Parallelization and Performance Characterization of Protein 3D Structure Prediction of Rosetta

Wenlong Li[§], Tao Wang[§], Eric Li[§], David Baker[§+], Li Jin[§], Steven Ge[§], Yurong Chen[§], and Yimin Zhang[§]

[§] *Intel China Research Center*
Intel Corporation
{wenlong.li, tao.wang,eric.q.li}@intel.com

[§+] *Department of Biochemistry*
University of Washington
dabaker@u.washington.edu

Abstract

The prediction of protein 3D structure has become a hot research area in the post-genome era, through which people can understand a protein's function in health and disease, explore ways to control its actions and assist drug design. Many protein structure prediction approaches have been proposed in past decades. Among them, Rosetta is one of the best systems. However, the huge time complexity of Rosetta, e.g. a few days to predict a protein, limits its wide use in practice.

To accelerate the prediction of protein 3D structure in Rosetta, this paper presents three different approaches, i.e., non-interactive, periodic interactive and asynchronous dynamic interactive scheme, to parallelize Rosetta. The asynchronous interactive scheme, with the adaptation of dynamic solution interaction, outperforms the other two, delivering much faster convergence speed and better solution quality. Detailed measurements and performance analysis also indicate that parallel Rosetta with asynchronous dynamic interactive scheme scales well.

1. Introduction

Prediction of protein 3D structures is one of the most important problems in Molecular Biology, which can be simply stated as: given the sequence of amino acids of a protein, what is the three dimensional structure? Biochemists have established that a protein's spatial structure dominates its function. In attempting to understand a protein's function in health and diseases and explore ways to control its actions, scientists often first determine its spatial structure.

In biology, the lineal amino acid sequences first fold to nonlinear secondary structures (alpha helix, beta sheet and loops) and then form tertiary and quaternary structures. The protein 3D structure is typically obtained by x-ray crystallography or nuclear magnetic resonance spectroscopy (NMR), which is costly and very time consuming for high-throughput production. Therefore a computational solution of this problem, i.e. "protein structure prediction", has been an active research field over the last forty years [1,2,3,4,5].

Prediction of protein 3D structures directly from their amino acid sequences is referred to as "ab initio" method. Among existing "ab initio" systems, Rosetta is one of the best in the international competition of "Critical Assessment of Techniques for Protein Structure Prediction" (CASP). Rosetta is developed by the Baker laboratory of University of Washington, which uses *simulated annealing* optimization algorithm (SA) to find optimum protein tertiary structures[5,6,7]. One of the key ingredients of Rosetta is *score function* that assigns an evaluation score to the 3D structure and guides the protein structure search toward a protein-like fold by simulated annealing. Figure 1 shows one prediction example of Rosetta [7].

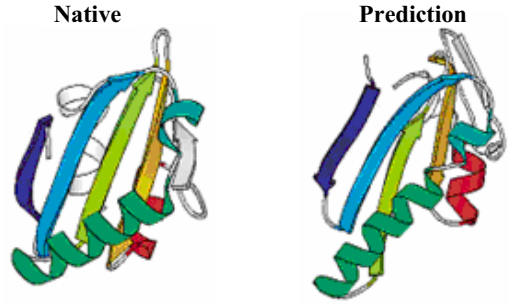


Figure 1. Prediction result of Rosetta for T087

Although Rosetta can predict protein 3D structures more accurately, the complex elevation scores and very large protein structure search space, or in other words, the huge computational requirement inhibits its wide use in practice, e.g., it will take a few days to perform a structure prediction per protein. In literature, there are little efforts to optimize and parallelize Rosetta so as to finish a given task in a reasonable time scale. Baker lab of Washington Univ. employs a simple parallel scheme, running Rosetta

1-4244-0054-6/06/$20.00

programs independently on 80 machines simultaneously, to shorten the computation time. Each machine predicts a list of possible protein structures. Then the server node clusters all the resulting candidates (roughly 2000~10000 structures), and selects the largest cluster as the highest confidence prediction. This scheme, though simple enough in implementation, is not very effective in terms of parallel efficiency. One reason is the time to get the candidate structures varies a lot for each machine, and all the other machines have to wait the one with the longest running time before clustering. It incurs a lot of load imbalance among the computing machines, and as a result, under-utilizes the computation powers. The other reason is that some machines may perform useless computations when they trap in a local optimal solution of the protein structure search of simulated annealing, which slows down the whole execution time.

