A Resource-Efficient Computing Paradigm for Computational Protein Modeling Applications

Yaohang Li
Department of Computer Science
North Carolina A&T State University
yaohang@ncat.edu

Douglas Wardell
Department of Computer Science
North Carolina A&T State University
dwardell@ncat.edu

Vincent Freeh
Department of Computer Science
North Carolina State University
vin@csc.ncsu.edu

Abstract

Many computational protein modeling applications using numerical methods such as Molecular Dynamics (MD), Monte Carlo (MC), or Genetic Algorithms (GA) require a large number of energy estimations of the protein molecular system. A typical energy function describing the protein energy is a combination of a number of terms characterizing various interactions within the protein molecule as well as the protein-solvent interactions. Evaluating the energy function of a relatively large protein molecule is rather computationally costly and usually occupies the major computation time in the protein simulation process.

In this paper, we present a resource-efficient computing paradigm based on "consolidation" to reduce the computational time of evaluating the energy function of large protein molecule. The fundamental idea of consolidation is to increase computational density to a computer in order to increase the CPU utilizations. Consolidation will be particularly efficient when the consolidated computations have heterogeneous resource demands. In computational protein modeling applications with costly energy function evaluation, we advocate the use of "thread consolidation," which is to spawn concurrent threads to carry out parallel energy function terms computations. Our computational results show that 7%~11% speedup in a protein loop structure prediction program on various hardware architectures where memory-intensive and computation-intensive terms coexist in the energy function. For an MD protein simulation program where computation-intensive energy function evaluations are divided and carried out by concurrent threads, we also find slight performance improvement when the thread consolidation technique is applied.

1. Introduction

Molecular Dynamics (MD) [5, 6, 8], Monte Carlo (MC) [16, 27, 28], and Genetic Algorithm (GA) [29] are popular simulation approaches to computationally model protein or protein complex structures. These approaches share a similar form of minimizing the energy function of protein or protein complex to obtain a stable conformation. During the simulation process, thousands to millions of time steps or iterations need to be carried out before convergence or system equilibrium is reached. At every time step or iteration step, one or more new conformations are generated and their energies must be computed. Protein energy functions are crucial elements in many protein modeling applications and evaluation of a protein energy function usually demands large amount of computation time. For a fine-grain energy function where each atom in the protein structure is treated explicitly, the computational demands of estimating the protein energy grow as the square of the number of atoms, e.g., tens of thousands of atom-atom interactions need to be calculated for evaluating the energy of a small protein with only 50 residues. Even when simplified representation of the atomic structure of the protein is applied, the energy function evaluation can still pose significant computational cost. In many the protein or protein complex structure modeling applications, the energy function evaluation of protein molecule occupies the majority of the computation time.

The typical computational time for the protein structure modeling applications ranges from several hours to several days or even longer. In practice, the protein modeling applications may be required to be

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

carried out on many protein targets. For example, in system biology study, the structure models and interactions on thousands of bacterial proteins are needed to understand the function of a biological system. Assuming that the average protein modeling computation time is around 2 hours, systematic study of a biological system with ~5,000 proteins can easily reach 10,000 hours of computation time. Such heavy computations are usually carried out on a large-scale computation platform such as a terascale high-performance supercomputer cluster or a computational grid with large number of distributed computational resources [1]. The computation time reduction of a protein modeling program, even only a small fraction, can free up considerable large amount of computational resources.

The most common approach to reduce the computation time of the protein modeling applications is to take advantage of the power of parallel computing. By scattering the computation of different components in an energy function to parallel processors, the energy function evaluation time can be reduced and so does the overall application computation time. However, this approach demands significantly more computational resources, which is not suitable in system biology computation when modeling a lot of proteins or protein complexes are required. For large number of protein simulations, the popular approach is to execute the protein modeling application in an embarrassingly parallel mode on large-scale computational platforms, such as folding@home [2], rosetta@home [3], and predictor@home [4].

Our analysis of the protein energy functions shows that many energy functions involve terms demanding heterogeneous computational resources. Computation of such an energy function in a naïve serial mode is not very resource-efficient with relatively low CPU utilization rate. In this paper, we apply the consolidation technique to increase the computational resource utilization rate so as to reduce the energy function computation time and thus speed up the protein modeling program. Our "thread consolidation" approach is to spawn concurrent threads to carry out parallel energy function term evaluations to increase the computation density assigned to a processor. We verify the thread consolidation approach by applying it to a protein loop backbone structure prediction program based on GA with energy function including memory-intensive and computation-intensive components as well as an MD program with computation-intensive system energy computation. Our computational results show that applying thread consolidation to protein energy function evaluation can lead to certain performance improvement in these applications.

