GPU/CPU Algorithm for Generalized Born/Solvent-Accessible Surface Area Implicit Solvent Calculations

David E. Tanner,[†,‡] James C. Phillips,[‡] and Klaus Schulten*,[†,‡,§]

[†]Center for Biophysics and Computational Biology, University of Illinois at Urbana−Champaign, Urbana, Illinois 61801, United States

[‡]Beckman Institute, University of Illinois at Urbana−Champaign, Urbana, Illinois 61801, United States

[§]Department of Physics, University of Illinois at Urbana−Champaign, Urbana, Illinois 61801, United States

ABSTRACT: Molecular dynamics methodologies comprise a vital research tool for structural biology. Molecular dynamics has benefited from technological advances in computing, such as multicore CPUs and graphics processing units (GPUs), but harnessing the full power of hybrid GPU/CPU computers remains difficult. The generalized Born/solvent-accessible surface area implicit solvent model (GB/SA) stands to benefit from hybrid GPU/CPU computers, employing the GPU for the GB calculation and the CPU for the SA calculation. Here, we explore the computational challenges facing GB/SA calculations on hybrid GPU/CPU computers and demonstrate how NAMD, a parallel molecular dynamics program, is able to efficiently utilize GPUs and CPUs simultaneously for fast GB/SA simulations. The hybrid computation principles demonstrated here are generally applicable to parallel applications employing hybrid GPU/CPU calculations.

■ INTRODUCTION

Laboratory and clinical studies of living cells are increasingly complemented by computational atomic level modeling of cellular processes.[1−8] In fact, modeling has developed today into a computational microscope[9] used to explore nanoscale structures and processes in structural cell biology,[10−12] cellular mechanics,[9,13,14] and nanosensor development.[15,16] Small size and short time scales of many subcellular processes challenge experimental methodologies, making the molecular dynamics computational microscope an ideal supporting tool for biological research.

Molecular dynamics (MD) has already enabled, in particular, modeling of health-relevant biomolecular systems and processes, e.g., viral infection,[17,18] interactions between tissues and therapeutics,[1,2] blood coagulation,[13,19−25] and amyloid fibril formation.[26−31] Often, however, simulation time scales are too short to be of value; only by continually adopting the latest computing technologies can simulation speeds increase and extend the reach of MD to the millisecond scale[32] that is necessary to describe many cellular processes.

MD simulations calculate interatomic forces of biopolymer systems and solve the classical equations of motion for each time step,[33] to explore the dynamic behavior of biological systems. The largest computational expense in MD simulations stems from calculating so-called "non-bonded" interactions, forces between atoms which are not covalently bonded, comprising Coulomb and van der Waals forces.

The calculation of nonbonded forces is highly amenable to parallel computing[34,33] because the modeled interaction between two atoms is independent of all other atoms; thus, atom-pair interactions can be calculated independently from one another, then accumulated to determine net atomic forces. Such independent calculations offer a high degree of concurrency, making them ideal for parallel computing.[35] Indeed, NAMD successfully conducted recently a 100 000 000-atom MD simulation on 200 000 CPU cores.[36] Along with multicore CPUs, MD programs have also harnessed new computing technologies such as graphics processing units (GPUs).[37,38]

Graphics Processing Units. Though originally developed only for interactive graphics rendering, GPUs are well suited for accelerating general scientific calculations. A single GPU may today contain 16 multiprocessors, each with 32 cores, yielding 512 cores per GPU. Maximizing the use of such highly parallel processors requires meeting strict criteria such as coalescing access to slow memory and issuing 10 000's of independent calculations. Because many molecular biological calculations can meet the strict criteria, GPUs have been used to accelerate biological computing applications such as molecular modeling,[39] electrostatic calculations,[37,40] molecular orbital display,[41] and simulations of protein diffusion in whole cells.[42]

The application of GPUs to molecular dynamics[43] has already demonstrated significant performance benefits[44] for both explicit solvent[38,45] and implicit solvent[46,47] models. Each implementation varies in the use of the GPU, ranging from employing the GPUs for calculating forces only[38] to using them to integrate the equations of motion[48,49] as well. GPU implementations also vary in regard to how restrictive they are of the MD methodologies they admit; for example, some implementations allow a nonbonded interaction cutoff length[37] which is common for large systems, while others do not,[46] thereby being applicable only to small systems.

The great success already achieved in accelerating full molecular dynamics calculations with GPUs inspires additional work in advancing MD calculations which involve both GPU-accelerated calculations and CPU calculations (those not yet adapted to the GPU) as they arise in, for example, steering[50] or

Received: April 16, 2012
Published: June 15, 2012

grid potentials.[51] Prior efforts at GPU acceleration of MD simulations had adapted almost all of the computation to the GPU, the CPU's role remaining secondary.[48,49]

Hybrid GPU/CPU Computing. Each GPU requires a so-called host CPU to send data to and receive data from as well as to instruct the GPU as to which calculations to perform. Therefore, a necessary and difficult component of hybrid GPU/CPU calculations is allowing host CPUs to switch between GPU-hosting responsibilities and performing their own scientific calculations. Because many existing MD methods have not yet been adapted to GPUs, or are not amenable to GPU calculation, it is vital to explore how to efficiently perform hybrid calculations which require full use of both GPUs and CPUs and which, therefore, require coordination between the two.

An ideal application for hybrid computing is the generalized Born (GB) implicit solvent with solvent-accessible surface area (SASA or SA) calculation, commonly abbreviated GB/SA. A GB/SA implicit solvent simulation employs the GB calculation to account for the polar, i.e., hydrophilic, effects of water while the SA calculation models the nonpolar, i.e., hydrophobic, effects.[52] It is known that the hydrophobic free energy of solvation is approximately equal to the product of the molecular surface area and a surface tension parameter.[53,54] The GB/SA calculation consists of three classes of nonbonded interatomic forces: the classical Coulomb and van der Waals forces, the generalized Born[53,55,56] (GB) force, and the solvent-accessible surface area (SA) force, calculated via the linear combination of pairwise overlaps[54] (LCPO) algorithm, as in the Amber MD program.[57] While the Coulomb, van der Waals, and GB force calculations are being calculated on the GPU, the LCPO algorithm's SA calculation is best performed on the CPU, thus making the GB/SA calculation an ideal candidate for hybrid GPU/CPU calculation.

The present work explores a variety of performance and functionality issues relevant to GPU accelerated calculations of the GB/SA implicit solvent model in the context of the NAMD parallel MD program[33] First, we analyze the necessary nonbonded force calculations and describe why each is best suited for GPUs or CPUs. Second, we outline NAMD's algorithmic strategy for GB/SA calculations on hybrid GPU/CPU computers. Finally, we explore the effect of the surface tension parameter as well as demonstrate the performance of the hybrid strategy through ~250 benchmark simulations.

■ METHODS

The generalized Born/solvent-accessible surface area (GB/SA) implicit solvent model constitutes a fast representation of a solvent's polar and nonpolar effects on biomolecules.[53] Fast simulation speed can be attained by calculating the Coulomb, van der Waals, and generalized Born (GB) forces on the GPU while simultaneously calculating the hydrophobic surface area (SA) force on the CPU. First, we present the equations arising in the Coulomb, van der Waals, GB, and SA force calculations which motivate the hybrid GPU/CPU algorithm employed. Then, NAMD's implementation of the various calculations will be described.

Coulomb and van der Waals Forces. The nonbonded forces arising in explicit and implicit solvent simulations are the Coulomb and van der Waals (calculated using the Lennard-Jones description) forces. The total system energy contributed by Coulomb and van der Waals interactions, E_T^{CW}, is

$$E_T^{CW} = (1/2)\sum_i \sum_{j\in N(i)} \{\underbrace{4\varepsilon_{ij}[(\sigma_{ij}/r_{ij})^{12} - (\sigma_{ij}/r_{ij})^6]}_{\text{(van der Waals)}} + \underbrace{(k_e/\varepsilon_p)(q_iq_j/r_{ij})}_{\text{(Coulomb)}}\} \tag{1}$$

where ε_{ij} is the well depth and $2^{1/6}\sigma_{ij}$ equilibrium interaction length parameters of the Lennard-Jones potential, r_{ij} is the distance between atoms i and j, k_e = 332 kcal Å/e^2 is the Coulomb constant, ε_p = 1 is the dielectric constant of the protein interior, and q_i is the atomic charge. $N(i)$ is the set of all neighbors, j, that are within the interaction cutoff, r_c, from atom i.

