License

Biomolecular electrostatics using a fast multipole BEM on up to 512 GPU and a billion unknowns

Rio Yokota[a], Jaydeep P. Bardhan[b], Matthew G. Knepley[c], L. A. Barba[a,*], Tsuyoshi Hamada[d]

[a] *Department of Mechanical Engineering, Boston University, Boston MA 02215*
[b] *Dept. of Molecular Biophysics and Physiology, Rush University Medical Center, Chicago, IL 60612*
[c] *Computation Institute, University of Chicago, Chicago, IL 60637*
[d] *Nagasaki University, Advanced Computing Center (NACC), Nagasaki, Japan*

Abstract

We present teraflop-scale calculations of biomolecular electrostatics enabled by the combination of 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, as well as the BIBEE approximation to BEM. 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. The applications demonstrated include the electrostatics of protein–drug binding and several multi-million atom systems consisting of hundreds to thousands of copies of lysozyme molecules. The parallel scalability of the software was studied in a cluster at the Nagasaki Advanced Computing Center, using 128 nodes, each with 4 GPUs. Delicate tuning has resulted in strong scaling with parallel efficiency of 0.8 for 256 and 0.5 for 512 GPUs. The largest application run, with over 20 million atoms and one billion unknowns, required only one minute on 512 GPUs. We are currently adapting our BEM software to solve the linearized Poisson–Boltzmann equation for dilute ionic solutions, and it is also designed to be flexible enough to be extended for a variety of integral equation problems, ranging from Poisson problems to Helmholtz problems in electromagnetics and acoustics to high Reynolds number flow.

Keywords: bioelectrostatics, fast multipole method, boundary element method, integral equations, graphics processors, GPU

1. Introduction

Electrostatic interactions play an essential role in the structure and function of biomolecules (proteins, DNA, cell membranes, *etc.*) [73, 83]. 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 only secondary importance for investigations of interest. Classical molecular dynamics (MD) methods, implemented in software libraries such as CHARMM [17] and NAMD [61], 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 are the most detailed approach to studying biomolecular systems, but computing an "average" electrostatic energy in such systems can be extremely expensive, even when one uses efficient methods for the long-range electrostatics between all the atoms, such as particle-mesh Ewald [26] or multilevel summation [76]. For many studies, a faster method based on adequate approximations is a necessity.

In contrast with all-atom MD computations, one can model the electrostatic interactions in solvated molecules using a continuum representation. Such a model is based on assuming that the molecules and the solvent can be treated as continuous dielectric media with different dielectric constants, low and high, respectively. 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 dielectric boundary 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, a Poisson–Boltzmann model for biomolecular systems is obtained [73]. Standard numerical approaches such as finite-difference methods and

[*] Correspondence to: 110 Cummington St, Boston MA 02215, (617) 353-3883, labarba@bu.edu

Email addresses: yokota@bu.edu (Rio Yokota), jbardhan@alum.mit.edu (Jaydeep P. Bardhan), knepley@ci.uchicago.edu (Matthew G. Knepley), labarba@bu.edu (L. A. Barba), hamada@nacc.nagasaki-u.ac.jp (Tsuyoshi Hamada)

finite-element methods can be used to solve the Poisson or Poisson–Boltzmann equations [84, 37, 7], but there are several challenges that must be overcome. These include the difficulty of 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.*, [18, 34]) and the availability of 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 use a boundary-integral formulation and the boundary-element method, BEM, to determine the induced charge distribution on the molecular surface which accounts for the change in polarization charge across the dielectric boundary. One significant advantage of 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 [45, 90, 50, 3]. 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 Krylov iterative methods such as GMRES [69], the fast multipole method (FMM) [66, 39, 57] can be used to calculate the dense matrix-vector product needed in the iterative solver in $\mathcal{O}(N)$ operations [14]. In this way, the FMM algorithm and similar approaches [62, 2] can enable calculations with hundreds of thousands or even millions of degrees of freedom (*e.g.* [14, 46, 50]).

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 collection of biomolecules, totalling more than 20 million atoms, for which the BEM problem has more than *one billion unknowns.* It was accomplished using a cluster of 128 nodes with a total of 512 GPUs and required approximately one minute to complete. Thus, the electrostatic interactions between large proteins and molecular machines with tens to hundreds 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 biomolecular electrostatics such that the new limiting factor is the generation of suitable meshes for the BEM, rather than fast solver approaches. We wish to emphasize that we are referring to calculations of static structures (for example, as in the post-processing stages of MD-based estimates of binding free energies using MM-PBSA methods [53]) rather than actual dynamics. Substantial hurdles remain before BEM electrostatics can be practical for implicit-solvent dynamics, including both meshing and the calculation of suitable forces [51, 79]. Our solver is currently experimental, but already available publicly via the repository of the open-source PetFMM library[1], which provides a parallel implementation of the FMM with dynamic load balancing [25]. The GPU kernel implementations for the GPU architecture are derived from the 2009 Gordon Bell prize-winning FMM code presented in [40].

There have been some notable recent publications reporting GPU-accelerated algorithms related to our present contribution. The multilevel summation method [76] for calculating electrostatic potentials is implemented for the GPU architecture in [41]. In this algorithm, the short-range interactions are equivalent to the particle-to-particle (P2P) operations in the FMM, while the long-range forces are obtained from hierarchical interpolations; both the short-range interactions, and the dominant part of the long-range ones were implemented as GPU kernels, achieving an overall speed-up of 26× over the CPU version. However, we note that the algorithm is completely dominated by the short-range pair-wise interactions, which on the CPU take 90% of the runtime and on the GPU take 75%, as shown in Table 1 of [41]. In this respect, the approach is quite different from the one followed with the FMM, where one aims for an optimal balance between the $\mathcal{O}(N^2)$ P2P and the multipole-to-local (M2L) transformation (see Figure 2 for definitions of the FMM operations). The demonstration runs in [41] use a 1.5 million atom water box, with a total runtime of 20 s on one GTX 280 GPU.

Another related work to the present contribution was presented in [47], where a variant of the FMM called the kernel-independent fast multipole method [85] is implemented for multi-GPU systems. That algorithm and implementation was demonstrated on an impressive 256 million point-system, taking 2 s to compute the interaction on 256 GPUs. The test consisted of a uniform random distribution of particles in a unit cube (tests with non-uniform distributions are also reported, but on CPU-only systems), and a consistent speed-up measure of 25× is obtained from the GPU on weak scaling tests. This GPU-accelerated kernel-independent FMM has been recently used in an award-winning application to simulation of red-blood cells [64]. Finally, in [80] a GPU-accelerated BEM for the Helmholtz equation is presented and demonstrated up to one million unknowns. However, this work does not use algorithmic acceleration by means of the FMM, performing instead the $\mathcal{O}(N^2)$ method.

Even with linear-scaling algorithms such as multigrid approaches for finite-element methods [7] and FMM for BEM, the computational costs of solving the continuum electrostatic model can be prohibitive, especially when

[1] To obtain the software and documentation, follow the links provided in http://barbagroup.bu.edu/

the scientific questions of interest require the simulation of many millions of unknowns. Such investigations include *in silico* screening of candidate drugs [49], protein design [38, 81], and implicit-solvent MD, in which one computes the trajectories of all protein atoms but replaces the effect of solvent molecules with an appropriate potential of mean force [36]. These applications have driven the development of much faster approximate models such as the popular generalized-Born (GB) models [78] which are frequently used in implicit-solvent molecular dynamics, *e.g.* [31]. Recently, more innovative approaches have also been described, many of which also employ GPUs [32, 4]. In the present paper, we approximate the BEM solution using the new BIBEE (boundary-integral-based electrostatics estimation) model [8, 12], which can be more than an order of magnitude faster than BEM simulation; thus, BIBEE is well-suited for rapid screening of candidate drug molecules (a common application of solvers such as APBS [7]), but like BEM it is not presently applicable to computing dynamics. Modern GB models and implementations have been extensively parameterized and optimized, giving them certain performance and accuracy advantages over early, unoptimized implementations of BIBEE when computing electrostatic solvation free energies. However, BIBEE does offer several of its own advantages, including the fact that it allows scientists to fully leverage well-known numerical techniques such as FMM, rather than necessitating new implementation. The ability to employ sophisticated computational primitives such as FMM as an algorithmic substrate is particularly important in the face of rapidly evolving hardware architectures. Thus, there remains a healthy (and in our view, productive) tension between efforts to develop general fast methods, one of which we apply in this work, and efforts to specialize existing fast methods for particular problems such as biomolecular electrostatics [32, 4].