The remainder of the paper is organized as follows. Section 2 analyzes different categories of the protein energy functions and their computational characteristics. Section 3 illustrates the consolidation technology in protein energy function evaluation. The computational results of consolidation in a GA program for protein loop structure prediction and an MD program for protein-solvent interactions are shown in Section 4, respectively. Finally, Section 5 summarizes our conclusions and future research directions.

2. Energy Functions in Protein Modeling Applications

In protein structure modeling, energy functions are used to evaluate the feasibility of a particular structure of a protein molecule. Protein modeling energy functions can be categorized into three major categories – physics-based energy functions, knowledge-based energy (scoring) functions, and hybrid energy functions.

2.1. Physics-based Energy Functions

The physics-based energy functions are developed from general knowledge of the laws of physics, which have the following generic form:

$$E(\mathbf{R}) = \sum_{bonds} B(\mathbf{R}) + \sum_{bond\ angles} A(\mathbf{R}) + \sum_{torsions} T(\mathbf{R}) + \sum_{non-bonds} N(\mathbf{R}),$$

where $\mathbf{R}$ is the conformation vector of the protein molecule. $B(\mathbf{R})$ is the bond energy term corresponding to the stretching and compression of the bond length. $A(\mathbf{R})$ is the bond angle energy term corresponding to changes in the angle between bonds. $T(\mathbf{R})$ is the torsional energy term with respect to torsion angles in series of three bonds. The non-bonds term $N(\mathbf{R})$ usually includes Van der Waals interactions, steric clash, and electrostatic interaction. In addition, terms such as hydrogen bonding, hydrophobic, salvation and cross, are often used in various energy functions. The physics-based energy function examples include CHARMM [5], AMBER [6], OPLS [7], GROMOS [8], etc.

The physics-based energy functions yield a rugged, funnel-like energy landscape, which is difficult to search in the protein modeling. From computation point of view, the physics-based energy functions are mainly computation-intensive where a lot of floating-point operations are involved. Very often, to avoid heavy run-time computation, some terms in the physics-based energy function are pre-calculated and

stored into a table – as a result, the physics-based energy function evaluation may also involve significant memory access and may be memory-intensive.

2.2 Knowledge-based Energy Functions

Knowledge-based approaches evaluate the increasing number of experimentally determined conformations by statistical approaches to extract rules on preferred configurations and combinations. These rules are converted into "pseudo-potential" energy (scoring) functions for protein modeling. There exist numerous knowledge-based energy functions based on various aspects or data sets. For example, distance-based energy function [9] provides the expect value of pair-wise distance of atoms in neighboring residues; residue triplet scoring function [10] involves the radii of curvatures formed among triplets of residues; DrugScore [11] describes the binding geometry of ligands in proteins; contact energy function characterizes the atom-atom and atom-solvent contact surfaces [12]; shape complementary scoring function measures the protein-protein binding interface [13]; and many others.

Compared to the physics-based energy functions, the knowledge-based energy functions yield much smoother energy landscape. Moreover, unlike the physics-based energy functions computations which require intensive floating-point operations, the knowledge-based energy functions usually demand large memory space. For example, the all-atom distance-based energy function [9] requires at least 485MB memory if fully-loaded; a doublet torsion angle scoring function can easily occupy over 100MB of memory storage. Evaluation of a knowledge-based energy function needs to search a large memory space to retrieve the appropriate value, which can be regarded as mainly memory-intensive computation.

2.3 Hybrid Energy Functions

The hybrid energy functions is a combination of physics- and knowledge-based terms by using the knowledge-based terms to replace some of the physics-based terms to "soften" the energy landscape. The weights of the physics- or knowledge-based terms are usually assigned by the regression method [14, 15] via fitting predicted and experimentally determined models to a given set of training structures. For example, the Rosetta energy function [16, 17] is a complex combination of physics- and knowledge-based terms of salvation, residue pair interactions, strand pairing, sheet, helix-strand packing, radius of gyration, Cβ density, and van der Waal repulsion. Due to the physics- and knowledge-based terms involved, the hybrid energy function computations can be both computation- and memory-intensive.

