High-Throughput Calculation of Protein−Ligand Binding Affinities: Modification and Adaptation of the MM-PBSA Protocol to Enterprise Grid Computing

Scott P. Brown* and Steven W. Muchmore

Abbott Laboratories, Bldg AP10-LL, 100 Abbott Park Rd, Abbott Park, Illinois 60064-6115

Received November 8, 2005

We have developed a system for performing computations on an enterprise grid using a freely available package for grid computing that allows us to harvest unused CPU cycles off of employee desktop computers. By modifying the traditional formulation of Molecular Mechanics with Poisson−Boltzmann Surface Area (MM-PBSA) methodology, in combination with a coarse-grain parallelized implementation suitable for deployment onto our enterprise grid, we show that it is possible to produce rapid physics-based estimates of protein−ligand binding affinities that have good correlation to experimental data. This is demonstrated by examining the correlation of our calculated binding affinities to experimental data and also by comparison to the correlation obtained from the binding-affinity calculations using traditional MM-PBSA that are reported in the literature.

INTRODUCTION

In the highly competitive environment of industrial drug discovery, timely assessment of data is crucial to decision-making processes that drive costly research and development. The extent to which computational analysis can positively impact discovery decision making is strongly dependent upon the quality of the physical model used, which is in turn limited by the computational resources available to the researcher. Increases in available computer processing power bring with it the ability to utilize more sophisticated physical models, which allow more challenging problems to be addressed that may have been too computationally expensive to consider in the past.

Perhaps the most dramatic impact on our raw ability to model the more challenging problems in biophysics is due to the adaptation of numerical algorithms to parallel computing paradigms. The increase in speed inherent to parallel computing architectures is driven by the distribution of workload among a group of processors. In order for a numerical problem to take advantage of the speed increase of parallelization, it must first be broken down into smaller (less costly) tasks, which can be distributed across the processors used in the calculation.

There are a number of established parallelization strategies, and the particular strategy employed depends on the type of problem being addressed.[1] On a fundamental level, the parallelization of any algorithm requires the removal of internal dependencies. Problems that can be broken down into fully independent tasks (i.e., coarse-grain parallelized) are by far the most straightforward to implement. With coarse-grain parallelization, the calculations running on separate processors do not need to communicate with each other but, rather, communicate with a "master" process, which coordinates all activities for the overall progress of the calculation. Problems that cannot be easily broken down into independent tasks are more challenging to parallelize, due to the fact that the solution in one part of the calculation depends on the outcome of the other parts. This is seen in applications such as many-body molecular dynamics (MD) simulations[2] and optimization problems such as in crystallographic refinement.[3]

For molecular simulations (the problem of interest here), the most prevalent parallelization strategy is one in which the simulation is replicated in its entirety across all processors in the calculation.[4,5] The atom positions in the simulation are then propagated in time through the concerted calculation of all processors. This design requires substantial interprocess communication, a fact that is at the heart of the weakness of this type of approach: poor parallel efficiency due to heavy demands for interprocess communication. On high-end massively parallel supercomputers, molecular simulations can be scaled up to thousands of processors,[6] due to the great lengths that designers have gone to in order to enable fast interprocess communication. But on more typical (and more accessible) parallel computer clusters, the scalability can only be extended to tens of processors, unless other more sophisticated forms of parallelization are employed.[7]

For computational problems that can be coarse-grain parallelized, a new emerging resource is found with the paradigm of "grid computing", the meaning of which we take here to be the numerical computation performed on a noncentralized, heterogeneous distribution of networked computers (the "grid"). Examples of high-profile distributed grid computing projects are SETI@Home[8] and Folding@Home,[9] which have accrued amazing numbers of CPU years on computers around the world in a relatively short period of time. A recent example of the utilization of a corporate enterprise grid is seen with the high-throughput docking results of Vangrevelinghe et al.[10]

The basic idea behind distributed grid computing is that individuals who choose to enroll their personal computers onto the grid can do so via installation of software onto their machine. The purpose of the software is to manage the

* Corresponding author phone: (847) 937-1996; fax: (847) 938-2478; e-mail: scott.brown@abbott.com.

10.1021/ci050488t CCC: $33.50
Published on Web 02/02/2006

computer by continuously monitoring its state, and then at the appropriate time, it requests that jobs be sent to the computer from a server. In this way, CPU cycles can be extracted from machines during times when the computer would otherwise sit idle. The types of machines that can be enrolled in grids may span a wide range of desktop platforms, making it highly cross-platform compatible.