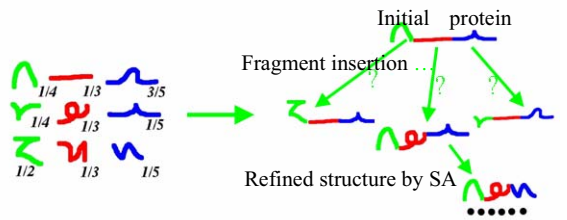
The contributions of this paper are as follows.

1) In order to overcome the limitations in the existing parallel scheme, we present three different data level parallelization schemes to improve Rosetta performance based on parallel simulated annealing, where the asynchronous dynamic interactive scheme exhibits the best performance in terms of speed and protein 3D structure prediction quality;
2) The three parallel schemes are implemented with the OpenMP parallel programming model and measured on a 16-way shared memory multiprocessor machine. The experiments show that asynchronous interactive scheme is the best among all the three schemes. We also conduct detailed performance analysis on this scheme. The analysis results indicate that the asynchronous dynamic interactive scheme has very low parallel overhead, delivering a very good scaling performance on shared memory multiple processor systems.

The rest of the paper is organized as follows. Section 2 gives an overview of Rosetta's structure prediction algorithm. Section 3 proposes our parallel Rosetta schemes. The experimental results and workload characteristics are reported in section 4. Finally, section 5 summarizes the paper.

2. Overview of Rosetta

According to homogeneous theory in biology, similar amino acid sequences generally are likely to have the similar protein structure. The Rosetta method of *ab inito* structure prediction is based on the assumption that the local structure of a protein is similar to the structures of a segment of amino acid sequences in known protein structure database (PDB). As shown in figure 2, each amino acid segment may have many possible 3D structures called *fragments*. Rosetta uses "fragment insertion technique" to assemble these local structure segments into a whole protein structure. Simulated annealing optimization algorithm searches the assembled optimum structure by an evaluation score that favors a protein-like fold, e.g., compact structures and buried hydrophobic residues etc. Since the searched protein structure space is very large and complex, Rosetta carries out a large number of independent simulations. Then these resulting structures about 2000~10000 candidates are clustered and the centers of the largest clusters are selected as the highest confidence predictions.



(a)Structure fragment library (b) fragment insertion by SA

Figure 2. Protein structure prediction by Simulated Annealing of Rosetta

2.1 Evaluation scores

In biology, a lower energy normally means better stability in structure. Given a possible protein structure, score functions are used to evaluate whether the structure is better than another structure. The score function of Rosetta is based on the probability of the structure being the native structure given the sequence of amino acids. It captures sequence dependent features of protein structures, such as the burial of hydrophobic amino acid in the core, as well as universal sequence independent features, such as the assembly of beta-strands into beta-sheets. The main components of score functions are defined as following:

$$P(structure|\,sequence) \propto P(sequence|\,structure)P(structure)$$
$$\approx P_{env} Pp_{air} P(structure)$$
$$with\ \ P_{env} = \prod_i P(aa_i \mid E_i)$$
$$P_{pair} = \coprod_{i<j} \frac{P(aa_i, aa_j \mid E_i, E_j, r_{ij})}{P(aa_i \mid E_i, r_{ij}) P(aa_j \mid E_j, r_{ij})}$$

where aa_i is the i^{th} amino acid of a protein, r_{ij} is the distance between the centroids of amino acid i and j, and E_i is the structural environment at position i which is usually defined in terms of solvent accessibility and secondary structure.