2.4 Analysis of Protein Energy Function Evaluation

Table 1. Analysis of performance and resource demands in evaluating hybrid, knowledge-based, and physics-based energy functions

	Rosetta Energy Function (hybrid)	Distance-based Energy (knowledge-based)	Lennard-Jones Potential (physics-based)
# of floating point operations per cycle	0.074	0.012	0.093
Floating operations per graduated instruction	0.113	0.020	0.153
% L2 Cache Missing	6.82%	23.01%	1.31%
% CPU utilization	87.92%	46.83%	95.53%

Our study shows that knowledge-based, physics-based, and hybrid protein energy function yield very different computational resource requirement. Table 1 shows the performance and resource demand analysis of the computation of the Rosetta energy function [16], the distance-based energy function [9], and the 6-12 Lennard-Jones potential [20], which are hybrid, knowledge-based, and physics-based, respectively. The computation is carried out exclusively on a single processor computer with 1.6GHz Intel Pentium 4 CPU, 512MB memory, 8K L1 cache, and 256K L2 cache. The application performance data is obtained by PAPI [18] and Perfsuite [19], which are performance tuning software packages based on hardware performance counters. One can find that the physics-based Lennard-Jones potential has a high density of floating point operation, low cache missing rate, and high CPU utilization rate. In contrast, the distance-based energy function, which is completely knowledge-based, has very few floating point operations, high cache missing rate, and low CPU utilization. The Rosetta energy function, which is a hybrid energy function with combination of knowledge- and physics-based terms, has the performance characteristics between the pure knowledge- and physics-based energy functions. This analysis inspires us that "consolidating" terms with heterogeneous resource requirements may lead to computational resource efficiency and application performance improvement.

3. Consolidation

"Consolidation" is an important approach to achieve the efficient usage of computer resources. The rationale of consolidation is to increase computational density to a processor or server in order to improve the CPU and other resources utilization. The well-known consolidation approach is "server consolidation" [24] by packing applications onto fewer servers so as to improve server usage efficiency and thus reduce the total number of servers required in an organization. In this paper, we advocate the approach of "thread consolidation" to improve resource efficiency of protein energy function computation so as to improve resource efficiency and reduce computation time in protein energy function computation.

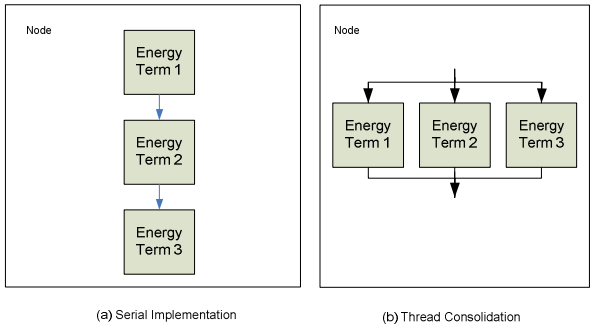

Figure 1. Thread Consolidation and Serial Implementations for a Hypothetical Protein Energy Function Evaluation with 3 Energy Terms

In most protein modeling applications, the protein energy function evaluation is executed in a naïve serial mode, i.e., various terms or interactions are computed sequentially. In thread consolidation, concurrent threads are spawned to carry out independent terms or interactions. When one thread is stalled on any resources, the other active threads can be swapped in the execution mode. As a result, the computational resource utilization rate can be improved and thus the protein energy function evaluation time can be reduced. Thread consolidation will be particularly effective if the computational components carried out by concurrent threads have different demands of computational resources.

The overhead introduced by thread consolidation is thread management, including operations of thread creation, scheduling, and termination. However, compared to the thread management overhead, a complicated protein energy function involving a lot of computations have much longer duration. Therefore, the reduced time gained from more efficiently usage of computation resource can exceed the thread management overhead and thus lead to overall performance improvement. Figure 1 illustrates evaluating a hypothetical protein energy function using thread consolidation and serial modes.

Another advantage of thread consolidation is its capability of automatically taking advantage of the multi-core or shared-memory multi-processor computing platform. When the protein modeling program is ported to a computer with multiple processors or cores, the multiple threads in energy function evaluation can run concurrently on these processors and cores and automatically adapt to the shared-memory multi-processor or multi-core platforms.