In a corporate setting, where there is a homogeneous distribution of employee desktop computers that have consistent, recurring idle times, the concept of an enterprise grid is very attractive as it represents a more efficient use of company resources. However, one key aspect to success is that the software must run unobtrusively on the individuals' computers. The advantages of grid computing would evaporate quickly were it to systematically disrupt worker activities. A more subtle advantage to grid computing is that, unlike dedicated compute clusters, a typical grid will have its hardware continually refreshed, as occurs when individuals (or corporations) replace or upgrade their desktop computers on a periodic basis. In our corporate environment, this turnover amounts to an annual refresh rate of about 20% of the total grid.

To illustrate the power of enterprise grid computing, we describe here the modification and grid adaptation of an algorithm for estimating free energies of protein−ligand binding. The ability to calculate protein−ligand binding free energies across a set of diverse molecules in a proprietary compound library is highly desirable in a drug discovery setting, as it holds the potential to systematically accelerate the process of finding and developing novel therapeutic compounds. Methodologies capable of producing accurate protein−ligand binding free energies suffer from a classic tradeoff: with the increased accuracy of a more powerful method comes a prohibitively high computational cost. The substantial computational barriers to obtaining accurate free energies have limited their application in drug discovery.

However, by harvesting CPU time during idle cycles on employee desktop computers, we demonstrate that it is feasible to obtain meaningful relative free-energy differences that can be used to rank-order the binding affinities of a set of compounds for a given protein target. We employ a modified, that is, streamlined, version of the method developed by Dave Case and the late Peter Kollman,[11,12] which has been termed Molecular Mechanics with Poisson−Boltzmann Surface Area (MM-PBSA).

With our modifications to MM-PBSA, we are able to readily distribute free-energy calculations onto an enterprise grid, built using the freely available grid-computing package Condor.[13] We validate our modifications to MM-PBSA by reproducing the work of Kuhn and Kollman,[14] in which they report the calculation of protein−ligand binding affinities for avidin and a set of biotin derivatives using their (highly computationally demanding) version of MM-PBSA. For an investigation into the applicability of alternate forms of MM-PBSA, the reader is directed to the recent work of Stahl et al.[15]

COMPUTATIONAL METHODS

A detailed description of our MM-PBSA methodology will be given elsewhere.[16] As the emphasis of the current work is on the use of high-performance computing, we present here only a brief summary of the MM-PBSA method, followed by a more detailed description of the design and implementation onto the enterprise grid environment.

MM-PBSA. The net change in binding free energy accompanying the formation of the protein−ligand complex is approximated by the following equation

$$\langle \Delta G_{\mathrm{MM-PBSA}} \rangle = \langle \Delta E_{\mathrm{MM}} \rangle + \langle \Delta G_{\mathrm{PBSA}} \rangle - T \langle \Delta S_{\mathrm{solute}} \rangle \quad (1)$$

where the brackets $\langle ... \rangle$ represent the ensemble averages of the enclosed quantities, ΔE_{MM} is the change in molecular mechanics (MM) energy, ΔG_{PBSA} is the net change in solvation free energy upon binding, T is the absolute temperature of the system, and $\Delta S_{\mathrm{solute}}$ is the internal entropy change of the solute upon binding.

E_{MM} is calculated from the particular MM force field used in the MD simulation, evaluated in the gas phase without any distance cutoff on nonbonded interactions. The G_{PBSA} term is given by the equation $\Delta G_{\mathrm{PBSA}} = \Delta E_{\mathrm{PB}} + \Delta G_{\mathrm{SA}}$, where E_{PB} is obtained by solution of the Poisson−Boltzmann (PB) equation, for which we use the ZAP module from OpenEye.[17] ΔG_{SA} is the free energy for surface formation of a hydrophobic cavity in water,[18,19] given by $\Delta G_{\mathrm{SA}} = \gamma \Delta A$, where $\Delta A = A^{\mathrm{complex}} - A^{\mathrm{protein}} - A^{\mathrm{ligand}}$ and $\gamma = 0.025$ kcal/(mol Å^2). Finally, in our calculations, $\Delta S_{\mathrm{solute}}$ is neglected.

The main procedural modification we have made to Kollman's MM-PBSA formulation is to shorten the length of the MD trajectories. This shortening is made possible by using an implicit solvation model to capture the nontrivial effects of aqueous solvation during the MD runs. We perform MD using SANDER from the AMBER 8.0 simulation package.[20] Hydrogen bonds are constrained using SHAKE, with a time step of 2 fs and a distance cutoff of 12.0 Å on the nonbonded interactions. The implicit solvation model we use is the recent generalized Born model of Onufriev et al.[21] Force-field parameters and partial charges for the proteins come from the *ff*03 force field.[22] The small-molecule ligands are assigned AM1-BCC[23] partial charges, and the generalized AMBER force field (GAFF) of Wang et al.[24] is used for force-field parametrization. Both the AM1-BCC partial charge calculations and GAFF parameter assignments are performed as an automated procedure, which is, in theory, capable of processing large numbers of small molecules with little to no user intervention.