This contribution aims primarily at demonstrating the power of multiplying speedups: fast linear-scaling algorithms, rigorous approximation of the continuum dielectric model using BIBEE, and hardware acceleration with GPUs. The first wave of successful applications of the GPU in scientific computing was crowded with highly parallel algorithms (like MD) which fit well to the hardware architecture. As could be expected, it is generally much more difficult to obtain high performance with the more elaborate hierarchical and $\mathcal{O}(N)$ algorithms. But it is in this combination where leaps in computing capability are possible which are orders of magnitude larger than what Moore's law would achieve in a given period. In the bioelectrostatics application, the multiplying speedups of algorithm and hardware are further enhanced by a mathematical model that does not lavish computational effort on nonessential degrees of freedom (the solvent molecules). Many applications may still require detailed modeling at the microscale, but where the continuum approach gives sufficient accuracy, the methods and software of our contribution can enable high-impact advances.

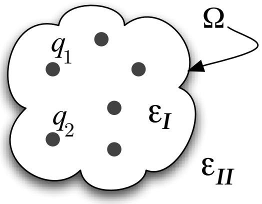


Figure 1: In the continuum, implicit-solvent model for biomolecular electrostatics, the interior of the biomolecule is a dialectric with permittivity ϵ_I and free charges q_i, while the exterior, solvent-filled region is a dialectric material with permittivity ϵ_{II}.

2. Background on the models and algorithm

2.1. Bioelectrostatics using the continuum model

The continuum electrostatic model we treat in this work is a mixed-dielectric Poisson problem, described as follows (see Figure 1). The molecular interior, denoted as region I, is treated as a homogeneous dielectric with permittivity ϵ_I; typical values are between 2 and 10 [73, 71]). The molecular charge distribution is 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,

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

The water surrounding the molecule, corresponding to region II, is also modeled as a homogeneous dielectric, of permittivity ϵ_{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 that there are no fixed or mobile charges. At the dielectric boundary, Ω, the potentials and (in the absence of a free surface charge) the normal components of the electric displacement fields are continuous:

$$\begin{aligned} \varphi_I(r_\Omega) &= \varphi_{II}(r_\Omega) && (2) \\ \epsilon_I \frac{\partial \varphi_I}{\partial n}(r_\Omega) &= \epsilon_{II} \frac{\partial \varphi_{II}}{\partial n}(r_\Omega). && (3) \end{aligned}$$

Here, n 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 known as the molecular surface) [65, 24].

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 [68, 55, 74, 9]:

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

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

$$\varphi^{\mathrm{REAC}}(r)=\int_\Omega\frac{\sigma(r')}{4\pi||r-r'||}dA', \tag{5}$$

such that the total potential $\varphi_I(r)$ is the sum of φ^{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* $\Delta G^{\mathrm{solv,es}}$, and it may be written as

$$\Delta G^{\mathrm{solv,es}}=\frac{1}{2}\sum_{i=1}^{n_c}q_i\varphi^{\mathrm{REAC}}(r_i). \tag{6}$$

Electrostatic solvation free energies and the bare Coulomb energies can be used to estimate quantities such as electrostatic contributions to protein stability [77] or the binding affinity between molecules [19].

One may also investigate the effect of dilute ionic solutions on a solute using either the linearized Poisson–Boltzmann (PB) equation $\nabla^2\varphi_{II}(r)=\kappa^2\varphi_{II}(r)$ to model the potential in the solvent, or the full nonlinear PB equation [72]. The linearized PB problem can also be solved using boundary integral equations [87, 45, 50]. For simplicity in presenting the FMM, we treat only the mixed-dielectric Poisson problem; however, our implementation can also solve Helmholtz problems and we are currently adapting the method for solving linearized Poisson–Boltzmann BEM problems.

The first step in using BEM to solve Equation(4) is to divide the boundary Ω 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 [48, 3], which we intend to explore in future work. The second step in BEM is to define the space of possible solutions; here, we define n_p piecewise-constant functions, the i^{th} of which takes a value of 1 on element i and zero elsewhere. The third step is defining what constraints the approximate solution should satisfy [6, 9, 11]. Galerkin methods enforce the residual to be orthogonal to the basis set, whereas point collocation methods enforce that the residual is exactly zero at specified points on the boundary.

By enforcing n_p constraints, one obtains a square matrix equation

$$Ax=Bq, \tag{7}$$

where B is the n_p-by-n_q dense matrix that maps the point charge values (the vector q) to the normal electric field that they induce at the surface, and A is the n_p-by-n_p dense matrix associated with the second-kind integral operator of Equation(4). The unknown basis-function weights x are found in practice by solving the linear system using Krylov-subspace iterative methods such as GMRES [69] and preconditioning, and instead of computing all the entries of A explicitly one uses fast-summation techniques such as the fast-multipole method (FMM) [66, 39, 57], precorrected-FFT [62], or FFTSVD [2] to compute dense matrix–vector products using only linear, $\mathcal{O}(n_p)$, or near-linear time and memory [27]. After obtaining x, the vector of reaction potentials at the charge locations can be computed as the matrix–vector product $\varphi^{REAC}=Cx$, where C is the n_q-by-n_p matrix resulting from discretizing the Coulomb operator of (5). Therefore, the overall electrostatic solvation free energy can be written as $\frac{1}{2}q^TCA^{-1}Bq$.

In this work, we use a Galerkin approach and the simplest possible quadrature rule — a single point, located at the center of the triangle — although the solver supports more sophisticated approaches. Representing $\sigma(r)$ using point charges is a common approach for estimating energies, especially in quantum chemistry [23], and is adequate for the current demonstration of the GPU-accelerated FMM using BIBEE. For planar elements and polynomial basis functions, one may also compute some of these integrals analytically [42, 59] rather than by numerical quadrature, and implementation of these approaches is underway.

2.2. BIBEE approximation to the continuum model

The recently developed BIBEE (boundary-integral-based electrostatics estimation) models compute approximations to a molecule's electrostatic solvation free energy in the mixed-dielectric Poisson model [8, 12]. BIBEE calculations can be at least an order of magnitude faster than BEM simulation, and the approximations reproduce numerous important characteristics of the actual Poisson model, which make BIBEE an appealing new approach for studying electrostatic interactions within and between large biological molecules.

A BIBEE calculation approximates the solution of the boundary-integral equation approach in the previous section. As described earlier, the BEM approach to calculating the electrostatic solvation free energy requires three steps: the computation of Bq, the normal electric field at the boundary; solving the linear system $Ax=Bq$ to obtain the induced surface charge distribution; and the computation of the reaction potentials, $\varphi^{reac}=Cx$. BIBEE approximates the second step by replacing the electric-field operator in (4) with a scaled identity operator. One therefore obtains an approximate surface charge density $\hat{\sigma}$ rather than the actual solution σ. We use a scale factor of

−1/2, which corresponds to an extremal eigenvalue of the electric-field operator [6]:

$$\left(1-\frac{\epsilon_I}{\epsilon_{II}}\right)\left(\frac{\partial}{\partial n}\sum_{i=1}^{n_c}\frac{q_i}{4\pi||r-r_i||}-\frac{1}{2}\hat{\sigma}(r)\right)=\hat{\sigma}(r). \quad (8)$$

Other useful BIBEE approximations may be obtained by employing 0 or +1/2 as scale factors, which are also important eigenvalues for the integral operator [8, 12].

Numerical BIBEE calculations can be performed using simple modifications of BEM implementations for solving (4). The BIBEE/CFA approximate solvation free energy is written as $\frac{1}{2}q^TCD^{-1}Bq$, where C and B are the same as in BEM, and D is a diagonal matrix [8]. BIBEE's speed advantage over BEM simulation arises because D^{-1} is trivial to apply, whereas application of A^{-1}, which entails the Krylov-iterative solve, is the most computationally expensive step in computing electrostatic solvation free energies using BEM.

Using Eq. (8), the approximate electrostatic solvation free energy for a single point charge (setting all others to zero) is equal to the volume integral of the energy density of the Coulomb field produced by the lone charge, where the volume of integration includes the entire solute volume except for a spherical region associated with the atomic charge in question (which eliminates the singularity [63]). This solvation free energy, because it is associated with setting $q_i = 1$ for some i and the other charges to zero, is known as the i^{th} self energy. The use of such a volume integral to approximate a point charge's self energy is well-known as the Coulomb-field approximation [63], and therefore we refer to Eq. 8 as the BIBEE/CFA model.

