JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

Accelerated Conformational Entropy Calculations Using Graphic Processing Units

Qian Zhang,[†] Junmei Wang,[‡] Ginés D. Guerrero,[§] José M. Cecilia,[∥] José M. García,[§] Youyong Li,[†] Horacio Pérez-Sánchez,*[,∥] and Tingjun Hou*[,†,‡]

[†]Institute of Functional Nano & Soft Materials (FUNSOM) and Jiangsu Key Laboratory for Carbon-Based Functional Materials & Devices, Soochow University, Suzhou, Jiangsu 215123, China

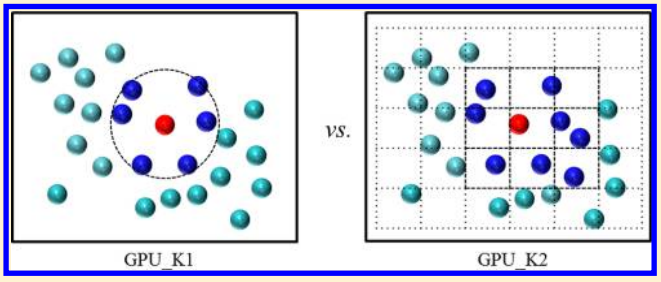
[‡]College of Pharmaceutical Sciences, Zhejiang University, Hangzhou, Zhejiang 310058, China

[§]Computer Engineering Department, School of Computer Science, 30100 University of Murcia, Spain

[∥]Computer Science Department, 30107 Catholic University of Murcia (UCAM), Spain

Supporting Information

ABSTRACT: Conformational entropy calculation, usually computed by normal-mode analysis (NMA) or quasi harmonic analysis (QHA), is extremely time-consuming. Here, instead of NMA or QHA, a solvent accessible surface area (SASA) based model was employed to compute the conformational entropy, and a new fast GPU-based method called MURCIA (Molecular Unburied Rapid Calculation of Individual Areas) was implemented to accelerate the calculation of SASA for each atom. MURCIA employs two different kernels to determine the neighbors of each atom. The first kernel (K1) uses brute force for the calculation of the neighbors of atoms, while the second one (K2) uses an advanced algorithm involving hardware interpolations via GPU texture memory unit for such purpose. These two kernels yield very similar results. Each kernel has its own advantages depending on the protein size. K1 performs better than K2 when the size is small and vice versa. The algorithm was extensively evaluated for four protein data sets and achieves good results for all of them. This GPU-accelerated version is ~600 times faster than the former sequential algorithm when the number of the atoms in a protein is up to 10^5.



1. INTRODUCTION

Prediction of free energy, especially binding free energy, is one of the central interests in computational chemistry and computational biology. Many approaches have been developed for the prediction of binding free energy, such as thermodynamic integration (TI),[1] free energy perturbation (FEP),[1] linear interaction energy (LIE) approach,[2] molecular mechanics/Poisson−Boltzmann surface area (MM/PBSA),[3,4] molecular mechanics/Generalized Born surface area (MM/GBSA),[3,4] etc. The main idea of TI and FEP is to compute the free energy difference (ΔG) between two similar structures from molecular dynamics (MD) and Monte Carlo (MC) simulations by summing the ΔGs of the transient states between two different structures. TI and FEP are theoretically rigorous, but they are computationally expensive, thus impeding their applications for large scale computations. For LIE, different fitted parameters are needed while applying this method into different systems, and therefore it is system-dependent and not convenient.[2] Recently, the MM/GBSA and MM/PBSA approaches are becoming more and more popular due to its high efficiency.[5−15] In the MM/PBSA and MM/GBSA theory, the free energy of a molecule is calculated using the following equations[3,4]

$$G = H_{gas} + G_{solv} - TS_{conf} \quad (1)$$

$$G_{solv} = G_{solv}^{pol} + G_{solv}^{nonpol} \quad (2)$$

$$S_{conf} = S_{trans} + S_{rot} + S_{vib} \quad (3)$$

where, in eq 1, G represents the Gibbs free energy, H_{gas} represents the gas phase enthalpy, which can be roughly replaced by E_{gas} for a molecule in condensed phase. G_{solv} consists of two parts, the polar and nonpolar solvation free energies. The polar part is usually calculated by either a Poisson−Boltzmann (PB) or Generalized Born (GB) model, and the nonpolar part is estimated by the solvent accessible surface area (SASA).[3] Certainly, whether G_{solv}^{nonpol} can be made a sole function of SASA is still an ongoing and unsettled debate. S_{conf} was decomposed into three components: S_{trans}, S_{rot}, and S_{vib}. S_{trans} and S_{rot} are translational and rotational contributions of entropy, and they can be directly estimated by the standard equations. The vibrational entropy (S_{vib}) is usually estimated by quasi harmonic analysis (QHA) of MD trajectories or by normal-mode analysis (NMA) on selected snapshots.[4]

Conformation entropy calculations by QHA or NMA is time-consuming, and therefore we have developed a SASA-based

Received: May 3, 2013
Published: July 18, 2013

approach to calculate S_{conf}.[6] This method shows good efficiency and accuracy when compared with NMA. In our method, the Shrake-Rupley algorithm[16] was employed to generate a mesh of points to represent the sphere of each atom. Then, the distance between any two points in the surface and core of each atom is calculated to determine which sphere points are accessible to solvent. Afterward, the SASA of each atom is computed by taking into account the proportion of all points accessible to solvent. Finally, based on a set of fitted parameters for different atom types, the calculated SASAs are used to compute S_{conf}. This method does perform well for small molecules, but it is computationally expensive for large molecules. For example, when the atom number of a protein is over 20,000, the computational cost for calculating the SASAs is more than 180 sec on an Intel-based Xeon processor used in this study. Therefore, the calculations of SASAs become the computational bottleneck for relatively large systems.