Starting from the crystal-structure coordinate file of the avidin−biotin complex (PDB ID: 1AVD), we preprocess to remove all nonligand and nonpeptide atoms. The starting coordinates for the additional six biotin-derived compounds are prepared by modifying the original biotin molecule in the avidin active site. The final set of biotin-related compounds is designated sequentially with the names BTN1 through BTN7 (see Figure 1). All initial structures are steepest-decent minimized for 500 steps, followed by an equilibration, during which the system is heated from 0 to 300 K over 6 ps using constant-temperature Langevin dynamics. The final structure from equilibration is then input as the initial starting configuration into the MD production runs. We perform the production runs over 13 ps, discarding all data from the first 3 ps. Over the course of the final 10 ps, we save to disk a series of 100 coordinate "snapshots" of the protein−ligand complex, taken at evenly spaced intervals in time along the trajectory.

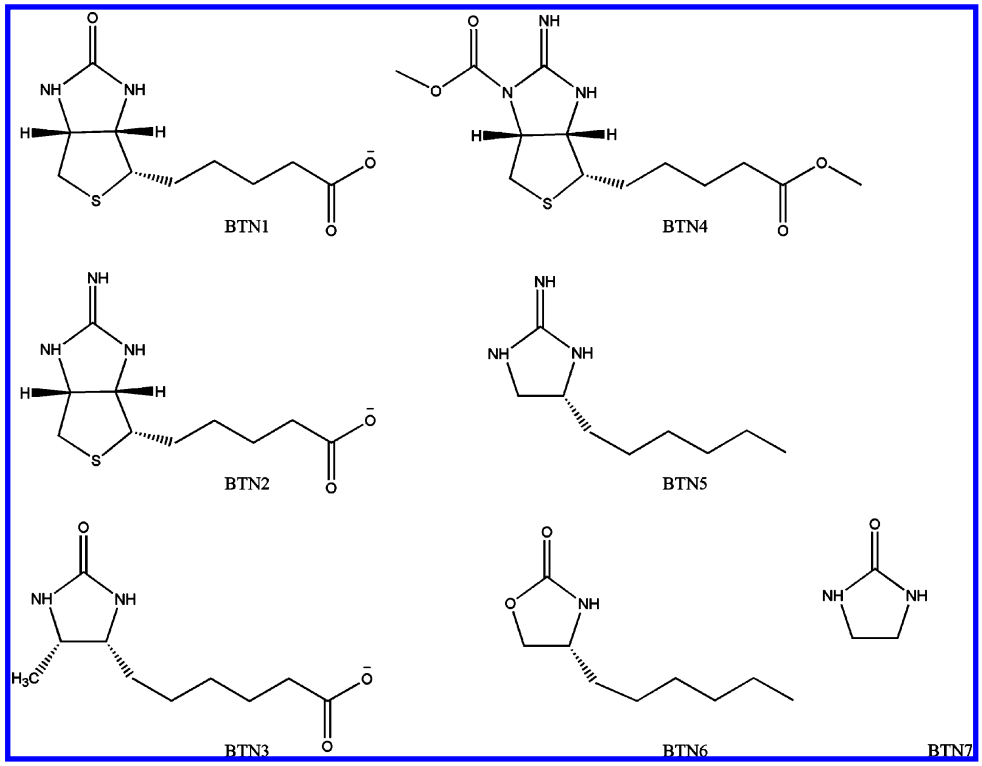

Figure 1. Chemical identities of the seven biotin-related compounds analyzed by MM-PBSA to calculate their binding affinities to avidin.

Grid Configuration. Our MM-PBSA calculations are particularly well-suited to deployment onto a coarse-grained parallel architecture, as the analysis is comprised of many independent jobs, each of which requires relatively short-duration runs of floating-point-intensive calculations. In fact, all results reported herein were obtained from calculations performed on an enterprise grid, from which we utilize otherwise wasted CPU cycles on employee desktop personal computers. Our enterprise grid is built with the freely distributed package Condor,[13] available from the Computer Science department at the University of Wisconsin. This has made for a very economical, and yet very powerful, resource for performing computer-intensive calculations. With our implementation of MM-PBSA on an enterprise grid computing environment, we are able to process data on a scale that would otherwise be inaccessible to us, even with the more efficient version of MM-PBSA employed.