The net force, $\vec{F}_i$, on an atom is calculated as

$$\vec{F}_i = -\sum_{j\in N(i)} (dE_T/dr_{ij})\hat{r}_{ij} \tag{2}$$

by applying eq 2 to eq 1, the net Coulomb and van der Waals forces on an atom, $\vec{F}_i^{CW}$, is

$$\vec{F}_i^{CW} = \sum_{j\in N(i)} \{\underbrace{24\varepsilon_{ij}[2(\sigma_{ij}^{12}/r_{ij}^{13}) - (\sigma_{ij}^6/r_{ij}^7)]}_{\text{(van der Waals)}} + \underbrace{(k_e/\varepsilon_p)(q_iq_j/r_{ij}^2)}_{\text{(Coulomb)}}\}\hat{r}_{ij} \tag{3}$$

The summation in eq 3 requires an MD program to iterate over all pairs of interacting atoms, i and j, where $r_{ij} < r_c$; the successful application of GPUs to computing atom-pair interactions, such as Coulomb and van der Waals forces, has previously been demonstrated in NAMD.[37,38]

Generalized Born Implicit Solvent Forces. The generalized Born implicit solvent model,[53,56] already implemented in NAMD for the CPU,[58] describes water as a bulk solvent acting as a dielectric;[53,56] the dielectric solvent screens,[59,60] i.e., reduces, electrostatic interactions between charged atoms. The total GB energy, i.e., hydrophilic energy of solvation, of the atomic system is given by the sum of pair and self energies according to

$$E_T^{GB} = (1/2)\sum_i \sum_{j\in N(i)} \underbrace{E_{ij}^{GB}}_{\text{(pair)}} + (1/2)\sum_i \underbrace{E_{ii}^{GB}}_{\text{(self)}} \tag{4}$$

where the pair- and self-GB energies are calculated according to

$$E_{ij}^{GB} = -k_e D_{ij} q_i q_j / f_{ij}^{GB} \tag{5}$$

Here, D_{ij} is an effective dielectric constant, which depends on the implicit ion concentration,[58,61] between atoms i and j, and $f^{GB}{}_{ij}$ is[53]

$$f_{ij}^{GB} = \sqrt{r_{ij}^2 + \alpha_i\alpha_j \exp(-r_{ij}^2/4\alpha_i\alpha_j)} \tag{6}$$

The quantities α_i arising in this expression, the so-called atomic Born radii, describe an atom's exposure to solvent and, thus, characterize the degree to which an atom's electrostatic interaction is screened; α_i is calculated, according to Onufriev, Bashford and Case's (OBC) description,[56] as

$$\alpha_i = [(\rho_i - 0.09\text{Å})^{-1} - (1/\rho_i)\tanh(\delta\psi_i - \beta\psi_i^2 + \gamma\psi_i^3)]^{-1} \tag{7}$$

where ρ_i is the atomic radius as parametrized by the OBC model; $\delta = 1$, $\beta = 0.8$, and $\gamma = 4.85$ are additional dimensionless parameters of the OBC model[56] which enable calculation of the correct Born radii, i.e., those derived from Poisson−Boltzmann calculations, from the sum, ψ_i, of pairwise atomic descreening, H_{ij},

$$\psi_i = (\rho_i - 0.09) \sum_{j \in N(i)} H_{ij} \tag{8}$$

The pairwise descreening, H_{ij}, between neighboring atoms i and j is given by a distance-dependent piecewise function, defined in four mutually exclusive interaction domains

$$H_{ij} = \begin{cases} 0 & (r_{ij} > r_c + \rho_j) \\ f_1(r_{ij}, \rho_j, r_c) & (r_{ij} > r_c - \rho_j) \\ f_2(r_{ij}, \rho_j) & (r_{ij} > 4\rho_j) \\ f_3(r_{ij}, \rho_j) & (r_{ij} > \rho_i - 0.09 + \rho_j) \\ f_4(r_{ij}, \rho_i, \rho_j) & \text{otherwise} \end{cases} \tag{9}$$

where f_{1-4} are also taken from the OBC model.[56]

Because the GB energy defined in eq 4 depends on the interatomic distances directly, through f_{ij}^{GB}, and also indirectly, through α_i, calculating the GB force on an atom, $\vec{F}_i^{GB}$, requires multiple partial derivatives,[58] namely,

$$\vec{F}_i^{GB} = \sum_{j \in N(i)} [(\partial E_{ij}^{GB}/\partial f_{ij}^{GB})(df_{ij}^{GB}/dr_{ij}) + (\partial E_T^{GB}/\partial \alpha_i)(d\alpha_i/dr_{ij}) + (\partial E_T^{GB}/\partial \alpha_j)(d\alpha_j/dr_{ij})]\hat{r}_{ij} \tag{10}$$

Of the various required derivatives, $\partial E_{ij}^{GB}/\partial f_{ij}^{GB}$, $\partial E_T^{GB}/\partial \alpha_k$, and $d\alpha_k/dr_{ij}$, the most expensive to calculate is $\partial E_T^{GB}/\partial \alpha_k$, as it requires an additional summation over atom pairs

$$\partial E_T^{GB}/\partial \alpha_k = (1/2) \sum_i \sum_{j \in N(i)} [\partial E_{ik}^{GB}/\partial \alpha_k + \partial E_{kj}^{GB}/\partial \alpha_k] + \sum_i \partial E_{ii}^{GB}/\partial \alpha_k \tag{11}$$

The three summations in eqs 8, 10, and 11, known as the three phases of the GB force calculation, require three successive iterations over all pairs of atoms each time step. The three phases compute, in order, the atomic Born radii, α_i; the partial derivatives $\partial E_T^{GB}/\partial \alpha_k$; and, finally, the GB force, $\vec{F}_i^{GB}$.

As in the case of Coulomb and van der Waals forces, iterating over atom pairs for the three GB phases is amenable to GPU acceleration. Because calculating the generalized Born force as outlined in eqs 4−11 is several times more computationally expensive than calculating only the Coulomb and van der Waals forces, and because GPUs are well suited for such arithmetically expensive calculations, the GB calculation stands to benefit strongly from GPU acceleration as previously demonstrated by other MD programs.[46,47] Even for the case that the computational expense of an implicit solvent simulation is greater than that of an all-atom explicit solvent simulation, because conformational relaxation processes are usually faster in implicit solvent,[58] such a model can still yield an overall benefit to conformational sampling.

Solvent-Accessible Surface Area Forces. The generalized Born model only describes the polar, i.e., hydrophilic, energy of solvation. It is desirable to account also for the nonpolar, i.e., hydrophobic, energy of solvation through a solvent-accessible surface area (SA) calculation, as it is known that the hydrophobic solvation energy is approximately proportional to SA.[53,62] We note here that the hydrophobic energy contributes to every surface element of a protein, even those involving charged side groups; this seemingly counterintuitive issue is discussed below. The linear combination of pairwise overlaps (LCPO) is an approximate method for calculating the SA[54] and, in particular, the spatial derivatives necessary for hydrophobic force calculation.