BIBEE/CFA provides new insights into the character of the CFA, including that the CFA is exact for charge distributions that generate uniform normal fields on the boundary [8] and that the BIBEE/CFA approximate electrostatic solvation free energy is an upper bound to the actual free energy that would be obtained by solving the Poisson model [12]. Ironically, although the CFA gives a mathematically rigorous approximation of the Poisson problem, until recently it had been relegated to computing parameters for non-rigorous, purely empirical electrostatic models in the generalized-Born methods [78, 63]. In fact, the CFA was originally introduced in volume-integral form to calculate the self-energies of point charges [63]. Despite the unphysical basis of GB methods, they have proven to be effective at approximating solvation free energies. As a result, later research focused on calculating more accurate self-energies (improvements "beyond the Coulomb-field approximation" [67]) rather than on analysis of the CFA. Borgis *et al.* achieved a significant theoretical advance with their variational Coulomb-field approximation [16], which allowed for the first time the treatment of multiple charges in the CFA, eliminating the need for the nonphysical approximations inherent in GB. Several years later, independent analysis of the boundary-integral equation (4) led to the BIBEE models [8]. This latter work was inspired by the observation of the relationship between the CFA volume integral and Equation (4) for a spherical solute with a single central charge [35]. BIBEE/CFA appears to be the surface form of the earlier variational CFA of Borgis *et al.*

2.3. 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

$$f(y_j)=\sum_{i=1}^{N}c_i\,\mathbb{K}(y_j,x_i). \quad (9)$$

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 points. Obtaining the field f at all the evaluation points requires in principle $\mathcal{O}(N^2)$ operations, if both sets of points have N elements each. Fast summation algorithms aim at obtaining f approximately with a reduced operation count, ideally $\mathcal{O}(N)$. In our application, 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 σ, evaluate the field at the same surface points, and use as the kernel the boundary integral operator in Equation 4. 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 Ω.

There are two main aspects to the FMM algorithm: one is related to the spatial hierarchy and tree data structure; the other, to the mathematical machinery needed to perform approximations which reduce the number of operations in exchange for accuracy. The spatial hierarchy refers to a successive sub-division of the three-dimensional spatial domain and construction of an associated oct-tree: the domain is initially divided in eight cells, which are then divided again, and so on, until the finest level of refinement or "leaf level". The set of cells created by each sub-division is associated to the nodes at a given level of a tree structure, such that near- and far-domains to each cell can be found by operations on the tree. The source points belonging to cells in the far-domain are then considered as a cluster, with their influence represented by a series expansion in the process of evaluation. This representation of many sources collectively by series expansions allows the reduction of operations in the FMM to $\mathcal{O}(N)$, with controllable accuracy.

One can illustrate the phases of the algorithm using a diagram of the tree associated to the spatial division, thereby directly relating the algorithm to the data structure used by the FMM; see Figure 2. First, we need to

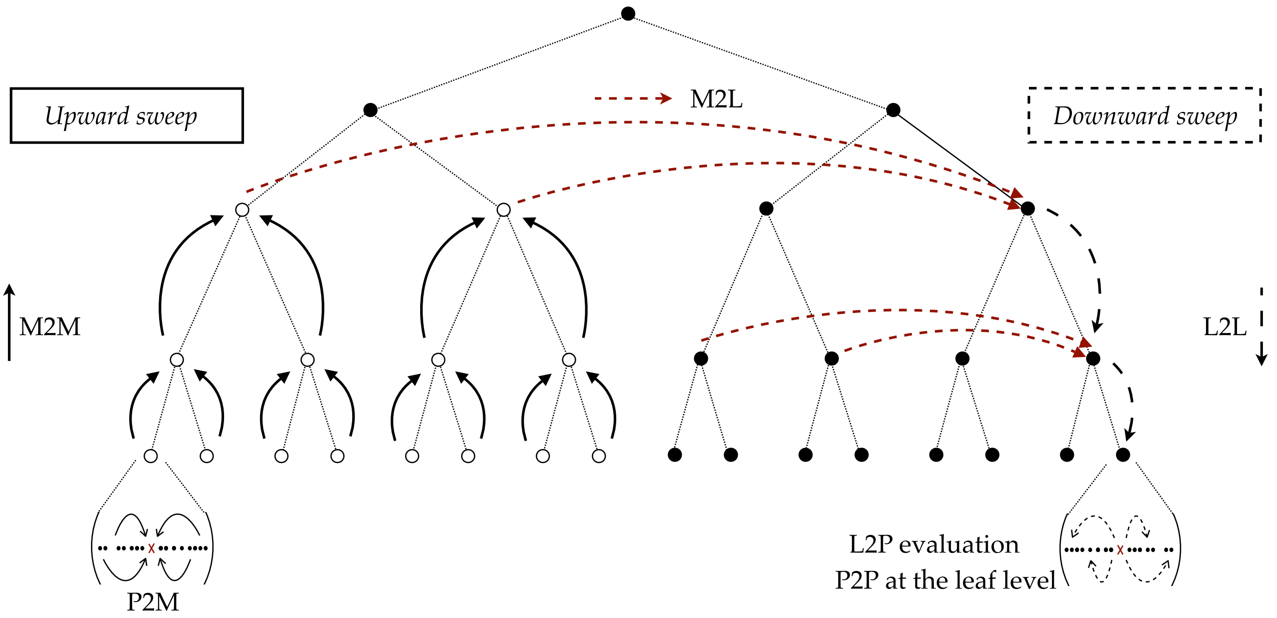


Figure 2: On a tree diagram, we can illustrate the organization of the algorithmic stages for the FMM: *upward sweep*, *downward sweep*, and *evaluation step*. The upward sweep combines the P2M and M2M operations, downward sweep the M2L and L2L, and the evaluation combines L2P with a direct calculation in the near domain, P2P.

introduce the terminology for the approximation of the kernel action at long and short distances, as follows.

- ▹ *Multipole Expansion (*ME*)*: a series expansion truncated after p terms that represents the influence of a cluster of source points, 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 stages: an *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 (P2M operation) and then translated to the center of the parent cells (M2M translation). This process is illustrated in Figure 2 by the black arrows pointing up from the nodes on the left side of the figure. 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 by a factor 8 of the number of expansions that needs to be calculated. 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, and denoted by M2L. 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 all child cells (L2L operation), and combined with the transformed MEs from the child level to obtain the complete far-domain influence for each box. This process is represented by the dashed arrows going down the right side of the tree in Figure 2. At the end of the *downward sweep*, each cell will have an LE that represents its complete far field. Finally, at the *evaluation stage*, the field is evaluated for every point contained in a cell by adding the near-field and far-field contributions: the near field contribution is obtained from directly computing the interactions between all the points in the adjacent cells, and the far-field contribution comes from evaluating the LE of the cell at each point location.

3. Multi-gpu fast multipole method

3.1. Brief overview of the distributed FMM

The distributed-memory parallelization of the FMM involves two main phases: *(i)* the partitioning of the tree among MPI processes; and, *(ii)* the communication of needed data to perform the FMM interactions. In the first phase, the standard and most commonly used technique is to equally distribute the Morton-indexed boxes at the leaf level [82]. This has been the approach used here.

The second phase involves the communication that occurs between partitioned domains, due to interacting cells residing in different processes. Two cells interact when they are present in each other's neighbor list or interaction list (for the far-field), as defined in the previous section. In the case of the neighbors, communication only occurs among processes holding geometrically adjacent points,

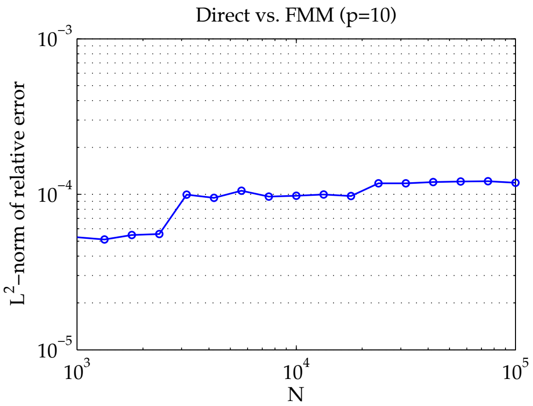


Figure 3: L^2-norm of the measured normalized error for the FMM evaluation *versus* direct evaluation on the GPU.

and data consist of position and weights of the points or particles. For the far-field, the data that needs to be communicated consists of ME coefficients of the cells in the interaction list, at every level of the tree. We overlap these communications with local computations, which results in effective hiding of communication time thereby enhancing scaling (as seen in the results section). Moreover, we have achieved linear scaling in memory requirements, which allows our large-scale computations of biomolecular electrostatics.

3.2. Synopsis of GPU *hardware utilization*

All the computational kernels of the FMM depicted in Figure 2 were reformulated to utilize the GPU architecture with CUDA [60], using single precision. The GPU kernels were verified against CPU calculations, and full FMM evaluations were also validated with respect to direct (all-pairs) evaluation. Figure 3 shows the measured error (L^2-norm of the relative error) for a set of calculations with various numbers of points, comparing the FMM on GPU with direct evaluation.