The schematic design of our Condor grid is depicted in Figure 2. The "central manager" machine is responsible for monitoring the status of all members on the grid. The actual number-crunching work on the grid is performed by the "execute nodes", and the central manager tracks the status of these nodes by intermittent polling of each machine. This allows the central manager to assign tasks to those computers that are currently available to run jobs and allows the central manager to allocate and schedule grid-resource usage.

New jobs enter the grid from a separate machine called the "submit node". The submit node takes care of all the housekeeping necessary for the execution of individual jobs. Upon the arrival of new jobs into the queue (located on the submit node), the job requirements are assembled and published in the form of a "classified ad" to the central manager, at which point the central manager attempts to pair each job with an available execute node. Once this "match making" process is complete, the central manager returns the matches to the submit node. The submit node then contacts each matched execute node and establishes the

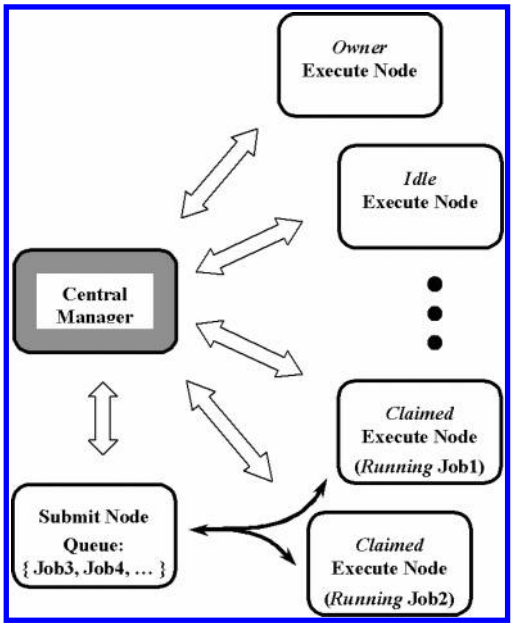

Figure 2. Schematic representation of our pilot-study Condor grid, which is composed of a central manager, submit node, and execute nodes (the computers that perform the actual numerical processing). The central manager is a dual-processor Intel Pentium III machine with a CPU clock at 1.3 GHz and 2 GB of RAM. The submit node is an Intel Xeon machine with a CPU clock at 2.4 GHz and 1 GB of RAM. For information on the execute nodes, see the Table 1 and also the relevant discussion in the text. An execute node is designated *Claimed* if it is currently in the process of executing a job. *Owner* indicates that either a user is interactively operating the computer or the current load on the execute node is above a predefined threshold. *Idle* indicates that the execute node is available to process a job. To become *Idle*, an execute node must have no interactive use of the keyboard and mouse for a continuous 15 min period.

Table 1. Snapshot of Composition and Current Activity of the Grid during Condor Pilot Study[a]

	machines	claimed	owner
Linux	26	19	7
Windows	86	50	36
total	112	69	43

[a] A machine listed as "claimed" is currently in the process of executing a job. A machine is designated "owner" if a user is interactively operating the computer, or if the current load on the computer is above a defined threshold.

communication necessary to negotiate job execution. Typically, the submit node has more robust hardware requirements than the central manager, due to the central manager tending to experience significantly lower overhead on average.

The execute nodes on the grid are comprised of a variety of Intel-based machines, running either Linux or Windows operating systems. A typical workday snapshot of the grid during our pilot study of Condor is shown in Table 1. The specific make up of the grid at any given time fluctuates as users shut down their computers or otherwise render them invisible to the Condor central manager. For our pilot grid, these fluctuations were small and amounted to changes of approximately two to five computers at any given time. We chose this particular grid size for the pilot study because of its manageability, in that it does not require any specialized hardware for either the central manager or the submit node. We found that we consistently obtained about 70% of the total CPU cycles, which amounts to approximately 35 CPU days/wallclock day.

When an execute node has no keyboard and mouse activity continuously for 15 min and no significant system load, it advertises itself to the central manager as being available to run a job. Once the central manager assigns a job to the execute node and the necessary files have been transferred, job execution begins and will run to completion unless interrupted by the user. The execution of any job becomes suspended upon user activity on either the mouse or keyboard, but the job remains resident in memory on the execute node. If the job resides in this interrupted state for a continuous 30 min, it will be "vacated" from the execute node, meaning execution is halted and all files are discarded. The job then goes back into the queue to be run on a different execute node. Because of this, the majority of jobs get completed during evening and nighttime hours, when employee desktop machines typically have the longest uninterrupted idle times.