4. Computational Results

We applied the thread consolidation technique to our protein loop structure modeling program [23], where there are both memory- and computation-intensive terms involved, as well as our MD program for protein-solvent interaction, where mainly computation-intensive terms involved. All executables are generated by the gcc compiler with "–O3" optimization option. The thread consolidation programs use the pthread library. The computation performance data are obtained by the PAPI [18] and PerfSuite [19] packages. All computations are executed exclusively and the computation time measures are based on the average of 10 runs. Both the serial and the consolidation implementations produce exactly the same simulation results.

4.1 Consolidation in Protein Loop Structure Prediction

We applied the consolidation technique to our protein loop structure prediction program [23] where three energy terms are incorporated using a GA method. These energy terms include the pair-wise atom distance-based score (knowledge-based) [9], the triplet torsion angle potential (knowledge-based) [24], and the soft-sphere energy (physics-based) [22]. Considering that the duration of distance-based score computation is significantly longer that of the other two, we divide the load of the distance-based score computation into three components carried out by three corresponding threads so that each thread has roughly the same duration as the threads computing the triplet torsion angle potential or the soft-sphere energy.

Figure 2 shows the average computation time comparison of the serial and the consolidation implementations of the energy function evaluation in our loop structure prediction program on a 12-residue

protein loop 1akz(181:192) on a computer with a single Intel Pentium 4 processor. The computation time to generate 10 decoys is reduced from 7,624 seconds in serial implementation to 7,064 seconds in thread consolidation implementation, which is 7.35% speedup. Figure 3 depicts that the average CPU utilization rate in consolidation mode is improved to 60.67% from 43.45% in serial mode for the energy function evaluation computation specified in Figure 2. Figure 4 shows the protein loop structure prediction computation carried out on a computer with a single PowerPC CPU, where the thread consolidation implementation yields 10.62% performance improvement. Figure 5 shows the protein loop structure modeling computations run on a computer with two SMP processors. One can find that the program with thread consolidation energy function evaluation can automatically take advantage of the two existing shared-memory processors with computational time that is 11.43% less than half of that of the serial mode. In conclusion, due to different computational resource demands of the energy terms presented in our protein loop structure prediction program, the thread consolidation technique improves the CPU utilization and thus leads to considerable overall simulation speedup.

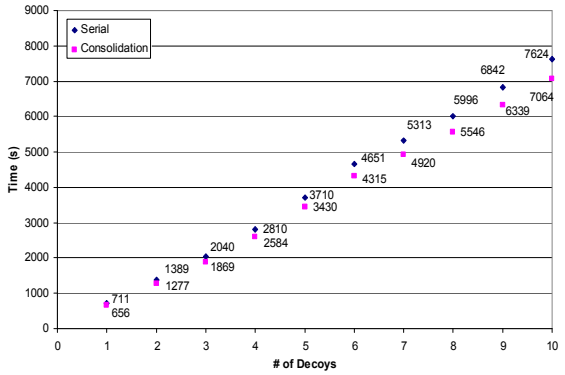

Figure 2. Average decoy generation time in serial energy function evaluation and thread consolidation energy function evaluation. The computation is carried out on a single processor computer with 1.6GHz Intel Pentium 4 CPU, 512MB memory, 8K L1 cache, and 256K L2 cache

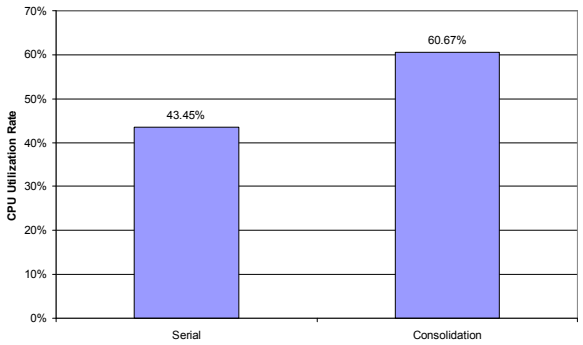

Figure 3. Thread consolidation improves the CPU utilization rate from 43.45% in serial mode to 60.67% in the computations specified in Figure 2