In this study, in order to accelerate the calculation of SASAs, a GPU (Graphic Processing Unit)-based algorithm named MURCIA (Molecular Unburied Rapid Calculation of Individual Areas) was implemented. The use of GPUs has attracted great attention recently for scientific computing. GPU is not only a graphics engine but also a processor that provides excellent performance in parallel calculations. It is designed for a particular class of applications with the following features: computational requirements are extensive, parallelism is substantial, and throughput is more important than latency.[17] The applications for GPUs are becoming more and more popular in different fields of computational biology and computer-aided drug design (CADD),[18−38] such as electrostatics/solvation and free energy calculations,[19,23,30,31,34] electron transfer calculations,[21] electron repulsion calculations,[29] molecular docking,[25,35] molecular dynamics simulations,[18,33,39] similarity and database searching,[20,24,26,37,38] normal-mode analysis,[28] etc. Moreover, many molecular simulation packages have been updated in line with the GPU architecture. For example, AMBER includes full GPU support for the MD simulations in PMEMD;[19] VMD employs GPUs for displaying and animating large biomolecular systems using 3-D graphics and built-in scripting;[40] and NAMD, the first MD package implemented on GPU, achieves a significantly improved boost of performance with GPUs.[39]

Here, we used CUDA (Compute Unified Device Architecture) that is NVIDIA's parallel computing architecture to develop our algorithm.[41] The flowchart of the algorithm for entropy calculations is shown in Figure 1. The key point is to compute the neighbor atoms and then the SASA of each atom. Once the SASA of each atom has been computed, the following formula is used to calculate the entropy of a molecule

$$S = \sum_{i=1}^{N} w_i(SASA_i + kBSASA_i) \quad (4)$$

$$BSASA_i = 4\pi(r_i + r_{prob})^2 - SASA_i \quad (5)$$

where w_i has already been parametrized in our previous study,[6] r_{prob} is 0.8 Å, $SASA_i$ represents the SASA of atom i, and $BSASA_i$ stands for the buried SASA of atom i. k is set to be 0.461, which was found to be the most suitable value in this formula. S represents S_{conf}, and it is the weighted sum of the $SASA$ and $BSASA$ for each atom.

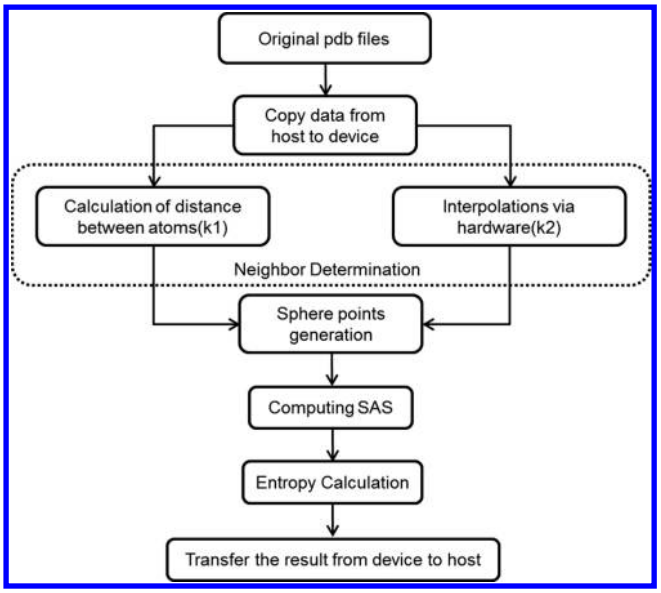


Figure 1. Flowchart for entropy calculation using MURCIA. Neighbor calculation is carried out either using distances (K1) or via hardware interpolation (K2).

2. MATERIALS AND METHODS

This section describes how the calculation of entropy is carried out started from the values of the radii and the positions of the atoms in a molecule. First, the SASA calculation using an improved version of MURCIA[42] was described, and then the conformational entropy was calculated. The information of CPU, GPU, and CUDA is listed below:

CPU: Intel Xeon E5506 @ 2.13 GHz

GPU: NVIDIA Tesla C2075, 448 CUDA Cores @1147 MHz (1.15 GHz)

CUDA: Driver Version = 4.2, Runtime Version = 4.2

Copying Memory from Host to Device. We use GPU for the SASA calculations, and the processors on GPU cannot access CPU memory directly. One important step is to copy the protein data from host memory to device memory on GPU. In CUDA, we should perform memory allocation and free this memory when calculations are finished. In the MURCIA method protein data are read and stored on CPU, and then the size of required GPU memory is calculated and reserved. Finally, all protein data are copied from host to device.

Neighbor Calculation. Neighbor calculation is necessary for computing SASA, and it is the crucial step to accelerate the entropy calculations. As depicted in Figure 1, two different kernels, K1 and K2, were designed and discussed in the following section.

In K1, each block performs the calculations for one atom. In one block, there are 128 threads by default. For atom i, 128 threads are used to calculate the distances between the other atoms. If the distance is smaller than the predefined cutoff, we then consider the atom as a neighbor. If the number of atoms is smaller than the maximum number of blocks, the block size will be equal to the number of atoms, and if not, it will be set as the maximum number of blocks. The time cost of this kernel is $O(n^2)$, so when the number of atoms increases, the computational cost increases substantially. The pseudocode of this method is shown in Figure 2(A). How the blocks and threads actually work in this method is illustrated in Figure 2(B). All threads in a thread-block cooperate together to