One particular feature of the GPU execution that we've incorporated in the code is an effective buffering of input and output data to each kernel, such that a kernel is always executed on sufficiently large data sets. In other words, a given kernel, *e.g.*, M2L, is not executed for each of the interactions, but for a group of them. At the same time, data is made contiguous in memory before the kernel call and output memory is also made contiguous in the GPU buffer; these are essential measures to attain good performance. On the GPU, 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.

4. Computational results

4.1. Hardware

The calculations were performed using the *Degima* cluster at the Nagasaki Advanced Computing Center, which presently consists of 288 nVIDIA GTX 295 cards, each with two GPUs. There are two cards per host node, amounting to 144 CPUs and 576 GPUs—however, in this work we have only used 128 nodes out of the total 144, as some of the nodes remain in experimental use. 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. With 4 QDR networks, a total bandwidth of 160 Gbps is achieved between the switches. Each circle in the Figure represents one compute node equipped with 1 CPU and 4 GPUs.

4.2. Overview of computational experiments

The calculations we report here demonstrate the speed and scalability of our FMM algorithm on realistic biomolecular problems, without any attempt at this point of offering biological insights into our chosen examples. Much more detailed simulation and analysis are 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 present the scalability of the FMM on a large number of GPUs to demonstrate the computational power of the present method. Subsequently, we study the convergence and accuracy of the FMM-based BEM approximation BIBEE using as a model problem a clinically relevant protein (implicated in cancer) bound to a small-molecule inhibitor. We then 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 [54], who were among the first to conduct atomistic-level simulations of concentrated protein solutions. This kind of large-scale, computationally demanding study is important because proteins in biological systems, especially inside cells, operate in crowded environments that can strongly influence protein behavior [56]. 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, [33]) 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 evidence that protein shape plays a significant role [58, 21]. Atomistic-level treatments have also been forced to use some approximations to the physics

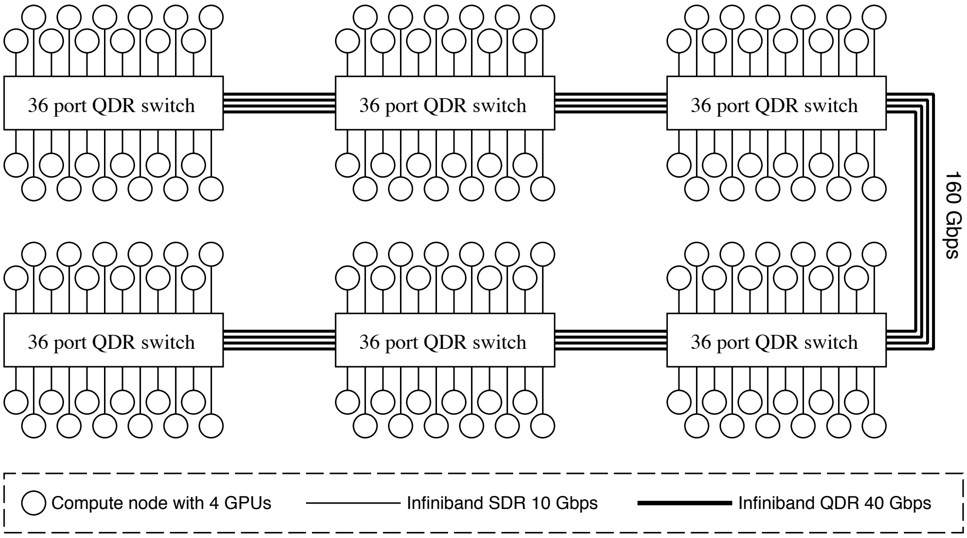


Figure 4: Infiniband network configuration of the *Degima* cluster in Nagasaki.

(approximating the electrostatics) to make the computations feasible [29]. 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 $\epsilon_{II} = 80$.

4.3. Scalability of the FMM *on a* GPU *cluster*

In this section, we present and discuss results from a scalability study of the FMM on many GPUs. We investigate scalability under the more severe *strong scaling* scenario, where the amount of computational work in the experiments remains constant as the number of computing devices (here, GPU chips) is increased from 1 to 512. In addition to parallel efficiency, we analyze the breakdown of execution time by computational kernels, and also by CUDA states. The results show excellent parallel efficiency up to 256 GPUs, and subsequent decline as serial fraction and communication overheads start to dominate. Note that the 256 GPU chips offer a total of 61,440 CUDA cores; the case on 512 GPUs is in actual fact running several million simultaneous threads.

It must be emphasized that achieving high parallel efficiency on a cluster of GPUs is generally much more difficult than on a cluster of CPUs. As pointed out above, each GPU chip is already a parallel device, in this case (GT200b GPU) offering 240 multi-threaded cores. This results in a high compute capability which makes it very difficult to hide the time required for communications. Moreover, the current technology does not provide an avenue for direct communication between GPU cards, and all data must be transfered to and from the host CPU first; communication between nodes then occurs via standard MPI calls. Despite these serious challenges, we can report excellent parallel efficiency, as already mentioned.

Experimental setup. The scalability study is based on a fixed-size problem with $N = 10^8$ points placed randomly in a cubic volume. The order of the multipole expansions was set to $p = 10$, which results in 4 significant digits of accuracy, and spherical harmonic rotations are performed before each translation at $\mathcal{O}(p^3)$ cost. One FMM evaluation is performed (equivalent to one matrix-vector product, or one BIBEE evaluation, or the cost of one iteration in a BEM solution) for each case, with one MPI process running per GPU. We run one MPI process per cluster node up to 128 processes, and then two and four processes per node, respectively, in the 256- and 512-GPU cases (recall that each node has two dual-GPU cards).

Results. As shown in Figure 5, parallel efficiency remains perfect up to 128 GPUs, is 78% for 256 GPUs, and 48% for 512 GPUs. We observe "super-linear" scaling between 4 and 32 GPUs, which we ascribe to better cache usage in the tree construction phase of the calculation, done on the CPU. This is evidenced by the breakdown of time shown in Figure 6, where the tree construction takes less per-process time when going from 1 to 2 and 4 GPUs.

Figure 6 is a bar plot of the runtime multiplied by the number of GPUs (number of MPI processes), where each bar is also color-coded to indicate the time spent in each computational kernel, including tree construction (done on CPU) and MPI communications. Equal bar heights would indicate perfect strong scaling, so once again we observe super-linear scaling for 4–32 processes and efficiency decline for more than 128 GPUs. The breakdown helps to make these observations: tree construction on the CPU is overburdened on one process, possibly due to cache misuse; as expected, the P2P and M2L kernels take the largest

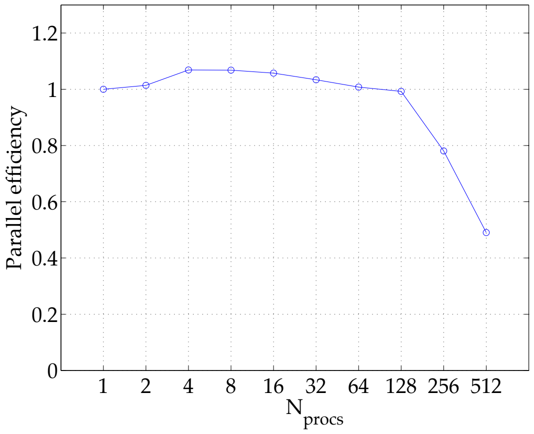


Figure 5: Parallel efficiency in strong scaling of the FMM on multi-GPUs; $N = 10^8$.

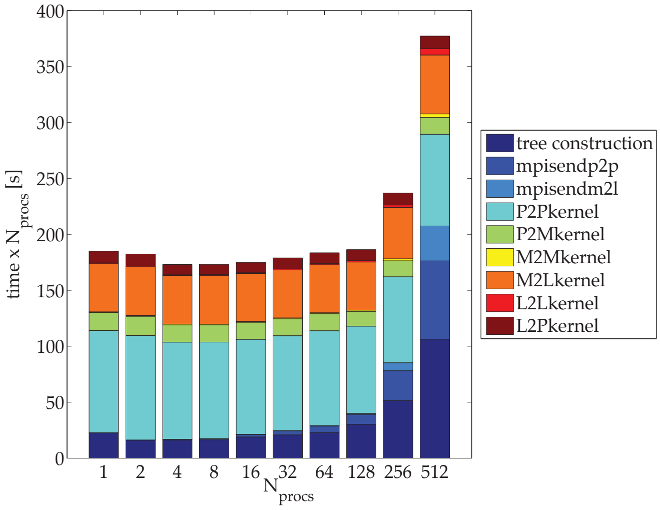


Figure 6: Breakdown of the FMM calculation time corresponding to the runs in Figure 5, with $N = 10^8$.

fractions of runtime; the fraction of time for MPI communications is minor, up until 256 processes; and, tree construction and communications become significant at 512 processes.

