on their flexibility. Even answering the seemingly simple question of a protein side chain’s affinity to bind a hydrogen ion (known as the pKa value) can require accounting for a surprisingly large (and, computationally speaking, expensive) number of protein conformations [63], [64]. Continuing away from rigidity to the other extreme, flexibility, are the intrinsically disordered proteins [65]. Many of these proteins participate in cellular processes critical to survival, such as transcription of DNA and translation of DNA to RNA. The proteins that give rise to Huntington’s disease, Creutzfeld–Jakob disease, and bovine spongiform encephalopathy (mad cow disease) are also partially intrinsically disordered [66], [67]. Thus, estimating the behavior of a highly flexible protein requires a way to thoroughly sample the different conformations (shapes) that it can take, and calculating the energy of each shape. Molecular dynamics obviously generates a vast number of conformations, but the computational cost of generating sufficient numbers of independent conformations can be again prohibitive. This practical limitation has led to the development of advanced Monte Carlo modeling strategies such as **ABSimTH** [68] for studying flexible polymers. We believe that PetFMM will be a valuable tool in extending our ability to account for biomolecular flexibility.

Third, as the arrays of thousands of lysozyme molecules demonstrated, PetFMM allows the scientist to study complex molecular environments with unprecedented detail without sacrificing model fidelity. For example, currently almost all theoretical studies of molecular binding treat the two binding molecules at “infinite dilution”—that is, the two molecules

<table>
<thead>
<tr>
<th>Description</th>
<th>Equation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of points</td>
<td>( N_p )</td>
<td>( 6.4 \times 10^9 )</td>
</tr>
<tr>
<td>Number of FMM cells</td>
<td>( N_{cell} )</td>
<td>( 8^8 )</td>
</tr>
<tr>
<td>Points per cell</td>
<td>( N_p / N_{cell} )</td>
<td>( 381 )</td>
</tr>
<tr>
<td>Interaction list</td>
<td>( N_{list} )</td>
<td>( 27 )</td>
</tr>
<tr>
<td>Operations per interaction</td>
<td>( K )</td>
<td>( 30 )</td>
</tr>
<tr>
<td>Time to solution</td>
<td>( T )</td>
<td>( 57 )</td>
</tr>
<tr>
<td>Total number of operations</td>
<td>( KN_p N_{list} N_p / N_{cell} / T )</td>
<td>( 3.46 \times 10^{13} )</td>
</tr>
</tbody>
</table>
are surrounded only by molecules of the solvent (water and dissolved ions). Biological systems are far from dilute, however; inside the cell, as much as 30% of the available volume is occupied by other molecules [69]. The precise effects of “molecular crowding,” and its control by the cell, are not yet fully understood, but it is clear that crowding impacts protein diffusion, conformation, and association [45], [70]. The accelerated petFMM enables us to study mixtures of large numbers of proteins and small molecules, and therefore to study crowding in great detail. Note also that our implementation did not employ any optimizations for the multiple protein problem, which could further accelerate the repeated computations that would be performed in an actual Brownian dynamics simulation.

VI. ACKNOWLEDGMENTS

The authors thank B. Tidor for the use of the ICE (Integrated Continuum Electrostatics) software library and B. Roux for the use of CHARMM.

APPENDIX: STRUCTURE PREPARATION

All molecular structures were downloaded from the Protein Data Bank [52] (CDK–inhibitor accession code 1OIT; HIV-1 protease–inhibitor accession code 1T3R; small subunit accession code 3KC4; hen egg-white lysozyme accession code 1HEL). The PSFGEN module of VMD [71] was used to add missing atoms as well as appropriate patches at the N- and C-termini of all peptide chains. For the CDK–inhibitor, HIV protease–inhibitor, and lysozyme examples, atomic radii and partial charges were taken from the PARSE parameter set [72]. All surface meshes were generated using MSMS [73].

REFERENCES