Precompiled executables are needed for each particular platform on the grid, which, along with all the initial data files, get transferred to the execute node at runtime. On our grid, two versions of all of the executables reside on the submit machine; one version is for WindowsNT5.x and the other for RedHat Enterprise Linux 3.2. On Linux, it is straightforward to compile an executable that can be distributed to any of the Linux platforms on the grid, provided that all Linux machines have compatible kernel versions and system libraries. We use version 8.1 of the Intel Fortran Compiler[25] to statically link and compile Fortran95 code on Linux, and for all other source code, we use the GNU Compiler Collection that comes standard with the RedHat Linux distribution.

To create executables for Windows, the situation is a bit more involved. In our corporate environment, we desire a grid configuration for the Windows machines that does not require any additional installation of software (beyond what is required to initially install the Condor package). For the execution of Condor jobs on Windows machines, we seek to create stand-alone executables. To accomplish this, we compile Windows executables under Cygwin[26] using version 9.0 of the Absoft Fortran Compiler[27] to compile Fortran95 source code and the GNU Compiler Collection distributed with Cygwin 1.5.10 for the compilation of all other source code. To execute the Cygwin-compiled binaries under Windows, a dynamically linked library from the Cygwin distribution (cygwin1.dll) must be present in the same directory as the Windows executable. Thus, we transfer the library file along with the executable file onto the Windows execute machines. With this construction, we only need to install the Condor package on each Windows machine in the grid, which, from a practical standpoint, makes for an environment that is substantially easier to support.

On average, we observe that jobs on the Windows execute nodes tend to take noticeably longer to complete than do jobs on the Linux nodes. This is most likely due to a number of factors. Typically, our Linux machines have more RAM available at any given time. While the majority of Windows machines have 512 MB of system RAM (some machines have as little as 256 MB and others as much as 1 GB), the Linux boxes all have at least 1 GB of RAM; however, as the memory footprints of our MD jobs are around 10−20 MB, memory is not likely to be the differentiating factor in performance comparisons between the two platforms. More likely, the major contributor to the observed difference between Windows and Linux platforms is due to the extra overhead associated with running an executable on Windows within the emulation environment provided by the Cygwin dynamically linked library.

For the submission of a set of MM-PBSA jobs onto the grid, we break the calculation down into a series of sequential steps. The multistep execution of a job is then managed using a Condor feature called directed acyclic graph (DAG), which allows a series of jobs to be constrained to execute in a certain order (e.g., we require that the MM part of the calculation occur prior to the PBSA part). The first step of the DAG distributes the minimization, equilibration, and production MD runs (i.e., the MM part of MM-PBSA) in the proper sequence onto the grid. Once the MD runs have completed, and the resulting trajectory files are moved back to the submit node, the DAG then shuttles out a new round of jobs to the grid for processing the MD trajectories. During this part of the analysis (the PBSA part of MM-PBSA), the trajectories are split up into a series of files, with each file containing an individual snapshot from MD. PB energies and surface areas are then calculated for each file. At the conclusion of this step, when all jobs have returned their results, the final free-energy estimates are assembled using eq 1.

With this scheme, the average CPU time required for calculating the binding free energy of a single protein−ligand complex is around 100−200 min. While this figure is system-size dependent, it is sufficiently low such that reasonable throughput can be realized, even for larger systems. For

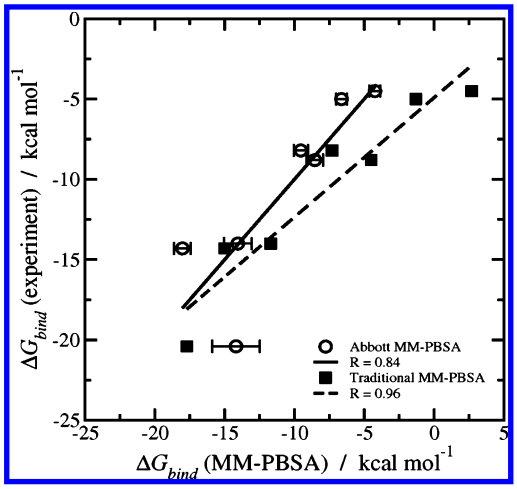

Figure 3. Calculated binding free energies ΔG_{bind}(MM-PBSA) versus experimentally measured binding free energies ΔG_{bind}(experiment) for seven biotin derivatives binding to avidin. The lines show linear fits to the data, with the corresponding correlation coefficients indicated in the legend. Note that for visualization purposes we have rescaled our data in order to facilitate a direct comparison. The experimental free energies are those reported by Kuhn and Kollman.[14]