The LCPO method,[54] which considers only non-hydrogen atoms, is founded on calculating the surface area overlap between two spheres representing atoms; for two spheres, centered on atoms i and j, with radii R_i and R_j, separated by a distance r_{ij}, close enough that their surfaces overlap, i.e., it must hold $r_{ij} < R_i + R_j$, the surface area of atom i overlapped by atom j, A_{ij}, is[54]

$$A_{ij} = 2\pi R_i[(R_i - (r_{ij}/2) - (R_i^2 - R_j^2)/(2r_{ij})] \tag{12}$$

where the radius, R_i, is the atomic van der Waals radius plus the 1.4 Å solvent probe radius.[54] According to the LCPO method, the surface area of atom i, SA_i, can be evaluated approximately by multiple overlap summations,

$$SA_i = P_{1,i} 4\pi R_i^2 + P_{2,i} \sum_{j \in N(i)} A_{ij} + P_{3,i} \sum_{j \in N(i)} A_{ij} \sum_{k \in N(i) \cap N(j)} A_{jk} + P_{4,i} \sum_{j \in N(i)} [A_{ij} \sum_{j \in N(i)} \sum_{k \in N(i) \cap N(j)} A_{jk}] \tag{13}$$

where $k \in N(i) \cap N(j)$ represents all atoms k which overlap atoms i and j, thus requiring the LCPO algorithm to iterate over atom triplets, sets of three atoms whose surfaces overlap (i.e., it must hold $r_{ij} < R_i + R_j$, $r_{ik} < R_i + R_k$, $r_{jk} < R_j + R_k$). The total molecular surface area is the sum of atomic surface areas, $\sum_i SA_i$. The four atomic parameters $P_{1-4,i}$ were fitted by atom type such that the LCPO algorithm most closely approximates the correct surface area as calculated by numerical methods.[54] The hydrophobic, surface area force on atom l, $\vec{F}_l^{SA}$, can then be calculated employing the LCPO algorithm

$$\vec{F}_l^{SA} = -T_S \sum_{i \in N(l)} (dSA_i/dr_l) \tag{14}$$

where T_S, with units of kcal/mol/Å^2, is the surface tension parameter. The relatively inexpensive derivatives required for the SA force calculation within the LCPO algorithm were previously described.[54]

For the above three force calculations, approximately 13% of the computational cost is due to Coulomb and van der Waals forces, 70% due to GB forces, and the remaining 17% due to SA forces. Having transferred most of the force calculation (Coulomb, van der Waals, GB) to the GPU, the CPU can perform the relatively inexpensive (17% of total) SA calculation in the same time as the GPU's expensive (83% of total) force calculations. The SA algorithm requires iterating over all atom triplets, compared to only pairs of atoms as in the case of GB; the summation is feasible due to the low computational cost involved and because the summation over triplets can be performed well by CPUs.

Incorporating the GB/SA Calculation into NAMD. To achieve a high performance implementation of the GB/SA model, formulated by eqs 1−14, we thought to optimize use of hybrid GPU/CPU computers as outlined. Aside from fast performance, an ideal implementation should interfere as little as possible with NAMD's already highly scalable internal structure.

To achieve advanced parallel scaling, NAMD decomposes a simulated system (Figure 1A) into a 3D grid of atom groups (Figure 1B). In the case of Coulomb, van der Waals, and GB forces, atom groups are defined such that atoms in any atom group interact only with atoms of the same atom group and with atoms of adjacent atom groups. Calculating the atom-pair interactions between two neighboring atom groups (Figure 1D, red and green) constitutes an independent force work unit. To illustrate our GB/SA implementation, we apply NAMD to the 13 340-atom glycogen phosphorylase protein, Protein Data Bank (PDB) ID 1GPB, with missing atoms placed by VMD's[63] psfgen tool. The atoms of this protein system can be divided into roughly 100 atom groups, shown in Figure 1B, and 3000 force work units which can be assigned across hundreds of CPU cores or GPUs.[37,38]

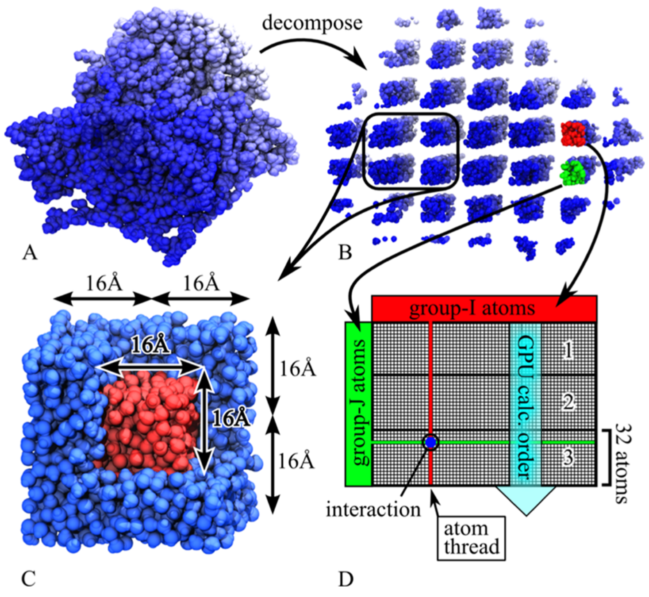

Figure 1. Hybrid GB/SA decomposition. (A) Simulated protein glycogen phosphorylase. (B) Protein decomposed into a 3D grid of atom groups. (C) LCPO force work unit involving a 2 × 2 × 2 set of eight adjacent atom groups. The force work unit calculates forces for the inner 1/8th core of atoms (red) due to overlap with neighbors (red and blue). (D) Thread block design for GB force calculation on GPU multiprocessor. Each thread (vertical line) calculates the force on one group-I atom (red) due to interactions (blue) with group-J atoms (green); group-J atoms are loaded 32 at a time, in order 1−3, for coalesced and, therefore, accelerated memory access.

GB Force Calculation on GPUs. NAMD has already achieved a 6-fold speedup on its 92 000-atom ApoA1 benchmark system by calculating Coulomb and van der Waals forces on the GPU using the CUDA language.[37,38] Because of the success of the Coulomb and van der Waals calculations on the GPU and the close algorithmic similarities, NAMD's GB calculation on the GPU, outlined below, employs a similar GPU implementation as NAMD's previously reported Coulomb and van der Waals force calculation on the GPU.[37,38]

For parallel calculations, each processor core requires a list of instructions to drive computation; for both GPU and CPU processors, a "thread" is the list of instructions detailing what operations a processor core must perform. Multicore CPUs and GPUs both require multiple threads, i.e., multiple lists of instructions, to drive the many processor cores. In CUDA, a thread block is the group of threads which drives one of the GPU's many 32-core multiprocessors.

In NAMD, a force work unit consists of calculating the atomic interactions between two atom groups, such as group I (red) and group J (green) of Figure 1B and D. The calculation of the force work unit is assigned to a thread block, i.e., is calculated by a single 32-core multiprocessor, as depicted in Figure 1D. Each thread (vertical line) in the thread block calculates the force on a single atom of group I (red) due to its interactions (blue circle) with atoms of group J (green). To accelerate memory access, the GPU's 32-core multiprocessors access group J atoms 32 at a time, in order 1−3 shown in Figure 1D, and intermediately store the data in "shared" memory, a fast memory shared only by the 32 cores of a multiprocessor.

The above pattern of NAMD's GPU calculation of Coulomb, van der Waals, and GB forces exploits several features of GPU hardware to achieve fast performance. First, because adjacent threads operate on atoms adjacent in memory, reading atomic coordinates from and writing atomic forces to slow memory is coalesced, giving significant memory speed-up over non-coalesced memory access. Second, because GPUs operate fastest when threads on a multiprocessor perform the same calculation, NAMD presorts atomic coordinates before transferring them to the GPU to reduce the likelihood that threads on a multiprocessor must evaluate differing pairwise functions, f_{1-4} of eq 9, as doing so dramatically reduces performance. Much work has previously been done exploring implementation patterns for optimal GPU performance of molecular modeling applications.[37−39,64,65]

SA Force Calculation on CPUs. Because eq 14 requires iteration over all atom triplets, the LCPO calculation cannot be carried out using the structure of atom-group pairs which NAMD employs for Coulomb, van der Waals, and GB force calculation, as it would be possible for a triplet of three overlapping atoms to belong to three neighboring atom groups, in which case no atom-group pair would evaluate the triplet. Therefore, an LCPO force work unit instead consists of a 2 × 2 × 2 set of eight adjacent atom groups, as illustrated in Figure 1C. Each LCPO force work unit calculates the surface area and hydrophobic force on the inner 1/8th fraction of atoms (Figure 1C red) due to surface area overlaps with neighboring atoms (red and blue). LCPO force work units can also be partitioned such that the force work unit in Figure 1C is duplicated with each copy calculating forces for only a fraction of the inner core atoms (red). The ability to partition the SA calculation into many small force work units will be shown to be critical to the efficiency of the hybrid GPU/CPU algorithm.

Balancing the GPU and CPU Calculations. With fast algorithms in place to perform the GB force calculation on the GPU and the SA force calculation on the CPU, the remaining obstacle of combining the two into a hybrid GB/SA calculation requires coordination of the GPU and CPU computation. First, the right balance of GPUs and CPU cores will allow the Coulomb, van der Waals, and GB calculation on the GPU and the SA calculation on the CPU to be executed in the same amount of time. Second, partitioning the SA calculation into small force work units will allow the host CPU to switch between SA calculations and GPU interaction with high frequency, thereby minimizing the delay of either calculation.