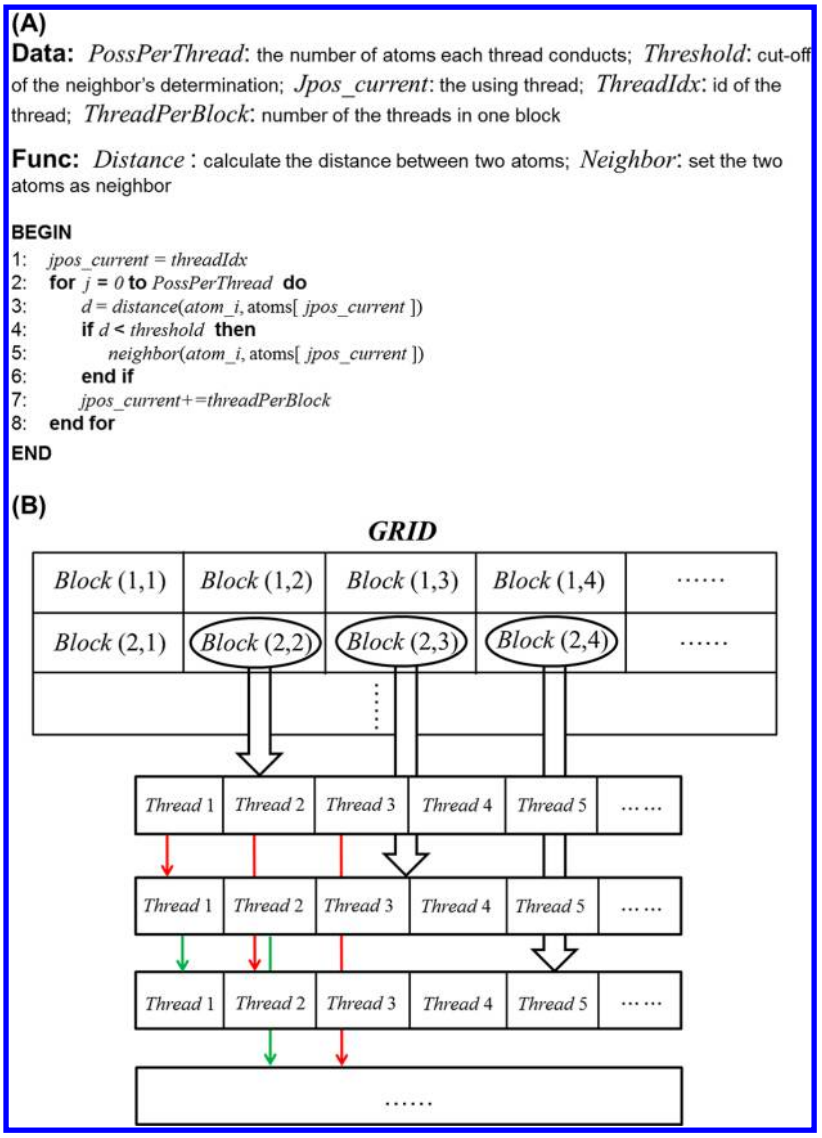


Figure 2. (A) The pseudocode of K1. (B) The way that blocks and threads work in K1. Threads in a block represents one atom, and arrows means the distance calculation between two atoms.

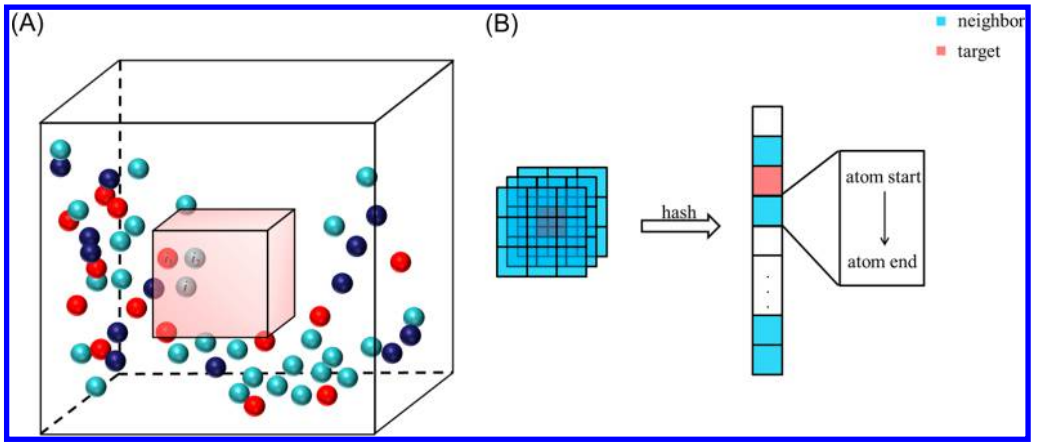


Figure 3. (A) 3D display of target and its neighbors (not marked). (B) 2D typed map of A, and each square represents one box. Targets in A and B are the same one, and both A and B are just part of the whole protein.

calculate all their neighbors using shared memory for sharing common variables to all of them. Calculations between two threads in different blocks are used to generate the distance between two atoms they represent to judge whether they are neighbors or not. For example, thread 1 of Block(2,2) works together with thread 1 of Block(2,3) to confirm if these two

dx.doi.org/10.1021/ci400263t | J. Chem. Inf. Model. 2013, 53, 2057−2064

atoms can be regarded as neighbors, the same for thread 2 of Block(2,2) and thread 2 of Block(2,4).

In order to accelerate the speed of calculation, we developed a new algorithm called K2 to compute the neighbors. In this alternatively designed kernel, the whole system is divided into several cubic grids, and the edge length of one grid is set to be $2r_{max}$, and r_{max} is the maximum radius among the radii of all atoms. Each atom is allocated to a hash value according to its position

$$hash = pos.z*size.y*size.x + pos.y*size.x + pos.x \quad (6)$$

where *pos* is the position of the cell in which the atom locates, and *size* represents the number of cells in each direction.