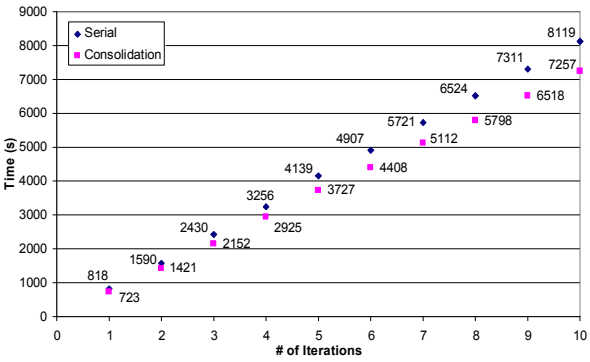

Figure 4. Average decoy generation time in serial energy function evaluation and thread consolidation energy function evaluation. The computation is carried out on a computer with 1.67GHz PowerPC CPU, 1GB memory, and 512KB L2 cache

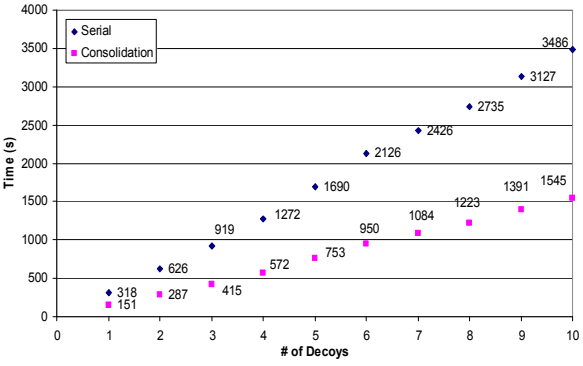

Figure 5. Average decoy generation time in serial energy function evaluation and thread consolidation energy function evaluation. The computation is carried out on a 2-processor SMP computer with 2.8GHz Intel Xeon CPU, 1GB memory, 16K L1 cache, and 1MB L2 cache

4.2 Consolidation in Molecular Dynamics

We also applied the thread consolidation technique to our MD program studying the many-body friction tensor of protein molecule in water [25]. The computations of the molecular system Hamiltonian include the interactions between protein and water molecules as well as the inter-water molecules interactions, which are estimated by calculating the simple Lennard-Jones potential and the electrostatic potential. These interactions are later used to estimate the protein force and torque for Brownian motion simulation. The MD Hamiltonian computation is mainly computation-intensive.

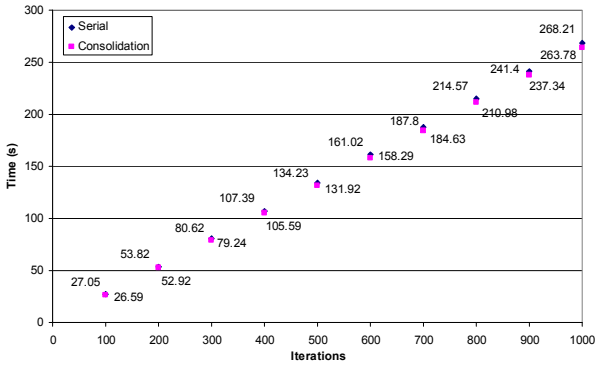

Figure 6. Comparison of computational time vs. iterations using serial and consolidation energy function evaluations in the MD program. The computation is carried out on a single processor computer with 1.6GHz Intel Pentium 4 CPU, 512MB memory, 8K L1 cache, and 256K L2 cache

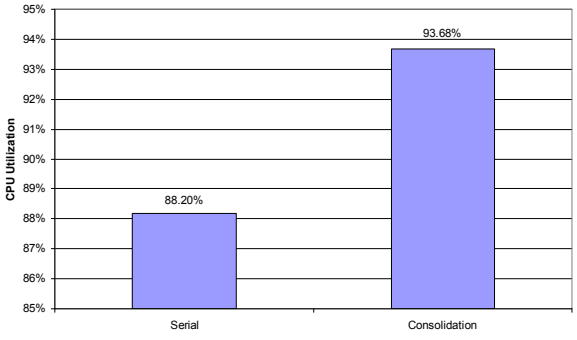

Figure 7. Thread consolidation improves the CPU utilization rate from 88.20% in serial mode to 93.68% in the MD program running on the single processor computer with 1.6GHz Intel Pentium 4 CPU, 512MB memory, 8K L1 cache, and 256K L2 cache