Because GPUs and CPUs possess different computational power and are responsible for calculating workloads of different size, a computationally efficient simulation requires that one must first determine, through benchmarking the particular system to be simulated, an optimal ratio of CPU cores per GPU that balances the durations of GB and SA calculations. While the thorough benchmarking of our implementation provided below (see Table 2) will aid a user in judging performance a priori, the user may want to verify in case of any particular calculation if a chosen ratio of GPUs to CPU cores yields efficient performance, for example by varying the ratio and judging the resulting overall performance. In order to achieve optimal performance, it must also be verified that CPU and GPU calculations overlap; the corresponding analysis will be carried out below.

Simulations Carried Out. To analyze the performance of NAMD's hybrid GB/SA algorithm, four test systems, differing in size, were simulated on 0−4 GPUs and 1−32 CPU cores. Figure 2 depicts the four tested systems, and Table 1 lists their SA values. Atoms missing in the public PDB structure files were placed by VMD's[63] psfgen tool.

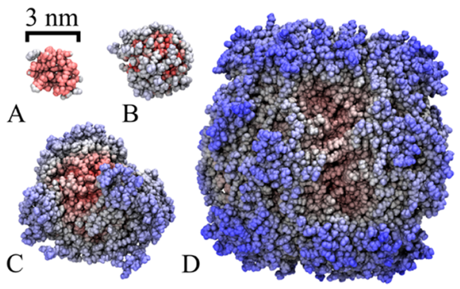

Figure 2. Benchmarked protein systems. (A) Villin headpiece (PDB ID 1YRI) with 582 atoms; (B) flavodoxin (PDB ID 2W5U) with 2412 atoms; (C) glycogen phosphorylase (PDB ID 1GPB) with 13 340 atoms; (D) RuBisCO (PDB ID 1IR2) with 74 926 atoms.

Table 1. Benchmarked Protein Systems[a]

name	PDB ID	# atoms	NAMD SA	amber SA
villin headpiece	1YRI	582	2706.72	2706.72
flavodoxin	2W5U	2412	9988.72	9988.72
glycogen phosphorylase	1GPB	13 340	46 106.66	46 106.68
RuBisCO	1IR2	74 926	176 335.86	176 335.90

[a]Listed are associated surface areas, in units of Å^2, as calculated by NAMD's and Amber's implementations of the LCPO[54] algorithm.

For benchmarking, three types of simulations were performed, each with increasing computational cost and solvent accuracy. In vacuo simulations evaluate only the Coulomb and van der Waals nonbonded forces as defined through eqs 1−3, GB simulations additionally evaluate the generalized Born forces, determined through eqs 4−11, and GB/SA simulations further include the SA calculation following the LCPO algorithm as stated through eqs 12 and 14.

The following simulation parameters were employed for all simulations. A value of 16 Å was employed for the Coulomb and van der Waals interaction cutoff as well as for the GB phase 2 calculation, c.f. eq 11, while the GB phase 1 and 3 calculations, as defined through eqs 8 and 10, were cut off at 14 Å. For GB and GB/SA simulations, an implicit ion concentration of 0.3 M was assumed.[61] The GB/SA simulations employed a surface tension of 0.005 kcal/mol/Å^2,[62] unless otherwise specified. A time step of 2 fs was employed, requiring the SHAKE[66] algorithm to restrain covalent bonds to hydrogen atoms, with all forces being evaluated every time step.

Benchmark simulations were run for 620 time steps on a 2.2 GHz 48-core AMD Opteron 6174 computer accelerated with an NVIDIA S2070 GPU system. The first 20 steps of simulation performed conjugate gradient minimization; the next 500 steps consisted of NAMD load balancing to optimize parallel performance. The simulation speed, in units of seconds/time step, was averaged over the final 100 steps. Results of the benchmark simulations are presented in Figures 3 and 4.

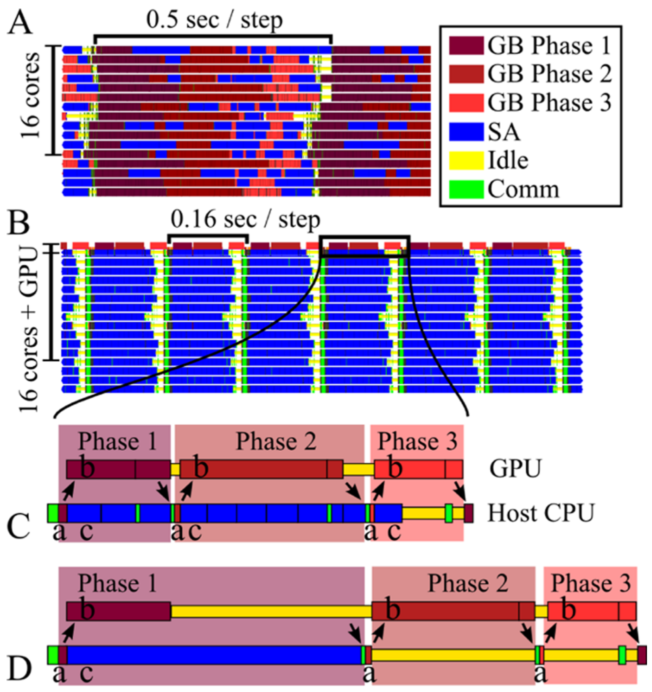

Figure 3. NAMD's GB/SA calculation for the 1GPB system. (A) Performance on 16 CPUs only. (B) Performance on 16 CPUs with 1 GPU added. Colors represent calculations being performed on the processors: three phases of GB calculation (see text), SA calculation, idle time, communication. (C) Detailed view of host CPU switching between communicating with the GPU and performing the SA calculation. Arrows represent the transfer of data between host CPU and GPU. For GB phase 1, 2, and 3 calculations: (a) host CPU initializes the GB calculation on the GPU, (b) GPU calculates a GB phase while (c) host CPU performs SA calculations. (D) Detailed view of how switching without SA partitioning slows performance.

To verify the computational accuracy of NAMD's new GPU-accelerated GB calculation and multicore LCPO calculation, energy and surface area calculations were compared to those of prior reference implementations.[57,58] The relative error, $(E_{ref} - E_{new})/E_{ref}$, of NAMD's GB energy calculation, determined through eq 4, on the GPU, with NAMD's CPU implementation[58] as a reference, falls below 5×10^{-6} for all four benchmark proteins. The relative error, $(SA_{ref} - SA_{new})/SA_{ref}$, of NAMD's surface area calculation, determined through eq 14, with the Amber[57] implementation as a reference, falls below 4×10^{-7} for all four benchmark systems; total molecular surface areas of the four systems, as calculated by the Amber and NAMD implementations of LCPO, are listed in Table 1.

Hydrophobic solvation energy is a significant contributor to solvent behavior,[67,68] necessitating its inclusion along with GB calculations. The effect of the hydrophobic energy contribu-

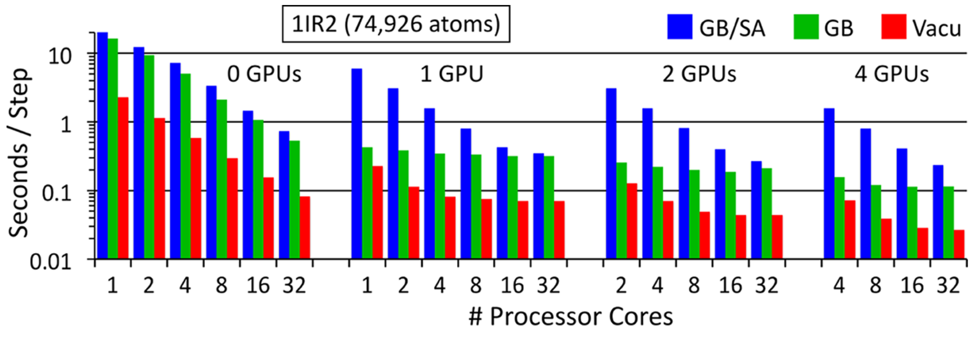

Figure 4. Simulation speed, in seconds/time step, for the 1IR2 benchmark system on 0−4 GPUs and 1−32 CPU cores; speeds for in vacuo (red), GB (green), and GB/SA (blue) calculations are shown. Both the seconds/step and processor cores axes are logarithmic.