Same value will be set for the atoms in the same cell, and then they are sorted by the hash values. The sorting function is implemented from the Thrust library,[43] which uses the radix sort method. After the sorting, atoms in the same cell are reordered according to its previous index, and the index of the starting and ending atoms of each cell are calculated. Atoms are regarded as neighbors if they are in the same or surrounding cells. This method will change the complexity from $O(n^2)$ to $O(n)$, because it does not need to go over all atoms to confirm whether this atom is a neighbor or not. Figure 3A and 3B show a brief description of this kernel. Atom i is in one of the cells, and the others in the same cell like i_1, i_2, or the surrounding cells are regarded as neighbors.

Creation of Sphere Points. A regular spherical grid of points is placed over the surface of each atom. This idea can simplify the calculation of SASA by just computing the portion of sphere points which fits the condition we set with its neighbor atoms.

There are two ways to set the sphere points or atomic grid. One is to import a file of sphere point, which contains the predefined grid coordinates, but only 72 and 500 point models are available. The other is to use the grid coordinates that we define in the grids.h file. The number of points ranges from 12 to 1000. The 72 point model was obtained from the Lebedev's algorithm,[44] and the others were gained from the Saff E's sphere grid.[45]

Computing SASAs. After all the preparations have been done, the calculations of SASAs can be accomplished in a very efficient way. Similar to the generation of sphere points, in this part, for K1, each thread is used to qualify whether this point is in the solvent accessible surface of the corresponding atom by computing the distance between this point and the atom's neighbors. If the distance is bigger than the threshold we set, the out points' number $d_{nonburied}$ increases by one. After the calculation is finished, we will get the $SASA_i$ for atom i

$$SASA_i = \frac{d_{nonburied}}{N_{sphere}} * 4\pi(r_i + r_{prob})^2 \quad (7)$$

where N_{sphere} represents the number of each atom's sphere points.

For K2, there is a little difference with respect to K1. The whole system is divided into many cells, and there are several atoms in each cell. For each sphere point of one atom, the distance is calculated between this point and the other atoms which are in the same cell or its neighbor cells. If the distance is bigger than the cutoff, the number of out points increases. The rest of the procedure follows those used by K1.

Calculation of Conformational Entropy. After the calculation of SASA for each atom is finished, the conformational entropy of a molecule is computed by using eqs 4 and 5. These two equations are explained and tested in our previous study.[46] Here we used a GPU-based algorithm to accelerate the process of entropy calculation, and we obtained significant improvements in terms of computing speed.

3. RESULTS AND DISCUSSIONS

We used four data sets to test our method. Data Set I contains 12 protein decoys generated by the Rosetta software package (http://www.rosettacommons.org), Data Set II includes 120 proteins that are randomly chosen from the RCSB protein data bank[47]_ENREF_48 and larger than those used in the first data set, and Data Set III contains 511 PDB files downloaded from the RCSB protein data bank with the number of chains ranging from 15 to 25. Data Set IV contains 175 proteins, which were used for training the protein−protein docking method in ZDOCK.[48]

Validation for 12 Protein Decoys. The first data set was used in our previous study to evaluate the algorithm of entropy calculations. The computational costs of the CPU-based Shrake-Rupley algorithm and the GPU-based MURCIA algorithm for calculating SASAs were compared by averaging the results from 5 independent runs (details are shown in Supporting Information **S1**). The ratio between the two running times was used to compare the performance of these two algorithms, and the results are summarized in Table 1. We can observe that when the number of atoms increases, the running time ratio between CPU and GPU becomes larger.

Table 1. Running Time Ratio between CPU and GPU (K2) for Small Proteins

AtomN[a]	ratio[b]	AtomN	ratio
209	5.195531	462	9.296971
335	7.779898	489	12.29839
342	7.733893	494	12.26562
364	7.451665	512	12.71114
372	8.535273	568	14.61603
410	9.212902	575	12.04921

[a]AtomN represents the number of atoms. [b]Ratio represents the time ratio between CPU and GPU.

The conformational entropies for the proteins in Test set I were predicted based on the SASA values calculated by the CPU-based Shrake-Rupley algorithm and those calculated by the GPU-based MURCIA algorithm. We found that the conformational entropies predicted by the CPU-based and GPU-based algorithms are almost identical with r^2 of 1 (Figure 4). Therefore, the new method based on MURCIA performs as well as the previous one in predicting the conformational entropies for Test set I.

We should mention that for small proteins, the performance of the GPU-based method is not optimal, because GPU cannot generate a sufficient number of threads at the same time. So for the proteins with a small number of atoms (10−100) or we call them small peptides, the calculations of the GPU-based method cannot be substantially accelerated compared with those of the CPU-based version.

Validation for Big Proteins. For the second data set, the running time ratio between CPU and GPU versus the number of atoms is shown in Figure 5. Because of the bigger protein size than the first data set, the results become more significant. As we can see, the ratio arises as a quadratic form, indicating

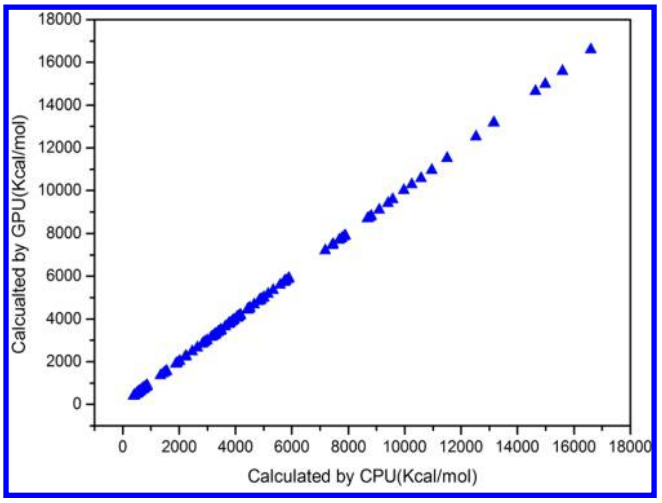


Figure 4. Comparison of the entropies predicted by CPU- and GPU(K2)-based algorithms for small proteins.