In practice, Rosetta has five main evaluation scores, i.e., score0, score1, score2, score3 and score5. Each score

consists of weighted sub scores, e.g. score0 = vdw_weigth*vdw_score+ env_weight*env_socre+…. The scores and their weights are shown in table 1. For more information, it can be referred in [5, 6].

Table 1. Weights of Score0,1,2,3,5 in Rosetta

Weights	Score0	Score1	Score2	Score3	Score5	Functions
vdw_weight	1	1	1	1	1	Penalties for over compact and overlaps between atom pairs
env_weight		1	1	1	1	Evaluate the energy of a structure based on atoms, e.g. Ca, Cb atoms
pair_weight		1	1	1	1	
cb_weight			0.5	1	0.5	
sheet_weight		1	1	1	1	Evaluate the energy of a structure based on the secondary structure, e.g., alpha helix, beta sheets and strands.
ss_weight		0.3	1	1	1	
hs_weight		1	1	1	1	
rsigma_weight				1		Distance between secondary structure pairs
rg_weight				3		The average distance between all pairs of Ca atoms

2.2. Simulated Annealing

The searched protein structure space is very large and complex with many local minimums. Rosetta adopts simulated annealing algorithm to discover the global optimum protein structure. Simulated annealing (SA) is a heuristic optimization algorithm for difficult combinatorial optimization problems, especially ones where a desired global extremum is hidden among many poor local extrema[8,9].

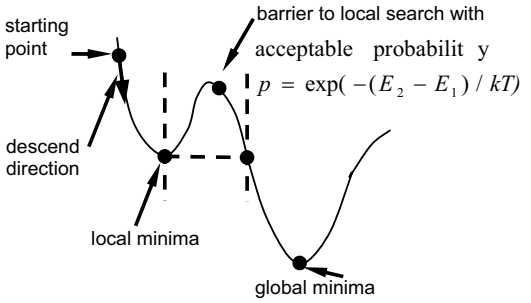


Figure 3. Simulated Annealing

The basic idea of SA is to track a Markov chain in the feasible solution space of the given optimization problem as shown in figure 3. Starting with an initial solution, SA repeatedly generates succeeding solutions using a local search procedure. The succeeding solution will be accept or rejected according to the Boltzmann acceptance probability $e^{-\Delta C/T}$. After some iterations of the local search procedure, the temperature is decreased and the optimization continues on a new temperature level. When the system is frozen to zero temperature, the best solution found during the optimization is outputted. An outline of the SA algorithm is described as following:

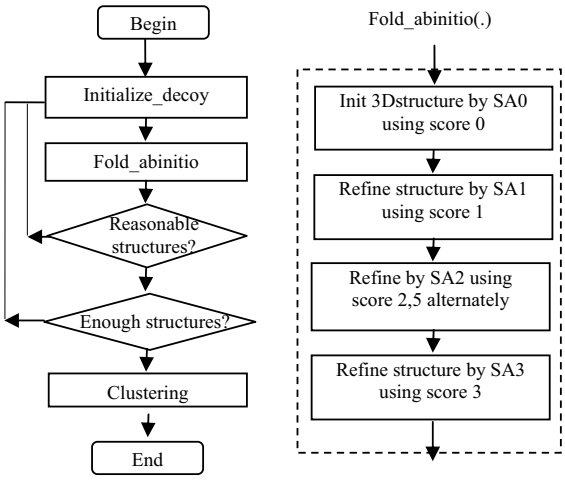
```
PROCEDURE Sequential SA
BEGIN
  s ← Initial Solution in S
  T← Initial Temperature T0
  DO
    DO
      s*← Perturb(s)
      ΔC← Cost(s*) - Cost(s)
      IF ΔC< 0 OR e^(-ΔC/T) > random(.) THEN
        s← s*
      END IF
    UNTIL Equilibrium
    T← Decrement(T)
  UNTIL Frozen
END PROCEDURE
```