The tree construction consists of all the preprocessing required to execute the FMM kernels, including the Morton indexing and bookkeeping of the interaction list, taking as input the set of points already assigned to corresponding processes. The MPI communication time, on the other hand, consists of the time required to update the data of points in the local essential tree ("mpisendp2p") and the time required to update the multipole expansions ("mpisendm2l"). In sum, the height of each bar in Figure 6 is the total wall-clock time it takes to perform one complete evaluation using the FMM (multiplied by the number of processes).

The breakdown of the runtime can be also done by the phases of CUDA execution, as shown in Figure 7, rather than computational kernels. Of course, the communication time and tree construction on the CPU do not change in this version; only the fraction of time between the dashed lines in the Figure is analyzed differently. We have one large fraction of time (maroon color) labeled "cudaKernel"; this is the cumulative time spent in CUDA execution state—expressed in the CUDA code by the triple triangular brackets—for all the 6 computational kernels (shown in the left bar) combined. The FMM evaluation spends 70% of the total execution time in this state. Next in importance is the fraction spent "buffering data" (in green), which refers to the reordering of data that is performed to improve coarse-grained data locality when executing the GPU kernels. The required data is streamlined into a buffer in the exact order that it will be accessed inside the GPU. Next, but already a minor fraction of the total time, is "cudaMemcpy" (red), a label which is given to the total time of all data movement between the host memory and the device memory. This is the most important observation to make from the breakdown of timings in this manner: the total time of all data movement between GPU and CPU is a small fraction of the total time, whereas a large fraction of time is spent in the CUDA execution state. This is a favorable situation in terms of GPU performance, and shows that the FMM, unlike many algorithms, can afford to move data to/from the GPU due to a high compute/data ratio.

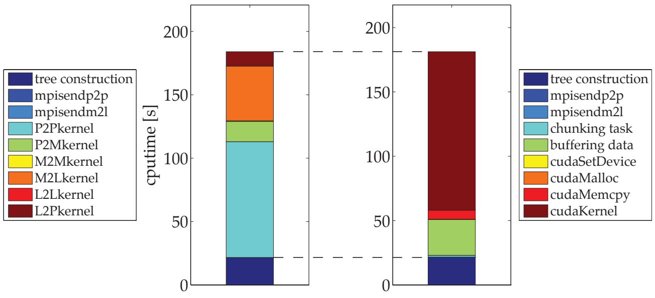


Figure 7: Breakdown of the FMM calculation time corresponding to the run with $N_{procs} = 1$ in Figure 6, with $N = 10^8$.

Discussion. We have shown strong scaling results, demonstrating excellent parallel efficiency of the FMM up to 256 GPUs and acceptable efficiency at 512 GPUs. These results were obtained in an experimental cluster using gaming GPU cards[2]. The configuration of the cluster has two dual-chip GPU cards per node, yet we used one GPU per node up to 128 nodes to allow for the whole bandwidth to be available to each MPI process. In the 256- and 512-GPU cases, the bandwidth must be shared within the node, and so we applied a strategy to alleviate possible constraints due to this. A hierarchical all-to-all communication was implemented using `MPI_Comm_split`, dividing the MPI communicator in 4. Effectively we are splitting the communicator into an inter-node and an intra-node communicator.

[2] The Degima cluster enabled some of the authors to be awarded the Gordon Bell prize in the price/performance category in 2009.

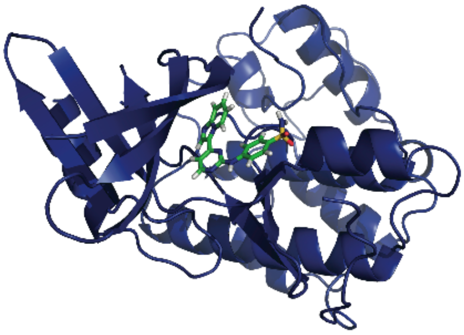


Figure 8: Rendering of CDK 2 bound to small molecule inhibitor, using atomic coordinates from PDB accession 1OIT [5] and the software PyMOL.

Thus, we first perform an intra-node `MPI_Alltoallv`, then an inter-node `MPI_Alltoallv`. We found that this strategy was able to speed-up the communication nearly 4 times at 512 processes; and these are the final results presented in Figures 5 and 6.

4.4. 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 [44]. The first model problem that we will use here 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 [5], 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 CDK 2 protein bound to a novel small-molecule inhibitor was solved using X-ray crystallography and deposited in the Protein Data Bank [13, 5] (see Appendix). Figure 8 shows the drug in the binding site.

As described in §2.2, the integral equation (4) is approximated using BIBEE/CFA to estimate the contribution to binding affinity that is due to the polarization of the solvent around the protein and drug. We use a widely accepted, 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 [3]. Thus, our solver with its GPU capability 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 of many experimental binding assays.

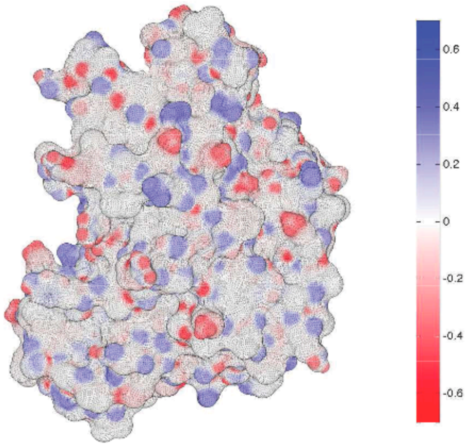


Figure 10: The induced charge distribution at the dielectric boundary for the inhibitor–protein complex. Results are in electrons per square Angstrom.

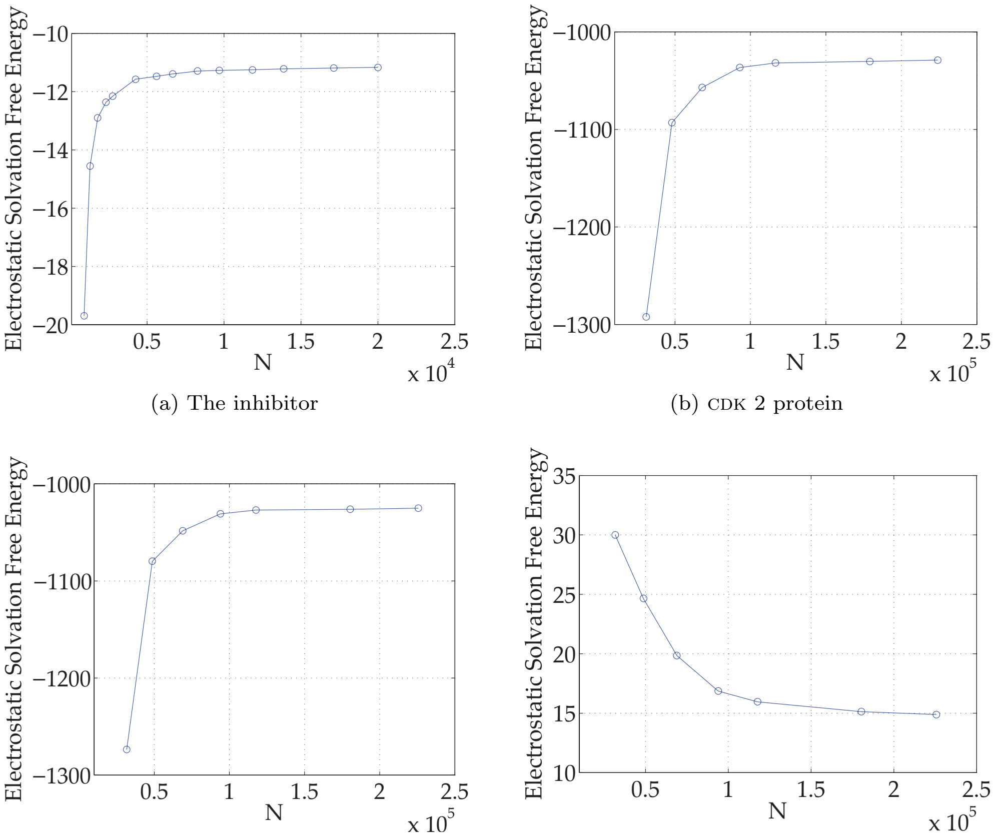
Figures 9(a), (b), and (c) contain plots of the computed electrostatic solvation free energies for the unbound inhibitor, unbound protein, and inhibitor–protein complex, as functions of the number of vertices N on the discretized surface. Figure 9(c) plots the solvation free energy of the complex minus the solvation free energies of the unbound species, *i.e.*,

$$\Delta G^{\text{solv,es}}_{\text{bind}} = \Delta G^{\text{solv,es}}_{\text{complex}} - \Delta G^{\text{solv,es}}_{\text{protein}} - \Delta G^{\text{solv,es}}_{\text{inhibitor}} \qquad (10)$$

Meshing becomes problematic for the protein and inhibitor–protein complex beyond a vertex density of 10/square Angstrom, corresponding to the last data point on Figure 9(d). 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. This is consistent with earlier work assessing the accuracy of planar elements versus curved elements [10]. However, we note that here we have used a point-charge approximation of the molecular charge distribution, rather than Galerkin or collocation BEM, and have also used the BIBEE approximation rather than full BEM calculation. Figure 10 shows a plot of the calculated surface charge density for the complex, in electrons per square Angstrom.