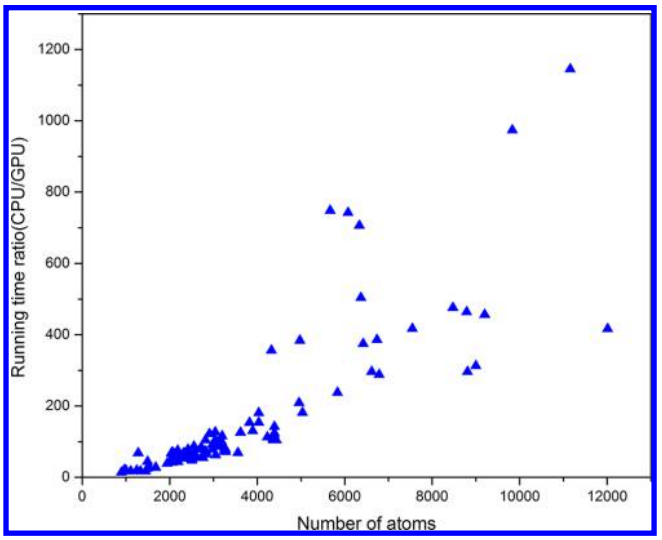


Figure 5. Running time ratio between CPU and GPU (K2) for proteins in the 10^4 size range of atom number.

that the GPU-based method will become faster than the CPU-based version when protein size increases. For K2, as we have introduced before, its time cost is $O(n)$. But for the CPU-based method, though it also cuts the whole system into cubes, the edge length of the cube just changes accordingly with change of the protein size to keep the number of cubes constant. Assuming there are n atoms in one protein and m cubes, the average number of the atoms in each cell is n/m, and it is easy to understand that the time cost of the CPU-based version is $O(n^2)$, so the tendency of the ratio is quite reasonable.

All the above GPU-based results were generated by K2, which computes the neighbor atoms in a very efficient way. We used Data set II and Data set I to evaluate K1 at the same time, and the results are illustrated in Figure 6. From the figure, we can see that when the number of atoms is less than 500, the K1 kernel is faster than the K2 kernel, but when the number of atoms exceeds 500, K2 outperforms K1 in terms of processing speed. The observations are not surprising because in K1, the algorithm just finds out the neighbors and calculates the SASA for each atom, but in K2, a cell sorting should be done first in order to make the further neighbor finding and SASA

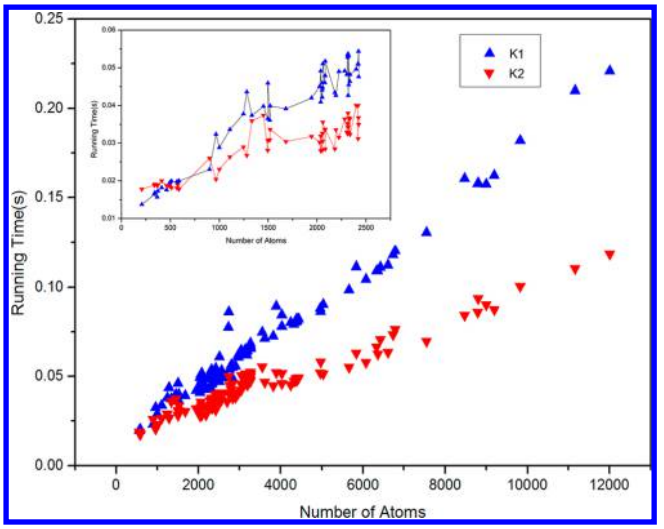


Figure 6. Running times comparison between K1 and K2 in the 10^4 size, and the subfigure is the part with the number of atoms ranging from 0 to 2500.

calculations faster. When the number of atoms is small, each thread just runs once for the calculation of SASA, and therefore K2 needs more time for the extra computation of cell sorting when the number of atoms is not very large.

Validation for Proteins with the Number of Chains Ranging from 15 to 25. Proteins in the third data set are larger than those in the second one. This data set was used to characterize the difference of the computing ability between K2 and K1 when the number of atoms exceeds 10,000 or even 50,000. The results for this data set are illustrated in Figure 7.

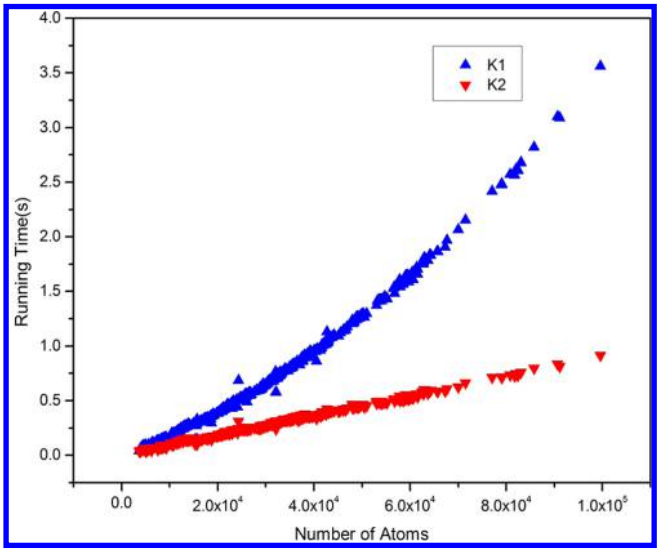


Figure 7. Running time (seconds) in SAS calculation for K1 and K2 kernels in the $1-10^5$ protein size range.

We can make the following conclusion: when K1 was used to generate the neighbor atoms, our new method based on MURCIA can be regarded to be quadratic, and if K2 was used, the MURCIA-based method is just linear with r^2 of 0.99.