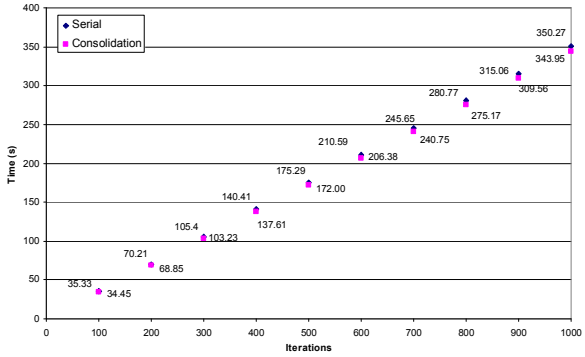

Figure 8. Comparison of computational time vs. iterations using serial and consolidation energy function evaluation in the MD program. The computation is carried out on a computer with 1.67GHz PowerPC CPU, 1GB memory, and 512KB L2 cache

In our thread consolidation program, two threads are spawned to carry out the protein-water interaction computation and water-water interaction computation, respectively. Figure 6 shows the computational times as the number of iterations increases in the MD program with serial and consolidation energy function computations on a computer with a single Intel processor. The molecular system in our computational experiment includes 3,125 water molecules and a short protein (1CZQ) with 208 atoms. One can find that the MD program using thread consolidation energy function evaluation yields averagely 1.65% less computational time than the one using serial energy function evaluation after 1,000 iterations are completed. Figure 7 shows that the CPU utilization in the MD program is improved from 88.20% in serial mode to 93.68% in thread consolidation mode. Figure 8 also shows that consolidation energy function evaluation leads to 1.80% computational time reduction in our MD program compared to serial energy function evaluation on a computer with a PowerPC CPU. Our computational results indicate that the thread consolidation technique may also lead to slight performance improvement even when only computation-intensive terms are involved in the energy function.

5. Conclusions

In this paper, we developed a resource-efficient computing paradigm based on thread consolidation to improve computational resource utilization in computationally costly protein energy function evaluation in order to reduce the overall protein

modeling simulation time. We applied the thread consolidation technique in computationally costly protein energy function evaluation, which spawns concurrent threads to carry out computations of various independent components in a protein energy function to increase computation density to a processor. By applying the thread consolidation technique to our protein loop structure prediction program using GA where computation- and memory-intensive energy components are present, our computational results show 7%~11% performance improvement on various hardware architectures compared to serial energy function computation. In our MD program where only computation-intensive terms are presented, the thread consolidation energy function evaluations are still slightly faster than the serial energy function evaluations.

The thread consolidation technique is particularly suitable for protein modeling applications required to evaluate complicated and costly energy functions. In our future research, we plan to investigate optimizing the thread consolidation technique. We also plan to apply the thread consolidation technique in several important protein modeling applications, including Rosetta [16], GROMACS [8], and NAMD [26], with expectation to achieve resource efficiency and performance improvement.

6. Acknowledgements

The work is partially supported by NSF under grant number CCF-0829382 to Y. Li and NCSA 2007 Summer Faculty Fellowship to Y. Li.

7. References

[1] I. Foster, C. Kesselman, and S. Tueske, "The Anatomy of the Grid," *International Journal of Supercomputer Applications*, **15**(3), 2001.

[2] B. Zagrovic, C. D. Snow, M. R. Shirts, V. S. Pande, "Simulation of Folding of a Small Alpha-helical Protein in Atomistic Detail using Worldwide Distributed Computing," *Journal of Molecular Biology*, **323**(5): 927-937, 2002.

[3] R. Das, B. Qian, S. Raman, R. Vernon, J. Thmpson, P. Bradley, S. Khare, M. D. Tyka, D. Bhat, D. Chivian, D. E. Kim, W. H. Sheffler, L. Malmstrom, A. W. Wollacott, C. Wang, I. Andre, D. Baker, "Structure prediction for CASP7 targets using extensive all-atom refinement with Rosetta@home," *Protein*, **69**(8)**:** 118–128, 2007

[4] M. Taufer, C. An, A. Kerstens, C. L. Brooks, "Predictor@Home: A Protein Structure Prediction Supercomputer Based on Public-Resource Computing," *Proceedings of 4th IEEE International Workshop on High Performance Computational Biology (HiCOMB)*, 2005.

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

[6] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, "A second generation force-field for the simulation of proteins, nucleic-acids, and organic-molecules," *J. Am. Chem. Soc.*, **117**:5179–97, 1995.

[7] W. Damm, A. Frontera, J. Tirado-Rives, W. L. Jorgensen, "OPLS All-Atom Force Field for Carbohydrates," *Journal of Computational Chemistry*, **18**(16): 1955-1970, 1997.