For Rosetta, S defines the searched protein structure space, Cost(.) uses score functions of section 2.1 to evaluate how good a protein-like structure is. Perturb(.) gets a succeeding structure *s** from *s* by randomly replacing the fragment at sequence position *i* with a new local structure fragment. The search path of SA can be observed as a *Markov chain* transition path, i.e., current solution only depends on the last previous solution. After enough annealing iterations, the discovered best solution is outputted as a candidate structure. Since the searched protein structure space is very large with many local minimums, Rosetta generates enough candidate structures, e.g. 2000~10000 structures and selects the centers of the largest clusters as the predicted protein structures.

2.3. Flowchart of Rosetta

There are five main score functions, i.e., score0, score1, score2, score 3 and score 5 illustrated in section 2.1. The kernel module *Fold_abinitio* of Rosetta uses these evaluation scores respectively in four simulated annealing processes to refine a 3D protein structure in serial. For convenient description, we name the four SAs as SA0, SA1, SA2, and SA3. As a whole, Figure 4 shows the Rosetta flowchart. Firstly, the *Initialize_decoy* module initializes the parameters and buffers. Then the module *SA0* generates an initial 3D structure with score0 by assembling local fragment structures of known amino acid segments. *SA1*, *SA2* and *SA3* perform similar calculations with different score functions. Each of them refines the structure generated in the previous module by repeatedly executing the fragment assembly and score evaluation until the termination condition is satisfied. The *Reasonable_structures* module filters out impossible

structures by rules. After *Fold_abinitio* module generates enough candidate structures, *Clustering* module classifies these large numbers of candidate structures into a few clusters. And the centers of the largest clusters are selected as the predicted protein structures.



(a) Flowchart of Rosetta (b) Flowchart of fold_abinitio()

Figure 4. Flowchart of Rosetta

3. Parallelization of Rosetta

With the booming of multi-core processor and the prevalence of shared memory processing, it is important to exploit thread level parallelism within application to fully take advantage of multi-core or multi-processor processing capability. As experimental data indicates, Rosetta spends 99.9% of time in the four simulated annealing(SA) processes of Fold_abinitio module. Therefore, parallelizing the four SA processes is a straightforward way to enhance the whole application's performance.

Task and data level decomposition are two primary schemes to parallelize an application. Specifically, in Rosetta, we don't exploit task level parallelism within each SA process due to the strong loop carried data dependency between consecutive iterations. Also, we don't exploit task level parallel scheme between adjacent SA processes due to dependency. Furthermore, the solution quality achieved by task level parallelization is not as good as data parallelization scheme, where single Markov search path of task level parallelization in global search space is more likely to trap in a local optimum than multiple Markov search paths in the data level parallelization[11]. In addition, task level parallelization scheme is more prone to have high communication cost due to frequent solution exchanges.

In order to improve both the execution speed and solution quality, we propose a novel data-level parallelization scheme, tracing multiple Markov-chain search paths within one global search space at the same time and allow solution communicate with each other to avoid local minimum. First, each processor initializes a random selected structure individually, assembles and evaluates its temporal solution in one Markov-chain search path of the global space. Then, during the process of each path searching, each processor communicates with each other by exchanging their solutions periodically or dynamically. Finally, the best solution among these processors is chosen and outputted. According to the interaction mode, they can be categorized into non-interactive, periodic interactive and asynchronous dynamic interactive parallel schemes as shown in figure 5.