The entropy calculations for K2 and K1 were compared to confirm whether this new algorithm can yield good results for this big data set. As shown in Figure 8, r^2 equals to 1. The

dx.doi.org/10.1021/ci400263t | J. Chem. Inf. Model. 2013, 53, 2057−2064

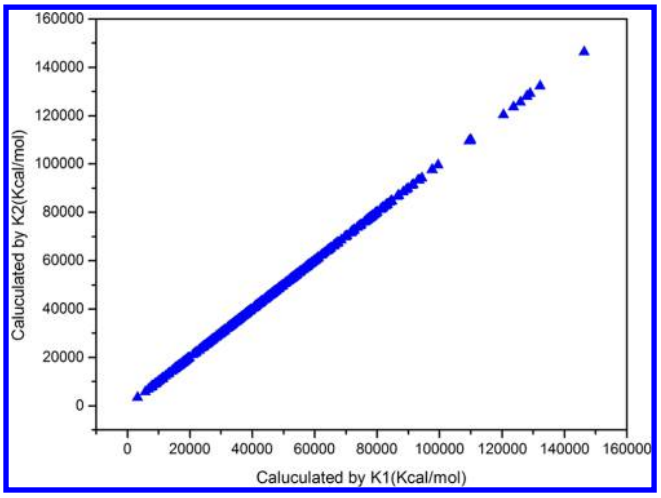


Figure 8. Entropy comparison between K1 and K2.

absolute error Δ was calculated to see if these two results are the same

$$\Delta = |E_{K2} - E_{K1}| \quad (8)$$

where E_{K2} and E_{K1} are the entropies calculated by GPU-based K2 and K1, respectively. The average Δ of the 512 proteins was 0.0014 kcal/mol (detailed information can be seen in Table S2 in Supporting Information), which is a very small value compared to the large entropy results, and obviously the predicted entropies based on the K1 and K2 kernels are almost the same.

According to the three data sets, we can conclude that when the number of atoms is smaller than 500, K1 is a good choice to get the neighbors for each atom. However, when the number goes over 500, K2 will be better than K1, and the difference will become larger when the protein size increases due to the different behavior, quadratic vs linear.

Comparison of the Relative Conformational Entropies. In molecular simulations, the relative conformational entropy (ΔS) upon ligand binding are usually more important than the absolute values. A total of 175 protein−protein complexes in the last data set were used to compare the ΔS values predicted by three different methods. The absolute entropies of the ligand, receptor, and complex in one complex were calculated separately, and then the ΔS for the protein−protein association was obtained. The correlation coefficient between the ΔS values predicted by our previous method[46] and those predicted by the GPU-based K2 method is equal to ∼1.0, and the average difference is only 0.18 kcal/mol (Figure 9). Moreover, the ΔS values predicted by GPU K1 and K2 are just the same. That is to say, the three methods have the same precision in calculating the relative conformational entropies.

4. CONCLUSIONS

We have developed a GPU-based method for accelerating the calculation of the conformational entropy of molecules. This method can speed up the computing time in a very impressive way. We have tested this method using four data sets. The first set proved that the calculated entropies predicted by the new method are almost identical to those predicted by the method reported in our previous study. The tests for the second data set suggest that this method is much faster than the previous one. The third data set was used to manifest the difference between K1 and K2. K1 is quadric and K2 is linear, and K2 will be faster than K1 when the number of atoms exceeds 500. The tests for the fourth data set illustrate that the GPU-based and CPU-based methods yield almost the same predictions for the relative conformational entropies.

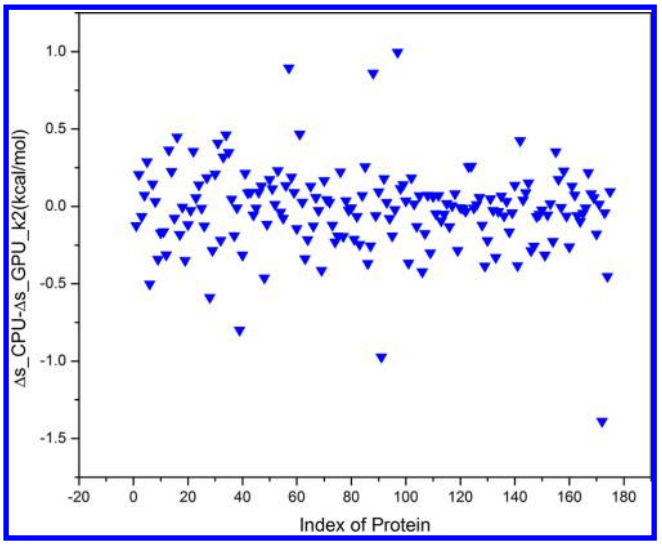


Figure 9. The difference of the ΔS values predicted by the CPU-based and GPU-based K2 methods.

It will be a very efficient way to use our method to compute the conformational entropy of biomolecules, especially for large structures, and we hope it will be helpful in calculating binding free energies and some other applications, such as protein folding and protein−protein docking, in computational biology and CADD in the future.

■ ASSOCIATED CONTENT

Supporting Information

Table S1. Comparison of the running time between CPU- and GPU-based methods generated by 5 independent runs; **Table S2**. Comparison of entropy calculation between K1 and K2. This material is available free of charge via the Internet at http://pubs.acs.org.

■ AUTHOR INFORMATION

Corresponding Author

*Phone: +34 (968) 277982. E-mail: hperez@ucam.edu (H.P.-S.). Phone: +86 (512) 65882039. E-mail: tingjunhou@hotmail.com or tingjunhou@zju.edu.cn (T.H.).