tions, accounted for in the SA calculation, was explored by simulating the 2W5U benchmark system using the GB/SA implicit solvent, employing eight different surface tension parameter values. Additionally, the 2W5U system was simulated in TIP3P[69] explicit solvent to determine which surface tension parameter values most closely reproduced protein behavior in explicit solvent. With the experimentally measured surface tension of hydrocarbons in aqueous solvent being 0.005 kcal/mol/Å^2,[62] surface tension parameter values in the range 0.001−0.128 kcal/mol/Å^2 were tested. Using otherwise the same simulation parameters previously outlined, the 2W5U system was equilibrated for 1.0 ns, at which time the total surface area, a molecular property closely affected by surface tension, had reached an equilibrium value. The surface areas evaluated during the simulations are plotted in Figure 5; for the TIP3P

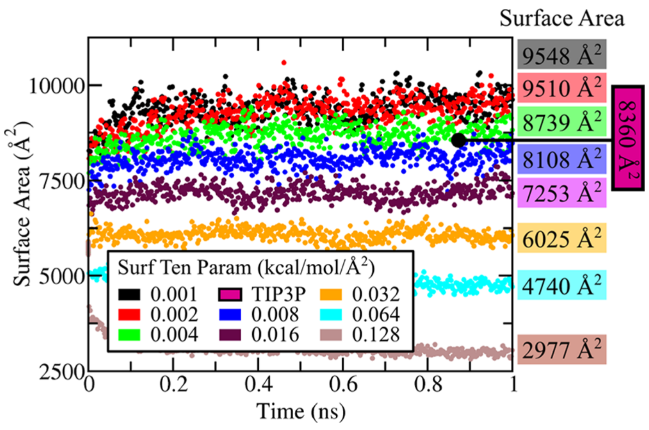

Figure 5. Surface area during eight GB/SA simulations of the 2W5U system employing different surface tension parameter values. The final surface area for each GB/SA simulation is listed at right; shown is also the final surface area resulting from the simulation with explicit (TIP3P) solvent.

simulation, only the final protein surface area, as calculated by VMD's[63] high precision solvent-accessible surface area tool, with a solvent probe radius of 1.4 Å, is shown.

■ RESULTS

The performed simulations examine the hybrid GPU/CPU algorithm by verifying simultaneous overlap of GB and SA force calculations; analyzing the ideal ratio of GPUs to CPU cores, which maximizes computing efficiency; benchmarking performance; and evaluating appropriate values for the surface tension parameter.

Hybrid GB/SA Performance Analysis. Central to fast hybrid GPU/CPU algorithms is the simultaneous overlap of calculations on both processor types, brought about by the host CPUs coordinating the GB calculation on the GPU with the SA calculations on the CPU. To demonstrate this coordination, Figure 3 illustrates, using the Projections[70] tool, the execution process of a GB/SA simulation of the 1GPB system on multiple GPUs and CPU cores.

Figure 3A illustrates the GB/SA calculation executed on 16 CPU cores without GPUs. The three phases of the GB calculation, described by eqs 8, 11, and 10, are carried out with SA calculations interspersed to utilize the CPU when it would otherwise be idle while waiting for other cores to complete the phase; the lack of idle time (shown in yellow) signifies a highly efficient calculation.

Figure 3B demonstrates how the GB/SA calculation executed solely on 16 CPU cores is accelerated through the addition of 1 GPU. Because of its powerful computing capability, the GPU completes all Coulomb, van der Waals, and GB calculations in the same time that the 16 CPU cores need to perform the SA calculation, resulting in a 3-fold overall performance increase. This performance increase should be judged on the basis of an approximate 1 GPU/16 CPU cores cost ratio of about 1, i.e., by doubling the hardware cost, one triples performance.

A bottleneck for hybrid GPU/CPU calculations is the CPU's limited ability to switch between performing the SA calculation and GPU-related operations. Figure 3C illustrates the interplay between the GPU execution and the host CPU calculations. For each of the three GB phases, the host CPU initializes the GB calculation on the GPU, then, while the GPU calculates the GB phase, the host CPU performs SA calculations. NAMD efficiently overlaps the GB and SA calculations by partitioning the SA calculation into many short, independent force work units, such that at the completion of each SA force work unit, the CPU can engage in GPU-related operations, if needed, then return to the next SA force work unit as highlighted by Figure 3C. Figure 3D illustrates how not partitioning the SA calculation increases GPU idle time during GB phase 1 and CPU idle time during GB phases 2 and 3, thereby decreasing performance.

Because of the performance difference between GPUs and CPUs and the different computational cost of the GB and SA calculations, it was not known what ratio of GPUs to CPUs would be the most efficient. The best ratio would have GPUs and CPUs requiring the same time to perform their respective force calculations. On the basis of the data in Table 2 and as illustrated in Figure 3, a ratio of 16 CPU cores per GPU allowed highly efficient performance, while deviating from this

Table 2. Benchmark Simulation Speeds, in Units of Seconds/Time Step, for the Four Benchmarked Protein Systems Tested on 0−4 GPUs and 1−32 CPU Cores[a]