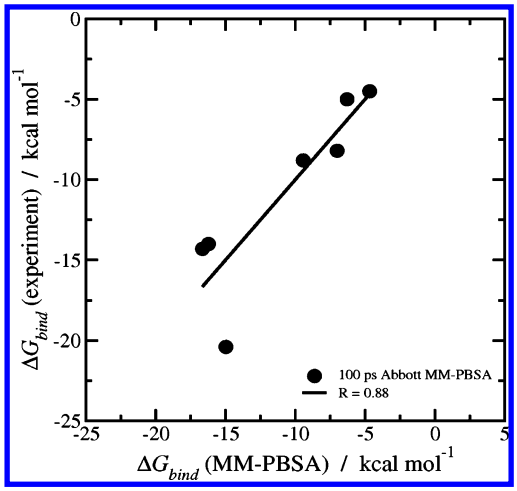

Figure 4. Calculated binding free energies ΔG_{bind}(MM-PBSA) versus experimentally measured binding free energies ΔG_{bind}(experiment) for seven biotin derivatives binding to avidin. For these data, the MD runs were extended to 100 ps. The additional 90 ps of dynamics produces a modest improvement in correlation, with the correlation coefficient going from 0.84 to 0.88, and a slight reduction in the spread of the data about the line.

example, on our pilot grid, we are able to achieve throughputs of roughly 110 structures per wallclock day.

RESULTS AND DISCUSSION

Shown in Figure 3 is the result of the grid free-energy calculations, in which we compare our calculated binding free energies against those predicted by Kuhn and Kollman.[14] We find that we are able to produce a significant correlation to experimental values, with correlations that are comparable to what is obtained by Kuhn and Kollman (designated "Traditional MM-PBSA" in Figure 3).

Clearly, the results of Kuhn and Kollman are superior, both in terms of correlation and the deviation, or spread, of the data about the linear fit. Yet, their full analysis required approximately 80 CPU days to complete. By comparison, we were able to obtain our results in approximately 120 min. This is an increase in speed of around 3 orders of magnitude; however, some care should be taken in making a direct comparison, due to differences in computer hardware used in the calculations. Nevertheless, we are able to achieve a dramatic improvement in overall throughput for the calculation.

For all but one of our calculated free energies in Figure 3, we find that the values are more negative than those calculated by Kuhn and Kollman. The exception is the lone data point at ΔG_{bind}(experiment) ≈ -20 kcal/mol, which is significantly more positive. Interestingly, this data point is BTN1 (see Figure 1), for which the initial coordinates input into the calculation are the unmodified crystallographically determined coordinates (subsequently used to derive the initial coordinates for all other compounds). A quantitative comparison of our calculated free energies to those of Kuhn and Kollman is perhaps a bit artificial due to the fact that we must rescale our values to place them on the same axes; however, qualitatively, there does appear to be a systematic bias toward overestimating all but the lone BTN1 data point. It should be noted here that the rescaling of our data is a linear transformation that does not affect the correlation of the data in any way. In a certain sense, this rescaling can be thought of as a form of solute entropy correction to our raw free-energy values, the raw values of which typically fall in the range of -50 to -150 kcal/mol. The incorporation of entropy calculations using crude normal-mode estimates of solute entropic change is sufficient to shift the calculated values into a similar range to that of the experimental values; however, we do not include the computationally expensive normal-mode calculation in our method as it does not appear to be necessary on the basis of the correlation we observe.

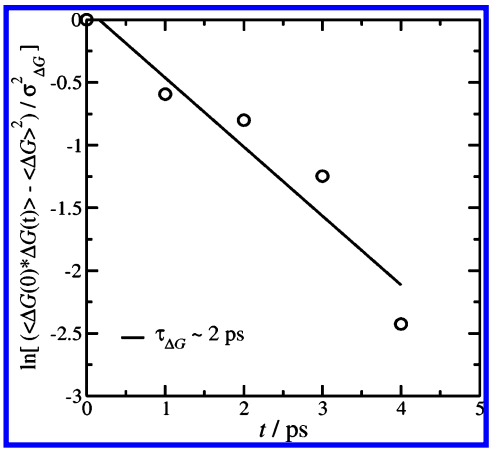

Figure 5. Decorrelation of calculated MM-PBSA free energies $\ln[(\langle \Delta G(0)\, \Delta G(t)\rangle - \langle \Delta G\rangle^2)/\sigma_{\Delta G}^2]$ as a function of MD simulation time t, where $\sigma_{\Delta G}^2$ is a scale factor given by $\sigma_{\Delta G}^2 = \langle \Delta G^2\rangle - \langle \Delta G\rangle^2$. The rough time constant for decay is obtained by linear fit to the data (indicated by the line), with its value indicated in the legend. The data for this plot are taken from the 100 ps MD trajectories.