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This research was supported by the National Science Foundation of China under grant 21173156, the National Basic Research Program of China (973 program) under grant 2012CB932600, the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD), the Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) under grant 15290/PI/2010, and European Commission FEDER under grant TIN2009-14475-C04. We also thank Nvidia Corporation for hardware donation. This work was partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT belongs to CIEMAT

and the Government of Spain. The authors also thankfully acknowledge the computer resources and the technical support provided by the Plataforma Andaluza de Bioinformática of the University of Málaga.

REFERENCES

(1) Beveridge, D. L.; Dicapua, F. M. Free-energy via molecular simulation - applications to chemical and biomolecular systems. *Annu. Rev. Biophys. Biophys. Chem.* **1989**, *18*, 431−492.
(2) Aqvist, J.; Medina, C.; Samuelsson, J. E. New method for predicting binding-affinity in computer-aided drug design. *Protein Eng.* **1994**, *7*, 385−391.
(3) Kollman, P. A.; Massova, I.; Reyes, C.; Kuhn, B.; Huo, S. H.; Chong, L.; Lee, M.; Lee, T.; Duan, Y.; Wang, W.; Donini, O.; Cieplak, P.; Srinivasan, J.; Case, D. A.; Cheatham, T. E. Calculating structures and free energies of complex molecules: Combining molecular mechanics and continuum models. *Acc. Chem. Res.* **2000**, *33*, 889−897.
(4) Wang, J. M.; Hou, T. J.; Xu, X. J. Recent advances in free energy calculations with a combination of molecular mechanics and continuum models. *Curr. Comput.-Aided Drug Des.* **2006**, *2*, 287−306.
(5) Gohlke, H.; Kiel, C.; Case, D. A. Insights into protein-protein binding by binding free energy calculation and free energy decomposition for the Ras-Raf and Ras-RalGDS complexes. *J. Mol. Biol.* **2003**, *330*, 891−913.
(6) Hou, T.; Li, N.; Li, Y.; Wang, W. Characterization of domain-peptide interaction interface: prediction of SH3 domain-mediated protein-protein interaction network in yeast by generic structure-based models. *J. Proteome Res.* **2012**, *11*, 2982.
(7) Hou, T.; Wang, J.; Li, Y.; Wang, W. Assessing the performance of the MM/PBSA and MM/GBSA methods. 1. The accuracy of binding free energy calculations based on molecular dynamics simulations. *J. Chem. Inf. Model.* **2011**, *51*, 69−82.
(8) Hou, T.; Yu, R. Molecular dynamics and free energy studies on the wild-type and double mutant HIV-1 protease complexed with amprenavir and two amprenavir-related inhibitors: mechanism for binding and drug resistance. *J. Med. Chem.* **2007**, *50*, 1177−1188.
(9) Hou, T. J.; Li, Y. Y.; Wang, W. Prediction of peptides binding to the PKA RII alpha subunit using a hierarchical strategy. *Bioinformatics* **2011**, *27*, 1814−1821.
(10) Hou, T. J.; Wang, J.; Li, Y. Y.; Wang, W. Assessing the performance of the molecular mechanics/Poisson Boltzmann surface area and molecular mechanics/generalized Born surface area methods. II. The accuracy of ranking poses generated from docking. *J. Comput. Chem.* **2011**, *32*, 866−877.
(11) Huo, S.; Wang, J.; Cieplak, P.; Kollman, P. A.; Kuntz, I. D. Molecular dynamics and free energy analyses of cathepsin D-inhibitor interactions: insight into structure-based ligand design. *J. Med. Chem.* **2002**, *45*, 1412−1419.
(12) Kuhn, B.; Gerber, P.; Schulz-Gasch, T.; Stahl, M. Validation and use of the MM-PBSA approach for drug discovery. *J. Med. Chem.* **2005**, *48*, 4040−4048.
(13) Li, L.; Li, Y.; Zhang, L.; Hou, T. Theoretical studies on the susceptibility of oseltamivir against variants of 2009 A/H1N1 influenza neuraminidase. *J. Chem. Inf. Model.* **2012**, *52*, 2715−2729.
(14) Zhang, J.; Hou, T.; Wang, W.; Liu, J. S. Detecting and understanding combinatorial mutation patterns responsible for HIV drug resistance. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 1321−1326.
(15) Wang, J.; Morin, P.; Wang, W.; Kollman, P. A. Use of MM-PBSA in reproducing the binding free energies to HIV-1 RT of TIBO derivatives and predicting the binding mode to HIV-1 RT of efavirenz by docking and MM-PBSA. *J. Am. Chem. Soc.* **2001**, *123*, 5221−5230.
(16) Shrake, A.; Rupley, J. Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *J. Mol. Biol.* **1973**, *79*, 351−371.
(17) Owens, J. D.; Houston, M.; Luebke, D.; Green, S.; Stone, J. E.; Phillips, J. C. GPU computing. *Proc. IEEE* **2008**, *96*, 879−899.
(18) Buch, I.; Harvey, M. J.; Giorgino, T.; Anderson, D. P.; De Fabritiis, G. High-throughput all-atom molecular dynamics simulations using distributed computing. *J. Chem. Inf. Model.* **2010**, *50*, 397−403.
(19) Götz, A. W.; Williamson, M. J.; Xu, D.; Poole, D.; Le Grand, S.; Walker, R. C. Routine microsecond molecular dynamics simulations with amber on GPUs. 1. generalized born. *J. Chem. Theory Comput.* **2012**, *8*, 1542−1555.
(20) Haque, I. S.; Pande, V. S.; Walters, W. P. SIML: A fast SIMD algorithm for calculating LINGO chemical similarities on GPUs and CPUs. *J. Chem. Inf. Model.* **2010**, *50*, 560−564.
(21) Höfinger, S.; Acocella, A.; Pop, S. C.; Narumi, T.; Yasuoka, K.; Beu, T.; Zerbetto, F. GPU-accelerated computation of electron transfer. *J. Comput. Chem.* **2012**, *33*, 2351−2356.
(22) Höfinger, S.; Yamamoto, E.; Hirano, Y.; Zerbetto, F.; Narumi, T.; Yasuoka, K.; Yasui, M. Structural features of aquaporin 4 supporting the formation of arrays and junctions in biomembranes. *Biochim. Biophys. Acta, Biomembr.* **2012**, *1818*, 2234−2243.
(23) Jha, P. K.; Sknepnek, R.; Guerrero-Garcia, G. I.; de la Cruz, M. O. A graphics processing unit implementation of Coulomb interaction in molecular dynamics. *J. Chem. Theory Comput.* **2010**, *6*, 3058−3065.
(24) Kim, J.; Martin, R. L.; Ruebel, O.; Haranczyk, M.; Smit, B. High-throughput characterization of porous materials using graphics processing units. *J. Chem. Theory Comput.* **2012**, *8*, 1684−1693.
(25) Korb, O.; Stutzle, T.; Exner, T. E. Accelerating molecular docking calculations using graphics processing units. *J. Chem. Inf. Model.* **2011**, *51*, 865−876.
(26) Liao, Q.; Wang, J.; Watson, I. A. Accelerating two algorithms for large-scale compound selection on GPUs. *J. Chem. Inf. Model.* **2011**, *51*, 1017−1024.
(27) Liao, Q.; Wang, J.; Webster, Y.; Watson, I. A. GPU accelerated support vector machines for mining high-throughput screening data. *J. Chem. Inf. Model.* **2009**, *49*, 2718−2725.
(28) Liu, L.; Liu, X.; Gong, J.; Jiang, H.; Li, H. Accelerating all-atom normal mode analysis with graphics processing unit. *J. Chem. Theory Comput.* **2011**, *7*, 1595−1603.
(29) Miao, Y.; Merz, K. M., Jr. Acceleration of electron repulsion integral evaluation on graphics processing units via use of recurrence relations. *J. Chem. Theory Comput.* **2013**, *9*, 965−976.
(30) Narumi, T.; Yasuoka, K.; Taiji, M.; Höfinger, S. Current performance gains from utilizing the GPU or the ASIC MDGRAPE-3 within an enhanced Poisson Boltzmann approach. *J. Comput. Chem.* **2009**, *30*, 2351−2357.
(31) Narumi, T.; Yasuoka, K.; Taiji, M.; Zerbetto, F.; Höfinger, S. Fast calculation of electrostatic potentials on the GPU or the ASIC MD-GRAPE-3. *Comp. J.* **2011**, *54*, 1181−1187.
(32) Pratas, F.; Sousa, L.; Dieterich, J. M.; Mata, R. A. Computation of induced dipoles in molecular mechanics simulations using graphics processors. *J. Chem. Inf. Model.* **2012**, *52*, 1159−1166.
(33) Ruymgaart, A. P.; Elber, R. Revisiting molecular dynamics on a CPU/GPU system: Water kernel and shake parallelization. *J. Chem. Theory Comput.* **2012**, *8*, 4624−4636.
(34) Tanner, D. E.; Phillips, J. C.; Schulten, K. GPU/CPU algorithm for generalized Born/solvent-accessible surface area implicit solvent calculations. *J. Chem. Theory Comput.* **2012**, *8*, 2521−2530.
(35) Tunbridge, I.; Best, R. B.; Gain, J.; Kuttel, M. M. Simulation of coarse-grained protein-protein interactions with graphics processing units. *J. Chem. Theory Comput.* **2010**, *6*, 3588−3600.
(36) Yokota, R.; Barba, L. A.; Narumi, T.; Yasuoka, K. Petascale turbulence simulation using a highly parallel fast multipole method on GPUs. *Comput. Phys. Commun.* **2013**, *184*, 445−455.
(37) Liu, P.; Agrafiotis, D. K.; Rassokhin, D. N.; Yang, E. Accelerating chemical database searching using graphics processing units. *J. Chem. Inf. Model.* **2011**, *51*, 1807−1816.
(38) Ma, C.; Wang, L.; Xie, X.-Q. GPU accelerated chemical similarity calculation for compound library comparison. *J. Chem. Inf. Model.* **2011**, *51*, 1521−1527.
(39) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(40) Humphrey, W.; Dalke, A.; Schulten, K. VMD: visual molecular dynamics. *J. Mol. Graphics* **1996**, *14*, 33−38.