		1YRI			2W5U			1GPB			1IR2		
GPU	CPU	vacu	GB	GB/SA	vacu	GB	GB/SA	vacu	GB	GB/SA	vacu	GB	GB/SA
0	1	8.4×10^{-3}	4.9×10^{-2}	7.3×10^{-2}	4.9×10^{-2}	3.5×10^{-1}	5.2×10^{-1}	3.4×10^{-1}	2.4×10^{0}	3.4×10^{0}	2.3×10^{0}	1.6×10^{1}	2.2×10^{1}
	2	5.7×10^{-3}	2.7×10^{-2}	4.0×10^{-2}	2.7×10^{-2}	1.8×10^{-1}	2.6×10^{-1}	1.8×10^{-1}	1.2×10^{0}	1.7×10^{0}	1.2×10^{0}	9.5×10^{0}	1.2×10^{1}
	4	3.8×10^{-3}	1.5×10^{-2}	2.2×10^{-2}	1.4×10^{-2}	9.3×10^{-2}	1.3×10^{-1}	9.4×10^{-2}	6.2×10^{-1}	8.8×10^{-1}	5.9×10^{-1}	5.1×10^{0}	7.2×10^{0}
	8	2.6×10^{-3}	8.7×10^{-3}	1.3×10^{-2}	8.0×10^{-3}	4.7×10^{-2}	6.8×10^{-2}	4.8×10^{-2}	3.2×10^{-1}	4.4×10^{-1}	3.0×10^{-1}	2.1×10^{0}	3.4×10^{0}
	16	2.2×10^{-3}	5.6×10^{-3}	7.5×10^{-3}	4.8×10^{-3}	2.6×10^{-2}	3.7×10^{-2}	2.6×10^{-2}	1.6×10^{-1}	2.3×10^{-1}	1.6×10^{-1}	1.1×10^{0}	1.5×10^{0}
	32	1.7×10^{-3}	3.7×10^{-3}	4.9×10^{-3}	3.1×10^{-3}	1.4×10^{-2}	2.1×10^{-2}	1.4×10^{-2}	8.3×10^{-2}	1.2×10^{-1}	8.2×10^{-2}	5.4×10^{-1}	7.3×10^{-1}
1	1	3.6×10^{-3}	3.8×10^{-3}	3.1×10^{-2}	8.1×10^{-3}	1.1×10^{-2}	1.7×10^{-1}	4.3×10^{-2}	6.7×10^{-2}	1.0×10^{0}	2.3×10^{-1}	4.3×10^{-1}	6.0×10^{0}
	2	3.2×10^{-3}	4.4×10^{-3}	1.9×10^{-2}	5.0×10^{-3}	1.1×10^{-2}	8.9×10^{-2}	2.3×10^{-2}	5.8×10^{-2}	5.2×10^{-1}	1.1×10^{-1}	3.9×10^{-1}	3.1×10^{0}
	4	2.1×10^{-3}	3.4×10^{-3}	1.2×10^{-2}	3.6×10^{-3}	9.9×10^{-3}	4.6×10^{-2}	1.4×10^{-2}	5.4×10^{-2}	2.7×10^{-1}	8.1×10^{-2}	3.5×10^{-1}	1.6×10^{0}
	8	1.8×10^{-3}	3.3×10^{-3}	6.9×10^{-3}	2.8×10^{-3}	1.0×10^{-2}	2.7×10^{-2}	1.3×10^{-2}	5.1×10^{-2}	1.4×10^{-1}	7.5×10^{-2}	3.3×10^{-1}	8.0×10^{-1}
	16	2.1×10^{-3}	3.3×10^{-3}	4.8×10^{-3}	2.7×10^{-3}	1.1×10^{-2}	1.8×10^{-2}	1.3×10^{-2}	5.4×10^{-2}	8.1×10^{-2}	7.1×10^{-2}	3.2×10^{-1}	4.3×10^{-1}
	32				2.8×10^{-3}	8.9×10^{-3}	1.2×10^{-2}	1.2×10^{-2}	5.0×10^{-2}	6.1×10^{-2}	7.0×10^{-2}	3.2×10^{-1}	3.5×10^{-1}
2	2	3.0×10^{-3}	3.9×10^{-3}	2.0×10^{-2}	5.0×10^{-3}	9.2×10^{-3}	8.9×10^{-2}	2.4×10^{-2}	4.4×10^{-2}	5.2×10^{-1}	1.3×10^{-1}	2.6×10^{-1}	3.1×10^{0}
	4	2.3×10^{-3}	3.4×10^{-3}	1.1×10^{-2}	3.7×10^{-3}	9.2×10^{-3}	5.0×10^{-2}	1.4×10^{-2}	3.7×10^{-2}	2.7×10^{-1}	7.1×10^{-2}	2.2×10^{-1}	1.6×10^{0}
	8	1.8×10^{-3}	3.2×10^{-3}	7.9×10^{-3}	2.5×10^{-3}	8.6×10^{-3}	2.9×10^{-2}	1.0×10^{-2}	3.6×10^{-2}	1.4×10^{-1}	4.9×10^{-2}	2.0×10^{-1}	8.1×10^{-1}
	16	1.7×10^{-3}	3.2×10^{-3}	5.8×10^{-3}	2.7×10^{-3}	8.7×10^{-3}	1.9×10^{-2}	8.9×10^{-3}	3.8×10^{-2}	8.1×10^{-2}	4.4×10^{-2}	1.9×10^{-1}	4.0×10^{-1}
	32				2.5×10^{-3}	7.6×10^{-3}	1.3×10^{-2}	7.9×10^{-3}	3.4×10^{-2}	5.3×10^{-2}	4.4×10^{-2}	2.1×10^{-1}	2.7×10^{-1}
4	4	2.7×10^{-3}	3.4×10^{-3}	1.2×10^{-2}	3.5×10^{-3}	1.0×10^{-2}	5.0×10^{-2}	1.4×10^{-2}	3.0×10^{-2}	2.7×10^{-1}	7.2×10^{-2}	1.6×10^{-1}	1.6×10^{0}
	8	1.8×10^{-3}	3.1×10^{-3}	1.2×10^{-2}	2.5×10^{-3}	8.1×10^{-3}	3.0×10^{-2}	8.5×10^{-3}	2.6×10^{-2}	1.4×10^{-1}	3.9×10^{-2}	1.2×10^{-1}	8.0×10^{-1}
	16	2.2×10^{-3}	3.2×10^{-3}	6.3×10^{-3}	2.6×10^{-3}	7.9×10^{-3}	2.1×10^{-2}	7.2×10^{-3}	2.8×10^{-2}	9.1×10^{-2}	2.9×10^{-2}	1.1×10^{-1}	4.1×10^{-1}
	32				2.4×10^{-3}	6.7×10^{-3}	1.2×10^{-2}	6.4×10^{-3}	2.3×10^{-2}	4.7×10^{-2}	2.7×10^{-2}	1.2×10^{-1}	2.4×10^{-1}

[a]Speeds are reported for three increasingly accurate and expensive simulation types: in vacuo (vacu), GB, and GB/SA. The small size of system 1YRI, see Figure 2A, prohibited utilization of 32 CPU cores in some cases.

Table 3. Computational Expense and Accuracy for Different Nonbonded Interaction Cutoff Lengths[a]

	no cutoff				30 Å cutoff					16 Å cutoff				
PDB	time	vac	GB	SA	time	vac	GB	SA	err	time	vac	GB	SA	err
1IR2	556 .0s	21%	76%	3%	54.9s	19%	69%	12%	0.04%	22.0s	10%	62%	28%	0.3%
1GPB	17.8 s	19%	74%	7%	6.61s	15%	72%	13%	0.3%	3.4s	10%	60%	30%	0.6%
2W5U	0.61 s	15%	75%	10%	0.57s	15%	73%	10%	0.04%	0.52s	10%	57%	33%	0.7%
1YRI	0.05 s	15%	60%	25%	0.05s	14%	59%	27%	0.3%	0.07s	8%	59%	33%	0.3%

[a]Listed for each of the four benchmarked systems and for each cutoff (no cutoff, 30 Å cutoff, and 16 Å cutoff) are the execution time required for one time step on one CPU core, the percent of computational expense associated with the Coulomb and van der Waals (vac), GB and SA calculations, and the relative error in the total GB energy calculated with cutoff, namely, $(E^{GB}_{cutoff} - E^{GB}_{no\text{-}cutoff})/E^{GB}_{no\text{-}cutoff}$.

ratio resulted in either the GB or SA calculation becoming rate limiting. For example, Table 2 shows that calculating GB/SA for the 1GPB system on one GPU is 1.9 times faster with 16 CPU cores than with eight CPU cores (adding more cores alleviates the SA calculation bottleneck on the CPU), while one GPU with 32 CPU cores is only 1.3 times faster than with 16 CPU cores (GB calculation on GPU begins to be the bottleneck), thereby sharply decreasing computational efficiency. As both GPU and CPU technologies evolve, the ideal ratio of CPU cores per GPU will also evolve, both for our GB/SA algorithm as well as for other hybrid GPU/CPU algorithms.

Performance Benchmark Results. The performance of the hybrid GB/SA implementation is demonstrated by the benchmarks, which are reported in full in Table 2 and illustrated for the 1IR2 system in Figure 4. The incorporation of GPUs into the MD simulation computation offers dramatic performance increases; for example, for the GB simulations of the 1IR2 system, simulations on one GPU are 25% faster than on 32 CPU cores. Additionally, the three largest systems tested were simulated, with the GB model, 30−35 times faster with one GPU than on one CPU core alone, while even the smallest system, 1YRI, was simulated 13 times faster with one GPU than on one CPU core alone. The GB/SA simulation of the 1IR2 system executed 50 times faster on one GPU with 16 CPU cores than on one CPU core and 3.5 times faster than on 16 CPU cores.

In the case of larger simulated systems, one can take advantage of more GPU accelerators; because GPUs operate most efficiently when they are issued a lot of concurrent work, for moderately sized systems one achieves the great simulation speeds on relatively few GPUs (see the GB columns of Table 2). The 582-atom villin headpiece is simulated at the same speed on one GPU as on 32 CPU cores, but not any faster on two GPUs. The larger 2412-atom flavodoxin is simulated 27% faster on one GPU than on 32 CPU cores, and 60% faster on two GPUs. The 13 340-atom glycogen phosphorylase is simulated 3.2 times faster on four GPUs than on 32 CPU cores while the 74 926-atom RuBisCO is simulated 4.5 times faster for the same comparison.

To enhance performance for large systems, we employed in the present study an interaction cutoff. Because cutoff lengths affect both accuracy and performance, we compare in Table 3 the computational expense of simulating the four benchmark systems as well as the relative error in calculating the total GB energy using a no-cutoff calculation as a reference, $(E^{GB}_{cutoff} - E^{GB}_{no\text{-}cutoff})/E^{GB}_{no\text{-}cutoff}$. In the case of the 16 Å cutoff, compared to a no-cutoff calculation, computational expense is reduced, the relative error is about 0.5%, and the fraction of computational expense associated with the SA calculation is higher (requiring more CPU cores per GPU).