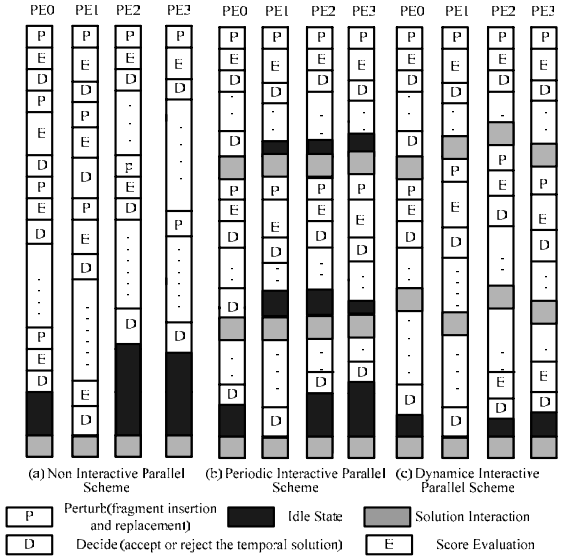


Figure 5. Parallel Rosetta schemes

There are several advantages of data parallel scheme over the task level decomposition and the naïve parallel scheme mentioned before. First, unlike the frequent communication pattern in the task level parallelization, all the working processors in this scheme are not close-coupled with each other. The communications only occur in some regular period of iterations, thus significantly reducing the high communication cost to transfer the state and score among all the processors. Second, tracing multiple search paths simultaneously is less likely to cause system trapping in a local optimum, which makes Rosetta more robust, achieving a much quicker convergence speed and better solution quality than task level parallelization. Following sections describe these three data-level parallelization schemes in detail, trying to exploit different communication patterns among the working threads. Since asynchronous dynamic interaction decreases much communication overhead

through asynchronous interaction and has lower probability of entering the local minimum, it achieves faster convergence speed and better solution quality than others.

3.1. Non-interactive scheme

In the non-interactive scheme, multiple simulated annealing processes run independently on different processors. During this period, each process will work only on one protein structure, without any interactions with other processes, until all the protein structures are obtained. After that, the best structure is selected as the predicted result.

The advantage of this parallel scheme is that there are multiple Markov chains can be searched at the same time, which significantly reduces the possibility of trapping in local minimum and delivers a better solution than single Markov chain search scheme. However, as mentioned before, this scheme also has several disadvantages: first, it suffers a lot from load imbalance especially when each Markov chain has different time to reach a solution, where some processors will be idle waiting for the slowest Markov chain searching. Second, some SA processes may still trap in local optimum although multiple chains provide a much larger searching space in practice. As shown in figure 5(a), the black blocks represent the time in the waste idle state. Apparently, PE1 has the longest evaluation time, and the other three processes have to wait until PE1 obtains the final result.

3.2 Periodic interactive scheme

To overcome the inefficiencies in the non-interactive scheme, timely exchanging temporal solutions among processors is essential to diminish the load imbalance. One possible way is to exchange intermediate structure solutions among processors at fixed time interval or fixed step iterations, to choose current best structure as a new starting point for the following searches. A master node will decide the solutions collected from each processor and dispatch the best one to them for the following searches.

In this scheme, periodic interaction among processors helps SA processes to achieve a quicker convergence time with a much lower probability to entering a local minimum. However, it still has some imbalance overhead, though it has successfully distributed the whole process into a number of periodic stages. During the interval of two adjacent interactions, these processes will work independently similar to the non-interactive scheme. While reaching the interaction time, all the processors will exchange information and the master node will collect all the solutions from each processor and dispatch the best one. There will be a mandatory synchronization point in the interaction time. Therefore, all the processors may still suffer from the load imbalance though not that obvious than the non-interactive one. Figure 5 (b) displays the periodic interactive parallel scheme. Grey blocks represent the stages of collecting, decision and dispatching solution in the master node and the black blocks are the waiting time in the idle state. It can be observed that there is a lot of idle time wasted at the synchronization point.