[8] W. F. van Gunsteren, H. J. C. Berendsen, "Groningen Molecular Simulation (GROMOS) Library Manual," *Groningen, The Nether.: BIOMOS*, 1987

[9] A. Rojnuckarin, S. Subramaniam, "Knowledge-based interaction potentials for proteins", *Proteins: Structure, Function, and Genetics,* **36**:54-67, 1999.

[10] S. C. Ngan, M. T. Intuye, R. Samudrala, "A knowledge-based scoring function based on residue triplets for protein structure prediction," *Protein Eng. Design and Selection*, **19**(5): 187-193, 2006.

[11] H. Gohlke, G. Klebe, "Statistical Potentials and Scoring Functions Applied to Protein-Ligand Binding," *Current Opinion in Structural Biology*, **11**(2):231-235, 2001.

[12] B. J. McConkey, V. Sobolev, M. Edelman, "Discrimination of native protein structures using atom-atom contact scoring," *Proc. of the National Academy of Sciences*, **100**(6):3215-3220, 2003.

[13] R. Chen, Z. Weng, "A Novel Shape Complementarity Scoring Function for Protein-Protein Docking," *Proteins: Structure, Function, and Genetics*, **51**:397-408, 2003.

[14] H. J. Bohm, "The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure," *J. Comput. Aided Mol. Des*., **8**(3): 243-256, 1994

[15] A. N. Jain, "Scoring noncovalent protein-ligand interactions: continuous differentiable function tuned to compute binding affinities," *J. Comput. Aided. Mol. Des.*, **10**(5): 427–440, 1996.

[16] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, D. Baker, "Rosetta in CASP4: progress in ab initio protein structure prediction," *Proteins: Structure, Function, and Genetics*, **5**: 119-126, 2001.

[17] P. Bradley, D. Chivian, J. Meiler, K. M. Misura, C. Rohl, W. Schief, W. J. Wedemeyer, O. Schueler-Furman, P. Murphy, J. Schonbrun, C. E. Strauss, D.

Baker, “Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation.” *Proteins: Structure, Function, and Genetics*, **53**: 457-468, 2003.

[18] S. Browne, C. Deane, G. Ho, P. Mucci, “PAPI: A Portable Interface to Hardware Performance Counters,” *Proceedings of Department of Defense HPCMP Users Group Conference*, 1999.

[19] R. Kufrin, “PerfSuite: An Accessible, Open Source Performance Analysis Environment for Linux,” 6th *International Conference on Linux Clusters: The HPC Revolution*, 2005.

[20] J. E. Lennard-Jones, “Cohesion,” *Proceedings of the Physical Society*, **43**: 461-482, 1931.

[21] H. Zhang, L. Lai, Y. Han, Y. Tang, “A Fast and Efficient Program for Modeling Protein Loops,” *Biopolymers*, **41**: 61-72, 1997.

[22] Y. Li, I. Rata, E. Jakobsson, “Multi-Scoring Functions Sampling in Protein Loop Structure Prediction,” submitted to *Journal of Computational Biology*, 2008.

[23] I. Rata, Y. Li, E. Jakobsson, “Backbone Statistical Potential from Local Sequence-Structure Interactions in Protein Loops,” in preparation, 2008.

[24] M. R. Marty, M. D. Hill, “Virtual hierarchies to support server consolidation,” *Proceedings of 34th Annual International Symposium on Computer Architecture*, 2007.

[25] Y. Zhang, Y. Li, M. H. Peters, “Nonequilibrium, Multiple-Time Scale Simulations of Ligand-Receptor Interactions in Structured Protein Systems,” *Proteins: Structure, Function, and Genetics*, **52**(3): 339-348, 2003.

[26] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, K. Schulten., “NAMD2: Greater scalability for parallel molecular dynamics,” *Journal of Computational Physics*, **151**: 283-312, 1999.

[27] Y. Li, A. J. Bordner, Y. Tian, X. Tao, A. Gorin, “Extensive Exploration of the Conformational Space Improves Rosetta Results for Short Protein Domains”, *Proceedings of 7th Annual International Conference on Computational Systems Bioinformatics*, 2008.

[28] Y. Li, C. E. M. Strauss, A. Gorin, “Parallel Tempering in Rosetta Practice,” *Proceedings of International Conference on Bioinformatics and its Applications*, 2004.

[29] A. A. Rabow, H. A. Scheraga, “Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator,” *Protein Science*, **5**(9):1800-1815, 1996.