The impressive speedups achieved by utilizing GPU technology demonstrate the benefit which GPUs offer to MD computing. While the benchmarks reported here are only for up to four GPUs, the hybrid algorithm also operates on supercomputers built from many GPUs and multicore CPUs as previously described for GPU acceleration of NAMD on distributed memory computers.[38]

Testing Surface Tension Parameters. When employing the SA calculation, the surface tension, in units of kcal/mol/Å^2, is a parameter of the model, controlling the hydrophobic energy of solvation; it is not a physical property measured from the simulation. To explore the effect of the parameter on a protein system, the 2W5U system was equilibrated through a GB/SA implicit solvent simulation employing surface tension parameters ranging from 0.001 to 0.128 kcal/mol/Å^2; the system was also equilibrated through a standard all-atom simulation employing explicit TIP3P[69] solvent. Figure 5 presents protein surface area arising in the GB/SA equilibration simulations as well as the final surface area of each. The protein surface area is expected to diminish with increasing surface tension as the latter imparts an energy penalty for the protein's surface exposed to solvent. The simulations utilizing surface tension parameter values of 0.004 and 0.008 kcal/mol/Å^2 returned final surface areas closest to the area of the reference TIP3P equilibrated system, suggesting that employing surface tension parameter values between 0.004 and 0.008 kcal/mol/Å^2 most closely reproduce protein behavior in explicit solvent. The experimentally measured surface tension of hydrocarbons in aqueous solvent is 0.005 kcal/mol/Å^2,[62] which validates our finding.

CONCLUSIONS

The structural biology community has been well served by technological advances of general purpose GPU computing, which have made molecular dynamics more powerful and accurate. Many biological computing programs have already achieved great improvements in performance through GPU acceleration. As it is often neither feasible nor ideal to perform all needed calculations on the GPU, it becomes increasingly important to develop methods for overlapping GPU and CPU calculations. From the present work results an efficient method for performing GB/SA simulations on hybrid GPU/CPU computers, permitting extremely fast and accurate molecular dynamics simulations that can be executed, for example, interactively by researchers.

AUTHOR INFORMATION

Corresponding Author

*E-mail: kschulte@ks.uiuc.edu.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors thank John Stone for helpful discussions regarding implementation and performance optimization for GPUs. This work was supported by the National Institutes of Health (NIH P41-RR005969).

■ REFERENCES

(1) Le, L.; Lee, E. H.; Schulten, K.; Truong, T. *PLoS Currents: Influenza* **2010**, 2009 Aug 27:RRN1015.
(2) Le, L.; Lee, E. H.; Hardy, D. J.; Truong, T. N.; Schulten, K. *PLoS Comput. Biol.* **2010**, *6*, e1000939.
(3) Cheng, L. S.; Amaro, R. E.; Xu, D.; Li, W. W.; Arzberger, P. W.; McCammon, J. A. *J. Med. Chem.* **2008**, *51*, 3878−3894.
(4) Acharya, R.; Carnevale, V.; Fiorin, G.; Levine, B. G.; Polishchuk, A. L.; Balannik, V.; Samish, I.; Lamb, R. A.; Pinto, L. H.; DeGrado, W. F.; Klein, M. L. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 15075−15080.
(5) Khurana, E.; Peraro, M. D.; DeVane, R.; Vemparala, S.; DeGrado, W. F.; Klein, M. L. *Proc. Natl. Acad. Sci. U.S.A.* **2009**, *106*, 1069−1074.
(6) Khurana, E.; DeVane, R. H.; Peraro, M. D.; Klein, M. L. *Biochim. Biophys. Acta* **2011**, *1808*, 530−537.
(7) Newhouse, E. I.; Xu, D.; Markwick, P. R. L.; Amaro, R. E.; Pao, H. C.; Wu, K. J.; Alam, M.; McCammon, J. A.; Li, W. W. *J. Am. Chem. Soc.* **2009**, *131*, 17430−17442.
(8) Fidelak, J.; Juraszek, J.; Branduardi, D.; Bianciotto, M.; Gervasio, F. L. *J. Phys. Chem. B* **2010**, *114*, 9516−9524.
(9) Lee, E. H.; Hsin, J.; Sotomayor, M.; Comellas, G.; Schulten, K. *Structure* **2009**, *17*, 1295−1306.
(10) Trabuco, L. G.; Villa, E.; Mitra, K.; Frank, J.; Schulten, K. *Structure* **2008**, *16*, 673−683.
(11) Davidovich, C.; Bashan, A.; Yonath, A. *Proc. Natl. Acad. Sci. U.S.A.* **2008**, *105*, 20665−20670.
(12) Poehlsgaard, J.; Douthwaite, S. *Nat. Rev. Microbiol.* **2005**, *3*, 870−881.
(13) Lim, B.; Lee, E. H.; Sotomayor, M.; Schulten, K. *Structure* **2008**, *16*, 449−459.
(14) Hsin, J.; Strümpfer, J.; Lee, E. H.; Schulten, K. *Annu. Rev. Biophys.* **2011**, *40*, 187−203.
(15) Venkatesan, B.; Polans, J.; Comer, J.; Sridhar, S.; Wendell, D.; Aksimentiev, A.; Bashir, R. *Biomed. Microdev.* **2011**, 1−12.
(16) Carr, R.; Comer, J.; Ginsberg, M. D.; Aksimentiev, A. *J. Phys. Chem. Lett.* **2011**, *2*, 1804−1807.
(17) Freddolino, P. L.; Arkhipov, A. S.; Larson, S. B.; McPherson, A.; Schulten, K. *Structure* **2006**, *14*, 437−449.
(18) Arkhipov, A.; Freddolino, P. L.; Schulten, K. *Structure* **2006**, *14*, 1767−1777.
(19) Tavoosi, N.; Davis-Harrison, R. L.; Pogorelov, T. V.; Ohkubo, Y. Z.; Arcario, M. J.; Clay, M. C.; Rienstra, C. M.; Tajkhorshid, E.; Morrissey, J. H. *J. Biol. Chem.* **2011**, *286*, 23247−23253.
(20) Ohkubo, Y. Z.; Tajkhorshid, E. *Structure* **2008**, *16*, 72−81.
(21) Ohkubo, Y. Z.; Morrissey, J. H.; Tajkhorshid, E. *J. Thromb. Haem.* **2010**, *8*, 1044−1053.
(22) Morrissey, J. H.; Pureza, V.; Davis-Harrison, R. L.; Sligar, S. G.; Rienstra, C. M.; Kijac, A. Z.; Ohkubo, Y. Z.; Tajkhorshid, E. *J. Thromb. Haem.* **2009**, *7*, 169−172.
(23) Morrissey, J. H.; Davis-Harrison, R. L.; Tavoosi, N.; Ke, K.; Pureza, V.; Boettcher, J. M.; Clay, M. C.; Rienstra, C. M.; Ohkubo, Y. Z.; Pogorelov, T. V.; Tajkhorshid, E. *Thromb. Res.* **2010**, *125* (Suppl. 1), S23−S25.
(24) Morrissey, J. H.; Pureza, V.; Davis-Harrison, R. L.; Sligar, S. G.; Ohkubo, Y. Z.; Tajkhorshid, E. *Thromb. Res.* **2008**, *122*, S23−S26.
(25) Interlandi, G.; Thomas, W. *Proteins: Struct., Funct., Genet.* **2010**, *78*, 2506−2522.
(26) Miller, Y.; Ma, B.; Nussinov, R. *J. Am. Chem. Soc.* **2011**, *133*, 2742−2748.
(27) Parthasarathy, S.; Long, F.; Miller, Y.; Xiao, Y.; McElheny, D.; Thurber, K.; Ma, B.; Nussinov, R.; Ishii, Y. *J. Am. Chem. Soc.* **2011**, *133*, 3390−3400.
(28) Miller, Y.; Ma, B.; Tsai, C.-J.; Nussinov, R. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 14128−14133.
(29) Fogolari, F.; Corazza, A.; Viglino, P.; Zuccato, P.; Pieri, L.; Faccioli, P.; Bellotti, V.; Esposito, G. *Biophys. J.* **2007**, *92*, 1673−1681.
(30) Buchete, N.-V.; Hummer, G. *Biophys. J.* **2007**, *92*, 3032−3039.
(31) Buchete, N.-V.; Tycko, R.; Hummer, G. *J. Mol. Biol.* **2005**, *353*, 804−821.
(32) Shaw, D. E.; Dror, R. O.; Salmon, J. K.; Grossman, J.; Mackenzie, K. M.; Bank, J. A.; Young, C.; Deneroff, M. M.; Batson, B.; Bowers, K. J.; Chow, E.; Eastwood, M. P.; Ierardi, D. J.; Klepeis, J. L.; Kuskin, J. S.; Larson, R. H.; Lindorff-Larsen, K.; Maragakis, P.; Moraes, M. A.; Piana, S.; Shan, Y.; Towles, B. Millisecond-Scale Molecular Dynamics Simulations on Anton. *SC'09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, New York, NY, 2009; pp 39:1−39:11.
(33) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(34) Kalé, L.; Skeel, R.; Bhandarkar, M.; Brunner, R.; Gursoy, A.; Krawetz, N.; Phillips, J.; Shinozaki, A.; Varadarajan, K.; Schulten, K. *J. Comput. Phys.* **1999**, *151*, 283−312.
(35) Schulz, R.; Lindner, B.; Petridis, L.; Smith, J. C. *J. Chem. Theory Comput.* **2009**, *5*, 2798−2808.
(36) Mei, C.; Sun, Y.; Zheng, G.; Bohm, E. J.; Kalé, L. V.; Phillips, J. C.; Harrison, C. Enabling and Scaling Biomolecular Simulations of 100 Million Atoms on Petascale Machines with a Multicore-optimized Message-driven Runtime. *Proceedings of the 2011 ACM/IEEE Conference on Supercomputing*, Seattle, WA, 2011.
(37) Stone, J. E.; Phillips, J. C.; Freddolino, P. L.; Hardy, D. J.; Trabuco, L. G.; Schulten, K. *J. Comput. Chem.* **2007**, *28*, 2618−2640.
(38) Phillips, J. C.; Stone, J. E.; Schulten, K. Adapting a Message-Driven Parallel Application to GPU-Accelerated Clusters. *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, Piscataway, NJ, 2008.
(39) Stone, J. E.; Hardy, D. J.; Isralewitz, B.; Schulten, K. In *Scientific Computing with Multicore and Accelerators*; Dongarra, J., Bader, D. A., Kurzak, J., Eds.; Chapman & Hall/CRC Press: New York, 2011; Chapter 16, pp 351−371.
(40) Hardy, D. J.; Stone, J. E.; Vandivort, K. L.; Gohara, D.; Rodrigues, C.; Schulten, K. In *GPU Computing Gems*; Hwu, W., Ed.; Morgan Kaufmann Publishers: 2011; Chapter 4, pp 43−58.
(41) Stone, J. E.; Hardy, D. J.; Saam, J.; Vandivort, K. L.; Schulten, K. In *GPU Computing Gems*; Hwu, W., Ed.; Morgan Kaufmann Publishers: Waltham, MA, 2011; Chapter 1, pp 5−18.
(42) Roberts, E.; Stone, J. E.; Sepulveda, L.; Hwu, W. W.; Luthey-Schulten, Z. Long time-scale simulations of in vivo diffusion using GPU hardware. *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*; 2009; pp 1−8.
(43) Brooks, B. R.; Brooks, C. L.; Mackerell, A. D.; Nilsson, L.; Petrella, R. J.; Roux, B.; Won, Y.; Archontis, G.; Bartels, C.; Boresch, S.; Caflisch, A.; Caves, L.; Cui, Q.; Dinner, A. R.; Feig, M.; Fischer, S.; Gao, J.; Hodoscek, M.; Im, W.; Kuczera, K.; Lazaridis, T.; Ma, J.; Ovchinnikov, V.; Paci, E.; Pastor, R. W.; Post, C. B.; Pu, J. Z.; Schaefer, M.; Tidor, B.; Venable, R. M.; Woodcock, H. L.; Wu, X.; Yang, W.; York, D. M.; Karplus, M. *J. Comput. Chem.* **2009**, *30*, 1545−1614.
(44) Harvey, M. J.; Giupponi, G.; Fabritiis, G. D. *J. Chem. Theory Comput.* **2009**, *5*, 1632−1639.
(45) Baker, J. A.; Hirst, J. D. *Mol. Inf.* **2011**, *30*, 498−504.
(46) Friedrichs, M. S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; Legrand, S.; Beberg, A. L.; Ensign, D. L.; Bruns, C. M.; Pande, V. S. *J. Comput. Chem.* **2009**, *30*, 864−872.
(47) Eastman, P.; Pande, V. S. *J. Comput. Chem.* **2010**, *31*, 1268−1272.
(48) Anderson, J. A.; Lorenz, C. D.; Travesset, A. *J. Chem. Phys.* **2008**, *227*, 5342−5359.
(49) Liu, F.; Gruebele, M. *J. Mol. Biol.* **2007**, *370*, 574−584.