4.5. A model calculation for protein solutions

To demonstrate the capability of the method and software on large problems, we use collections of randomly oriented lysozyme molecules arranged on a regular Cartesian grid, mimicking the Brownian dynamics calculations performed by McGuffee and Elcock [54] at each time step. In practice, calculations on such length scales may benefit from alternative theories that are very efficient for these



(a) The inhibitor

(b) CDK 2 protein

(c) Inhibitor–protein complex

(d) Solvation free energy

Figure 9: (a)–(c): Convergence of calculated free energies as functions of the number of vertices N on the dielectric boundaries for the components indicated. (d): Estimated electrostatic solvation free energy contribution to the inhibitor–protein binding affinity; the number of vertices N is that in the inhibitor–protein complex mesh. All energies are in kcal/mol.

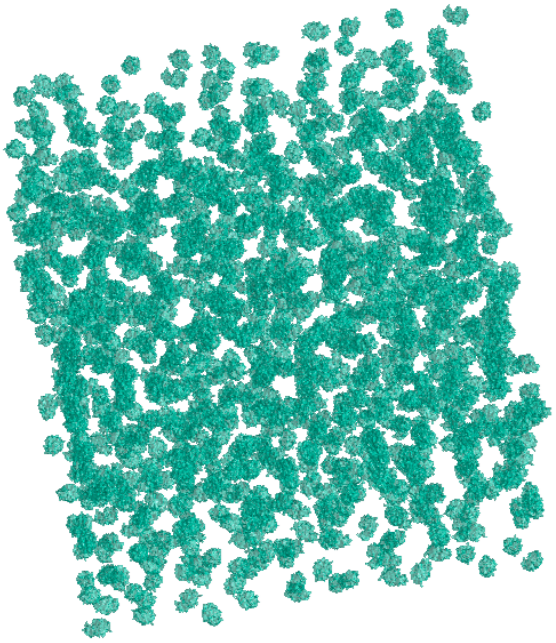


Figure 11: Rendering of the discretized surfaces of a collection of 1000 randomly oriented lysozyme molecules quasi-scattered from a regular Cartesian grid.

specialized problems, for instance the recent approaches of Onufriev *et al.* [32, 4], but for the present the size of the problem showcases the implementation's performance and parallel scaling.

A collection of 1000 proteins is shown in Figure 11. The surface charge density for an isolated lysozyme molecule, computed using our method, is plotted in Figure 12. Of course, actually implementing a Brownian-dynamics method requires some special adaptation of BEM [15, 52], which we have not implemented. The calculation did not employ periodic boundary conditions, which would also require further adaptation of the solver[86].

Figure 13 shows the computation times required for single-GPU BIBEE calculations of lysozyme arrays with different numbers of proteins. As expected, the direct method scales quadratically with the number of boundary elements, and the FMM scales linearly. Note that BIBEE **on a single gpu requires less than one second to compute electrostatics with up to one million unknowns**. 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 about one minute per BEM iteration on 512 nodes.

Our new ability to rigorously simulate such large systems enables the assessment and improvement of current heuristics. For example, one heuristic assumes that the electrostatic potential generated by each protein does not depend on the presence of surrounding proteins, an as-

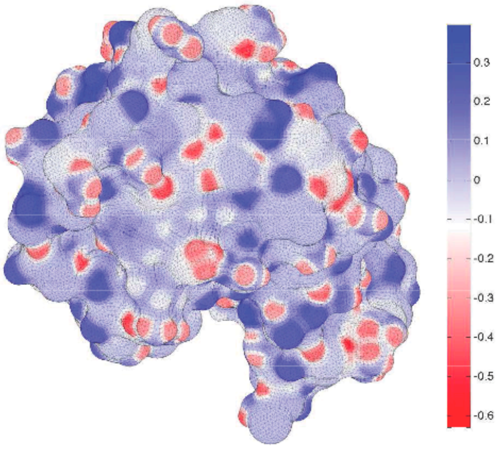


Figure 12: The induced charge distribution for an isolated lysozyme molecule. The surface has been discretized into 102,486 planar triangles.

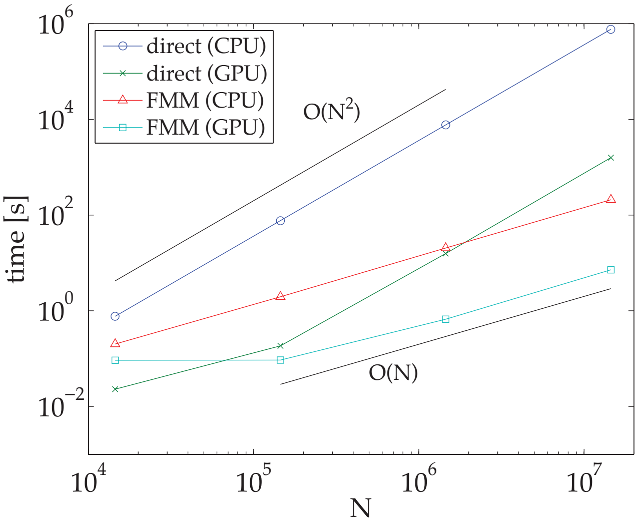


Figure 13: Average times required to compute the dense matrix–vector products for the BIBEE model, using a direct method or the FMM on a single GPU, as a function of the total number of boundary elements in lysozyme arrays; each lysozyme surface was discretized into 102,486 boundary elements.

sumption motivated by the high dielectric solvent, which might also contain mobile ions. In the boundary-integral equation formulation, this heuristic 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 full BEM problem self-consistently. Our fast solver raises the possibility of more detailed checks of such assumptions, by enabling computationally feasible comparison between the heuristic energy and the actual energy computed by realizing a full BEM solution.

5. Discussion

We have built a boundary-element method (BEM) solver using the BIBEE approximation on top of a 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, but it can also be used to address numerous other problems, including problems in electromagnetics and the vortex particle formulation in fluid dynamics. 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 experiments.

The performance demonstrated here will carry over to more sophisticated BEM applications with better accuracy, so we emphasize that the present work should be seen as a demonstration of the performance offered by the algorithms and hardware, rather than as a demonstration of the particular electrostatic energies calculated by the present implementation. Improved methods include the use of collocation or Galerkin formulations [9] or curved rather than planar elements [48, 3]. The present paper provides motivation to incorporate such methods into our solver for fast, accurate computation of molecular electrostatics. As a second step beyond the implementation described here, we will extend our FMM kernels in order to treat linearized Poisson–Boltzmann problems [45, 50].

In summary, substantial development is needed before the solver can be thought of as mature enough to be used for scientific studies even for simulations of fixed structures. Even with FMM on GPU it seems that the possibility of BEM-based implicit-solvent molecular dynamics remains quite distant, owing to the numerous unresolved theoretical issues. When coupled with higher-order BEM techniques, however, the fast multipole method with GPU acceleration should enable the routine calculation of quantities that were previously impractical due to computational cost. For example, it is now entirely feasible to conduct detailed electrostatic component analyses of protein–protein interactions in vital biological processes. Such studies [20] requiring thousands or tens of thousands of simulations, have been prohibitively expensive until now. To a large degree, the FMM eliminates actual computation as a limiting factor; the generation of suitable surface meshes, a topic that has received much attention recently [22, 89], appears to be one of the primary remaining challenges. In addition, the acceleration afforded by our FMM on GPU also makes it practical to sample many more protein conformations in accounting for molecular flexibility [88, 1, 28]. Finally, as the arrays of thousands of lysozyme molecules demonstrated, our method could allow scientists to study complex molecular environments with greater detail—for instance, biological solutions are quite "crowded" [30] and it is not yet clear how crowding impacts protein diffusion, conformation, and association [56, 91]. The GPU accelerated fast multipole method allows studying mixtures of large numbers of proteins and small molecules, and therefore to study crowding in more detail.