To investigate possible effects due to the short time scale of our MD, we performed the MM-PBSA calculations using significantly longer MD runs. Shown in Figure 4 are MM-PBSA calculations in which the MD runs have been extended to a length of 100 ps, a factor of 10 longer than the original runs. We observe a modest improvement in correlation with the longer runs, as well as a slight reduction in the spread of the values about the linear fit; however, for our purposes,

we find that a gain of 0.04 in correlation does not justify the 10-fold increase in calculation time.

Finally, it is worthwhile to look at the rate of convergence of the calculated free-energy values, which we can do in a quantitative manner by investigating the decorrelation of the free energies due to intrinsic system fluctuations. Shown in Figure 5 is the decay in the correlation of calculated free-energy values as a function of time for the 100 ps MD trajectories. Fit to the data yields a characteristic time constant for relaxation of about 2 ps. This rate of decorrelation indicates that a 10 ps trajectory spans roughly five relaxation times, $5\tau_{\Delta G}$, which suggests that 10 ps provides an adequate sampling duration for the complexes studied here.

CONCLUSIONS

We have constructed an enterprise grid with the software package Condor, which is freely available from the University of Wisconsin, Computer Science department. By installing Condor on employee desktop computers running both Windows and Linux operating systems, we perform calculations using CPU cycles on computers that would otherwise sit idle. This provides a more efficient use of company resources, while creating a substantial resource for computational analysis.

Using a modified version of MM-PBSA, we show that physics-based free-energy calculations can be adapted and deployed onto an enterprise-grid computing environment to obtain free-energy estimates having a significant degree of correlation to experimental values. Our streamlined MM-PBSA method allows us to produce protein−ligand free-energy estimates within approximately 100−200 CPU minutes. For our modest-sized grid, this translates into a throughput of approximately 110 structures per wallclock day. For moderately sized enterprise grids, throughputs on the order of thousands of structures per day could be realized.

In the original work of Kuhn and Kollman, the full calculation of the protein−ligand binding free energies for avidin and the seven biotin-related compounds required around 80 CPU days to achieve a correlation coefficient to experimental results of 0.96. We perform the same calculation in about 120 min and obtain a correlation coefficient of 0.84. As stated previously, some care should be taken in making this direct comparison, due to differences in the hardware used in the calculations.

It is worth emphasizing that our method employs a fully automated procedure for parametrizing the small-molecule ligands (both partial-charge assignments and force-field designations). This is significant in that we wish to apply this method systematically to large compound libraries, for which parametrization by hand would be a very tedious, if not untenable, process.

Finally, we find that (at least for the limited data set studied here) it is not necessary to explicitly include solute entropic effects into the calculation if one is interested primarily in rank ordering compounds on the basis of binding affinity for a particular target protein. Naturally, free-energy estimates produced in this way can only be valid for the rank comparison of compounds against a particular target. If an absolute free energy is required, then a full treatment of the entropic change upon binding is needed.

In continuance of this work, we plan to perform a MM-PBSA analysis on a significant fraction of the Abbott internal data, the sum total of which represents around 1600 crystallographic structures over roughly a dozen targets. Additionally, we plan to investigate the incorporation of explicit solute entropy into our calculations.

REFERENCES AND NOTES

(1) Scott, L. R.; Clark, T.; Bagheri, B. In *Scientific Parallel Computing*; Princeton University Press: Princeton, New Jersey, 2005.

(2) Allen, M. P.; Tildesley, D. J. In *Computer Simulation of Liquids*; Oxford University Press: New York, 1987.

(3) Bourne, P. E.; Hendrickson, W. A. A CPU benchmark for protein crystallographic refinement. *Comput. Biol. Med.* **1990**, *20* (4), 219−30.

(4) Fang, Z.; Haymet, A. D. J.; Shinoda, W.; Okazaki, S. Parallel molecular dynamics simulation: Implementation of PVM for a lipid membrane. *Comput. Phys. Commun.* **1999**, *116* (2−3), 295−310.

(5) Lin, S. L.; Mellor-Crummey, J.; Pettitt, B. M.; Phillips, G. N. Molecular Dynamics on a Distributed Memory Multiprocessor. *J. Comput. Chem.* **1992**, *13*, 1022−1035.

(6) Duan, Y.; Kollman, P. A. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science* **1998**, *282* (5389), 740−4.