(41) *NVIDIA*. NVIDIA CUDA C Programming Guide 5.0. 2012.
(42) Cepas Quiñonero, E. J.; H., P.-S.; Cecilia, J. M.; García, J. M. MURCIA: Fast parallel solvent accessible surface area calculation on GPUs and application to drug discovery and molecular visualization. *NETTAB 2011 workshop focused on Clinical Bioinformatics* **2011**, 52−55.
(43) Hoberock, J.; Bell, N. Thrust: A parallel template library. 2010. Online at http://thrust.googlecode.com (accessed July 29, 3013).
(44) Lebedev, V. I.; Laikov, D. In *A quadrature formula for the sphere of the 131st algebraic order of accuracy*; Doklady. Mathematics, 1999; MAIK Nauka/Interperiodica: 1999; pp 477−481.
(45) Saff, E. B.; Kuijlaars, A. B. Distributing many points on a sphere. *Math. Intell.* **1997**, *19*, 5−11.
(46) Wang, J.; Hou, T. Develop and test a solvent accessible surface area-based model in conformational entropy calculations. *J. Chem. Inf. Model.* **2012**, *52*, 1199−1212.
(47) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The protein data bank. *Nucleic Acids Res.* **2000**, *28*, 235−242.
(48) Hwang, H.; Vreven, T.; Janin, J.; Weng, Z. Protein−protein docking benchmark version 4.0. *Proteins: Struct., Funct., Bioinf.* **2010**, *78*, 3111−3114.