The version of the code used for this work unit is currently available in an experimental branch of the PetFMM repository. Over time, functionality for BEM will be added to the main branch of PetFMM, but we make available our experimental code at the time of publication of the paper, consistent with the research group's policy of openness. To find the repository, follow the links provided in http://barbagroup.bu.edu/.

6. 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. MGK supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-09-0488. LAB acknowledges partial support from NSF grant OCI-0946441 and from Boston University College of Engineering.

Appendix: Structure Preparation

All molecular structures were downloaded from the Protein Data Bank [13] (CDK 2–inhibitor accession code 1OIT; hen egg-white lysozyme accession code 1HEL). The PSFGEN module of VMD [43] was used to add missing atoms as well as appropriate patches at the N- and C-termini of all peptide chains. Atomic radii and partial charges were taken from the PARSE parameter set [75]. All surface meshes were generated using MSMS [70].

References

[1] E. G. Alexov and M. R. Gunner. Incorporating protein conformational flexibility into the calculation of pH-dependent protein properties. *Biophys. J.*, 72(5):2075–2093, 1997.

[2] M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. FFTSVD: A fast multiscale boundary-element method solver suitable for BioMEMS and biomolecule simulation. *IEEE Trans. Comput. Aid. D.*, 25:274–284, 2006.

[3] M. D. Altman, J. P. Bardhan, J. K. White, and B. Tidor. Accurate solution of multi-region continuum electrostatic problems using the linearized Poisson–Boltzmann equation and curved boundary elements. *J. Comput. Chem.*, 30:132–153, 2009.

[4] R. Anandakrishnan, T. R.W. Scogland, A. T. Fenley, J. C. Gordon, W.-C. Feng, and A. V. Onufriev. Accelerating electrostatic surface potential calculation with multi-scale approximation on graphics processing units. *J. Mol. Graph. Model.*, 28(8):904–910, 2010.

[5] M. Anderson, J. Beattie, G. Breault, J. Breed, K. Byth, J. Culshaw, R. Ellston, S. Green, C. Minshull, R. Norman, R. Pauptit, J. Stanway, A. Thomas, and P. Jewsbury. Imidazo[1,2-a]pyridines: A potent and selective class of cyclin dependent kinase inhibitors identified through structure-based hybridization. *Bioorg. Med. Chem. Lett.*, 13:3021, 2003.

[6] K. E. Atkinson. *The Numerical Solution of Integral Equations of the Second Kind.* Cambridge University Press, 1997.

[7] N. A. Baker, D. Sept, M. J. Holst, and J. A. McCammon. Electrostatics of nanoysystems: Application to microtubules and the ribosome. *P. Natl. Acad. Sci. USA*, 98:10037–10041, 2001.

[8] J. P. Bardhan. Interpreting the Coulomb-field approximation for Generalized-Born electrostatics using boundary-integral equation theory. *J. Chem. Phys.*, 129(144105), 2008.

[9] J. P. Bardhan. Numerical solution of boundary-integral equations for molecular electrostatics. *J. Chem. Phys.*, 130:094102, 2009.

[10] J. P. Bardhan, M. D. Altman, J. K. White, and B. Tidor. Numerical integration techniques for curved-panel discretizations of molecule–solvent interfaces. *J. Chem. Phys.*, 127:014701, 2007.

[11] J. P. Bardhan, R. S. Eisenberg, and D. Gillespie. Discretization of the induced-charge boundary integral equation. *Phys. Rev. E*, 80(011906), 2009.

[12] J. P. Bardhan, M. G. Knepley, and M. Anitescu. Bounding the electrostatic free energies associated with linear continuum models of molecular solvation. *J. Chem. Phys.*, 130(10):104108, 2009.

[13] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Res.*, 28(1):235–242, 2000.

[14] R. Bharadwaj, A. Windemuth, S. Sridharan, B. Honig, and A. Nicholls. The fast multipole boundary element method for molecular electrostatics: An optimal approach for large systems. *J. Comput. Chem.*, 16(7):898–913, 1995.

[15] A. J. Bordner and G. A. Huber. Boundary element solution of the linear Poisson–Boltzmann equation and a multipole method for the rapid calculation of forces on macromolecules in solution. *J. Comput. Chem.*, 24(3):353–367, 2003.

[16] D. Borgis, N. Lévy, and M. Marchi. Computing the electrostatic free-energy of complex molecules: the variational Coulomb field approximation. *J. Chem. Phys.*, 119(6):3516–3528, 2003.

[17] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[18] R. E. Bruccoleri, J. Novotny, M. E. Davis, and K. A. Sharp. Finite difference Poisson–Boltzmann electrostatic calculations: increased accuracy achieved by harmonic dielectric smoothing and charge antialiasing. *J. Comput. Chem.*, 18(2):268–276, 1997.

[19] J. A. Caravella, Jeffrey D. Carbeck, D. C. Duffy, G. M. Whitesides, and B. Tidor. Long-range electrostatic contributions to protein-ligand binding estimated using protein charge ladders, affinity capillary electrophoresis, and continuum electrostatic theory. *J. Am. Chem. Soc.*, 121:4340–4347, 1999.

[20] N. Carrascal and D. F. Green. Energetic decomposition with the generalized-Born and Poisson-boltzmann solvent models: Lessons from association of G-protein components. *J. Phys. Chem. B*, 114(15):5096–5116, 2010. PMID: 20355699.

[21] R. C. Chang, D. Asthagiri, and A. M. Lenhoff. Measured and calculated effects of mutations in bacteriophage T4 lysozyme on interactions in solution. *Proteins*, 41:123–132, 2000.

[22] H.-L. Cheng and X. Shi. Quality mesh generation for molecular skin surfaces using restricted union of balls. *Comp. Geom.-Theor. Appl.*, 42(3):196–206, 2009.

[23] D. M. Chipman. Charge penetration in dielectric models of solvation. *J. Chem. Phys.*, 106:10194–10206, 1997.

[24] M. L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, 16:548–558, 1983.

[25] F. A. Cruz, M. G. Knepley, and L. A. Barba. PetFMM —-a dynamically load-balancing parallel fast multipole library. *Int. J. Num. Meth. Engineering*, 85(4):403–428, 2010.

[26] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An $N \log N$ method for Ewald sums in large systems. *J. Chem. Phys.*, 93:10089–10092, 1993.

[27] W. Dijkstra and R. M. M. Mattheij. The condition number of the BEM-matrix arising from Laplace's equation. Technical report, OAI Repository of the Technische Universiteit Eindhoven (TU/e) [http://cache.libr.tue.nl:1972/csp/dare/DARE.Repository.cls] (Netherlands), 2006.

[28] H. J. Dyson and P. E. Wright. Intrinsically unstructured proteins and their functions. *Nat. Rev. Mol. Cell. Biol.*, 6:197–208, 2005.

[29] A. H. Elcock. Molecular simulations of diffusion and association in multimacromolecular systems. *Meth. Enzymol.*, 383:166–198, 2004.

[30] R. J. Ellis. Macromolecular crowding: an important but neglected aspect of the intracellular environment. *Curr. Opin. Chem. Biol.*, 11(1):114–119, 2001.

[31] H. Fan, A. E. Mark, J. Zhu, and B. Honig. Comparative study of generalized Born models: protein dynamics. *P. Natl. Aca. Sci. USA*, 102(19):6760–6764, 2005.

[32] A. T. Fenley, J. C. Gordon, and A. Onufriev. An analytical approach to computing biomolecular electrostatic potential. I. derivation and analysis. *J. Chem. Phys.*, 129(7):075101, 2008.

[33] R. R. Gabdoulline and R. C. Wade. Simulation of the diffusional association of barnase and barstar. *Biophys. J.*, 72:1917–1929, 1997.

[34] W. H. Geng, S. N. Yu, and G. W. Wei. Treatment of charge singularities in implicit solvent models. *J. Chem. Phys.*, 127:114106, 2007.

[35] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, 102:10983–10990, 1998.

[36] M. K. Gilson, J. A. McCammon, and J. D. Madura. Molecular-dynamics simulation with a continuum electrostatic representation of the solvent. *J. Comput. Chem.*, 16(9):1081–1095, 1995.

[37] M. K. Gilson, A. Rashin, R. Fine, and B. Honig. On the calculation of electrostatic interactions in proteins. *J. Mol. Biol.*, 184:503–516, 1985.

[38] D. F. Green and B. Tidor. Design of improved protein inhibitors of HIV-1 cell entry: Optimization of electrostatic interactions at the binding interface. *Proteins*, 60:644–657, 2005.

[39] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[40] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji. 42 TFlops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, New York, NY, 2009. ACM.

[41] D. J. Hardy, J. E. Stone, and K. Schulten. Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Comput.*, 35:164–177, 2009.