(50) Izrailev, S.; Stepaniants, S.; Isralewitz, B.; Kosztin, D.; Lu, H.; Molnar, F.; Wriggers, W.; Schulten, K. In *Computational Molecular Dynamics: Challenges, Methods, Ideas*; Deuflhard, P., Hermans, J., Leimkuhler, B., Mark, A. E., Reich, S., Skeel, R. D., Eds.; Springer-Verlag: Berlin, 1998; Lecture Notes in Computational Science and Engineering Vol. *4*; pp 39−65.
(51) Wells, D. B.; Abramkina, V.; Aksimentiev, A. *J. Chem. Phys.* **2007**, *127*, 125101.
(52) Shivakumar, D.; Deng, Y.; Roux, B. *J. Chem. Theory Comput.* **2009**, *5*, 919−930.
(53) Still, W. C.; Tempczyk, A.; Hawley, R. C.; Hendrickson, T. *J. Am. Chem. Soc.* **1990**, *112*, 6127−6129.
(54) Weiser, J.; Shenkin, P. S.; Still, W. C. *J. Comput. Chem.* **1998**, *20*, 217−230.
(55) Onufriev, A.; Bashford, D.; Case, D. A. *J. Phys. Chem.* **2000**, *104*, 3712−3720.
(56) Onufriev, A.; Bashford, D.; Case, D. A. *Proteins: Struct., Funct., Bioinf.* **2004**, *55*, 383−394.
(57) Case, D.; Cheatham, T., III; Darden, T.; Gohlke, H.; Luo, R.; Merz, K., Jr; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. *J. Comput. Chem.* **2005**, *26*, 1668.
(58) Tanner, D. E.; Chan, K.-Y.; Phillips, J.; Schulten, K. *J. Chem. Theory Comput.* **2011**, *7*, 3635−3642.
(59) Hassan, S. A.; Mehler, E. L. *Proteins: Struct., Funct., Genet.* **2002**, *47*, 45−61.
(60) Hassan, S. A.; Mehler, E. L.; Zhang, D.; Weinstein, H. *Proteins: Struct., Funct., Genet.* **2003**, *51*, 109−125.
(61) Srinivasan, J.; Trevathan, M. W.; Beroza, P.; Case, D. A. *Theor. Chim. Acta* **1999**, *101*, 426−434.
(62) Sitkoff, D.; Sharp, K. A.; Honig, B. *J. Phys. Chem.* **1994**, *98*, 1978−1988.
(63) Humphrey, W.; Dalke, A.; Schulten, K. *J. Mol. Graphics* **1996**, *14*, 33−38.
(64) Rodrigues, C. I.; Hardy, D. J.; Stone, J. E.; Schulten, K.; Hwu, W. W. GPU Acceleration of Cutoff Pair Potentials for Molecular Modeling Applications. *CF'08: Proceedings of the 2008 Conference on Computing Frontiers*, New York, NY, 2008; pp 273−282.
(65) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. *J. Mol. Graphics Modell.* **2010**, *29*, 116−125.
(66) Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327−341.
(67) Zhu, J.; Shi, Y.; Liu, H. *J. Phys. Chem. B* **2002**, *106*, 4844−4853.
(68) Daidone, I.; Ulmschneider, M. B.; Nola, A. D.; Amadei, A.; Smith, J. C. *Proc. Natl. Acad. Sci. U.S.A.* **2007**, *104*, 15230−15235.
(69) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926−935.
(70) Kalé, L. V.; Zheng, G.; Lee, C. W.; Kumar, S. *Fut. Gen. Comp. Sys.* **2006**, *22*, 347−358.