(7) Plimpton, S.; Hendrickson, B. A new parallel method for molecular dynamics simulation of macromolecular systems. *J. Comput. Chem.* **1996**, *17* (3), 326−337.

(8) NET NEWS: ...and a Search for Alien Life. *Science* **1998**, *282* (5390), 839b.

(9) Pande, V. S.; Baker, I.; Chapman, J.; Elmer, S. P.; Khaliq, S.; Larson, S. M.; Rhee, Y. M.; Shirts, M. R.; Snow, C. D.; Sorin, E. J.; Zagrovic, B. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers* **2003**, *68* (1), 91−109.

(10) Vangrevelinghe, E.; Zimmermann, K.; Schoepfer, J.; Portmann, R.; Fabbro, D.; Furet, P. Discovery of a potent and selective protein kinase CK2 inhibitor by high-throughput docking. *J. Med. Chem.* **2003**, *46* (13), 2656−62.

(11) Srinivasan, J.; Thomas, E.; Cheatham, I.; Cieplak, P.; Kollman, P. A.; Case, D. A. Continuum Solvent Studies of the Stability of DNA, RNA, and Phosphoramidate-DNA Helices. *J. Am. Chem. Soc.* **1998**, *120* (37), 9401−9409.

(12) Kollman, P. A.; Massova, I.; Reyes, C.; Kuhn, B.; Huo, S.; Chong, L.; Lee, M.; Lee, T.; Duan, Y.; Wang, W.; Donini, O.; Cieplak, P.; Srinivasan, J.; Case, D. A.; Cheatham, T. E., III. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. *Acc. Chem. Res.* **2000**, *33* (12), 889−97.

(13) Thain, D.; Tannenbaum, T.; Livny, M. In *Condor and the Grid*; John Wiley & Sons Inc.: New York, 2003; p 1080.

(14) Kuhn, B.; Kollman, P. A. Binding of a diverse set of ligands to avidin and streptavidin: an accurate quantitative prediction of their relative affinities by a combination of molecular mechanics and continuum solvent models. *J. Med. Chem.* **2000**, *43* (20), 3786−91.

(15) 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* (12), 4040−8.

(16) Brown, S. P.; Muchmore, S. W. Calculation of protein-drug binding affinities using a modified version of MM-PBSA. In preparation.

(17) OpenEye Scientific Software, Santa Fe, New Mexico. http://www.eyesopen.com/ (accessed Jan 2006).

(18) Reynolds, J. A.; Gilbert, D. B.; Tanford, C. Empirical Correlation Between Hydrophobic Free Energy and Aqueous Cavity Surface Area. *Proc. Natl. Acad. Sci. U.S.A.* **1974**, *71* (8), 2925−2927.

(19) Sitkoff, D.; Sharp, K. A.; Honig, B. Correlating solvation free energies and surface tensions of hydrocarbon solutes. *Biophys. Chem.* **1994**, *51* (2−3), 397−409.

(20) Pearlman, D. A.; Case, D. A.; Caldwell, J. W.; Ross, W. S.; Cheatham, I.; DeBolt, S.; Ferguson, D.; Seibel, G.; Kollman, P. AMBER, a package of computer programs for applying molecular mechanics, normal-mode analysis, molecular dynamics and free energy calculations to simulate the structureal and energetic properties of molecules. *Comput. Phys. Commun.* **1995**, *91*, 1−41.

(21) Onufriev, A.; Bashford, D.; Case, D. A. Exploring protein native states and large-scale conformational changes with a modified generalized Born model. *Proteins: Struct., Funct., Bioinf.* **2004**, *55* (2), 383−94.

(22) Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M. C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; Caldwell, J.; Wang, J.; Kollman, P. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *J. Comput. Chem.* **2003**, *24* (16), 1999−2012.
(23) Jakalian, A.; Bush, B. L.; Jack, D. B.; Bayly, C. I. Fast, Efficient Generation of High-Quality Atomic Charges. AM1-BCC Model I: Method. *J. Comput. Chem.* **2000**, *21* (2), 132−146.
(24) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and testing of a general amber force field. *J. Comput. Chem.* **2004**, *25* (9), 1157−74.
(25) Intel Corporation, Santa Clara, California. http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/index.htm (accessed Jan 2006).
(26) Cygwin is a Linux-like environment for the Microsoft Windows operating system distributed under the GNU Public License. http://cygwin.com/ (accessed Jan 2006).
(27) Absoft Corporation, Rochester Hills, Michigan. http://www.absoft.com/Products/Compilers/Fortran/Windows/windows.html (accessed Jan 2006).

CI050488T