[42] J. L. Hess and A. M. O. Smith. Calculation of non-lifting potential flow about arbitrary three-dimensional bodies. *J. Ship Res.*, 8(2):22–44, 1962.

[43] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. *J. Mol. Graphics*, 14(1):33–38, 1996.

[44] W. L. Jorgensen, J. P. Ulmschneider, and J. Tirado-Rives. Free energies of hydration from a generalized Born model and an all-atom force field. *J. Phys. Chem. B*, 108:16264–16270, 2004.

[45] A. H. Juffer, E. F. F. Botta, B. A. M. van Keulen, A. van der Ploeg, and H. J. C. Berendsen. The electric potential of a macromolecule in a solvent: A fundamental approach. *J. Comp. Phys.*, 97(1):144–171, 1991.

[46] S. S. Kuo, M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. Fast methods for simulation of biomolecule electrostatics. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, ICCAD '02, pages 466–473, New York, NY, USA, 2002. ACM.

[47] I. Lashuk, A. Chandramowlishwaran, H. Langston, T. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros. A massively parallel adaptive fast-multipole method on heterogeneous architectures. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09*, pages 1–12, Portland, Oregon, November 2009.

[48] J. Liang and S. Subramaniam. Computation of molecular electrostatics with boundary element methods. *Biophys. J.*, 73(4):1830–1841, 1997.

[49] H.-Y. Liu, S. Z. Grinter, and X. Zou. Multiscale generalized Born modeling of ligand binding energies for virtual database screening. *J. Phys. Chem. B*, 113(35):11793–11799, 2009. PMID: 19678651.

[50] B. Lu, X. Cheng, J. Huang, and J. A. McCammon. Order N algorithm for computation of electrostatic interactions in biomolecular systems. *P. Natl. Acad. Sci. USA*, 103(51):19314–19319, 2006.

[51] B. Lu and J. A. McCammon. Improved boundary element methods for Poisson–Boltzmann electrostatic potential and force calculations. *J. Chem. Theory Comput.*, 3:1134–1142, 2007.

[52] B. Lu, D. Zhang, and J. A. McCammon. Computation of electrostatic forces between solvated molecules determined by the Poisson–Boltzmann equation using a boundary element method. *J. Chem. Phys.*, 122(21):214102, 2005.

[53] I. Massova and P. A. Kollman. Combined molecular mechanical and continuum solvent approach MM-PBSA/GBSA to predict ligand binding. *Perspect. Drug Discov.*, 18:113–135, 2000.

[54] S. R. McGuffee and A. H. Elcock. Atomistically detailed simulations of concentrated protein solutions: the effects of salt, pH, point mutations, and protein concentration in simulations of 1000-molecule systems. *J. Am. Chem. Soc.*, 128:12098–12110, 2006.

[55] S. Miertus, E. Scrocco, and J. Tomasi. Electrostatic interactions of a solute with a continuum – a direct utilization of *ab initio* molecular potentials for the prevision of solvent effects. *Chem. Phys.*, 55(1):117–129, 1981.

[56] A. P. Minton. Implications of macromolecular crowding for protein assembly. *Curr. Opin. Struc. Biol.*, 10:34–39, 2000.

[57] K. Nabors and J. White. FastCap: A multipole accelerated 3-D capacitance extraction program. *IEEE Trans. Computer-Aided Design*, 10(11):1447–1459, 1991.

[58] B. L. Neal and A. M. Lenhoff. Excluded volume contribution to the osmotic second virial coefficient for proteins. *AIChE J.*, 41:1010–1014, 1995.

[59] J. N. Newman. Distribution of sources and normal dipoles over a quadrilateral panel. *J. Eng. Math.*, 20(2):113–126, 1986.

[60] NVIDIA Corp. CUDA programming guide version 2.2.1, May 2009.

[61] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 26:1781–1802, 2005.

[62] J. R. Phillips and J. K. White. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Trans. Comput. Aid. D.*, 16(10):1059–1072, 1997.

[63] D. Qiu, P. S. Shenkin, F. P. Hollinger, and W. C. Still. The GB/SA continuum model for solvation. A fast analytical method for the calculation of approximate Born radii. *J. Phys. Chem. A*, 101(16):3005–3014, 1997.

[64] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[65] F. M. Richards. Areas, volumes, packing, and protein structure. *Ann. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[66] V. Rokhlin. Rapid solution of integral equation of classical potential theory. *J. Comput. Phys.*, 60:187–207, 1983.

[67] A. N. Romanov, S. N. Jabin, Y. B. Martynov, A. V. Sulimov, F. V. Grigoriev, and V. B. Sulimov. Surface generalized Born method: A simple, fast, and precise implicit solvent model beyond the Coulomb approximation. *J. Phys. Chem. A*, 108(43):9323–9327, 2004.

[68] S. Rush, A. H. Turner, and A. H. Cherin. Computer solution for time-invariant electric fields. *J. Appl. Phys.*, 37(6):2211–2217, 1966.

[69] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[70] M. F. Sanner. Molecular surface computation home page. http://mgltools.scripps.edu/packages/MSMS (last checked Feb. 4, 2010), 1996.

[71] C. N. Schutz and A. Warshel. What are the dielectric constants of proteins and how to validate electrostatic models? *Proteins*, 44:400–417, 2001.

[72] K. A. Sharp and B. Honig. Calculating total electrostatic energies with the nonlinear Poisson–Boltzmann equation. *J. Phys. Chem.*, 94(19):7684–7692, 1990.

[73] K. A. Sharp and B. Honig. Electrostatic interactions in macromolecules: Theory and applications. *Ann. Rev. Biophys. Bio.*, 19:301–332, June 1990.

[74] P. B. Shaw. Theory of the Poisson Green's-function for discontinuous dielectric media with an application to protein biophysics. *Phys. Rev. A*, 32(4):2476–2487, 1985.

[75] D. Sitkoff, K. A. Sharp, and B. Honig. Accurate calculation of hydration free energies using macroscopic solvent models. *J. Phys. Chem.*, 98(7):1978–1988, 1994.

[76] R. D. Skeel, I. Tezcan, and D. J. Hardy. Multiple grid methods for classical molecular dynamics. *J. Comput. Chem.*, 23(6):673–684, 2002.

[77] S. Spector, M. H. Wang, S. A. Carp, J. Robblee, Z. S. Hendsch, R. Fairman, B. Tidor, and D. P. Raleigh. Rational modification of protein stability by the mutation of charged surface residues. *Biochemistry*, 39:872–879, 2000.

[78] W.C. Still, A. Tempczyk, R. C. Hawley, and T. F. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, 112(16):6127–6129, 1990.

[79] M. Stork and P. Tavan. Electrostatics of proteins in dielectric solvent continua. I. Newton's third law marries qE forces. *J. Chem. Phys.*, 126(16):165105, 2007.

[80] T. Takahashi and T. Hamada. GPU-accelerated boundary element method for Helmholtz' equation in three dimensions. *Int. J. Num. Meth. Eng.*, 80(10):1295–1321, 2009.

[81] C. L. Vizcarra and S. L. Mayo. Electrostatics in computational protein design. *Curr. Opin. Chem. Biol.*, 9(6):622–626, 2005.

[82] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, pages 12–21, New York, 1993. ACM.

[83] A. Warshel, P. K. Sharma, M. Kato, and W. W. Parson. Modeling electrostatic effects in proteins. *Biochim. Biophys. Acta*, 1764:1647–1676, 2006.

[84] J. Warwicker and H. C. Watson. Calculation of the electric potential in the active site cleft due to alpha-helix dipoles. *J. Mol. Biol.*, 157:671–679, 1982.

[85] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.*, 196(2):591–626, 2004.

[86] R. Yokota, T. K. Sheel, and S. Obi. Calculation of isotropic

turbulence using a pure Lagrangian vortex method. *J. Comput. Phys.*, 226(2):1589–1606, 2007.

[87] B. J. Yoon and A. M. Lenhoff. A boundary element method for molecular electrostatics with electrolyte effects. *J. Comput. Chem.*, 11(9):1080–1086, 1990.

[88] T. J. You and D. Bashford. Conformation and hydrogen ion titration of proteins: a continuum electrostatic model with conformational flexibility. *Biophys. J.*, 69(5):1721–1733, 1995.

[89] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Comput. Aided Geom. D.*, 23(6):510–530, 2006.

[90] H. X. Zhou. Boundary-element solution of macromolecular electrostatics—interaction energy between 2 proteins. *Biophys. J.*, 65:955–963, 1993.

[91] S. B. Zimmerman and A. P. Minton. Macromolecular crowding: Biochemical, biophysical, and physiological consequences. *Ann. Rev. Biophys. Biomol. Struct.*, 22(1):27–65, 1993.