3.3 Asynchronous dynamic interactive scheme

As previously described, we can find two critical issues: load imbalance and trapping in local minima, which are not solved well in above two parallel schemes. With the in-depth understanding of the SA algorithm, score calculation period, and possible communication patterns existed in Rosetta, we propose a novel asynchronous dynamic interactive parallel scheme. It uses the running profiling information to trigger the interaction dynamically. For example, we collect the profiling information of the trial number and the acceptance ratio in running stage of Rosetta. When a processor's acceptance ratio is below a specified threshold or runs enough trials, i.e. the processor arrives at a local optimum or need accelerating near a temporal solution for faster convergence, it triggers a interaction between the processor and the master node. Then the trial number and the acceptance ratio of the interacted processor are reset.

To further reduce the communication cost, the idea of latency hiding technique in compiling optimization is used to overlap the communication with computations, which is called *asynchronous communication*. In this scheme, the *global solution* in master node saving the current best solution among processors is protected by a lock. During the interaction, if processor's protein structure is better than the global solution in the master node, it updates the global solution with its structure. Otherwise it replaces its own solution with the global one. Since the interaction seldom occurs at the same time, each processor may have a different solution after interaction, which reduces the possibility of trapping in the local minimum and avoids the frequent communication contentions. Figure 5(c) illustrates the asynchronous dynamic interactive scheme. Represented by grey blocks, different processor adaptively interacts with the master node at different time. It is obvious that the black blocks, waste idle time, are decreased greatly than figure 5(a) and figure 5(b).

In contrast, the periodic interactive scheme is *synchronous communication*, in which all processors are stalled to wait for the solution selection among all processors until the interaction completes. Therefore, the communication overhead is much larger than the asynchronous dynamic interactive scheme. Another

potential benefit of asynchronous scheme is that each processor starts from a different solution, rather than the same solution in the synchronous scheme, resulting in a much lower probability of entering the local minimum.

4. Experimental results

We use OpenMP[13] programming model to implement different parallel schemes as illustrated in section 3. OpenMP provides a rich set of features to simplify the programming efforts to thread the application, where the natural "parallel" loops and the independence among all the candidate search paths make it an ideal case for OpenMP parallelization. Furthermore, we use the Intel VTune[12] performance analyzer to identify the hot spots in functional profiling, to guide the optimization with various techniques, e.g., loop splitting, and data structure reorganization. To characterize the parallel performance, Intel VTune Thread Profiler[12] is used to qualify the low level metrics, i.e., synchronization, locks, load imbalance, etc.

The performance measurement of parallel Rosetta is conducted on a 16-way Intel Xeon shared-memory multiprocessor system. It has 16 x86 processors running at 3.0GHz, 4 levels of cache with each 4MB L4 cache shared amongst 4 CPUs. The sizes of the L1, L2 and L3 caches are 8K, 512K and 4MB respectively. As for the interconnect, the system uses two 4x4 crossbars. We use Intel 8.0 Fortran OpenMP compiler tool chain to generate the executable codes with options –O3 –ipo –openmp, to enable the high levels of compiler optimizations.

For the test data set, we use 4 proteins with different scale in protein sequence length. They are 2ptl, 1557, 1554 and 1560 with 60, 174, 232, and 321 amino acids, respectively[6,7]. Due to the random nature of this SA application, we use 20 iterative running for a single round measurement, and average the results to generate the final performance. Two metrics, speedup and score, are both used to evaluate the application's performance, where the speedup is defined as follows:

$$Speedup = \frac{\text{Processing time on one processor}}{\text{Processing time on N processors}}$$

Score is the evaluation score of the final solution. To make an equal performance comparison, different runs should find the equivalent protein 3D structure close to the native one.

4.1. Solution quality comparison of different parallel schemes

As illustrated in section 2.1, the evaluation score represents the quality of the predicted structure. The lower score, the better protein structure can be achieved. In this work, we conducted extensive studies on a number of proteins, and all of them exhibit very similar behavior and performance. Therefore, we only choose a typical one to present the data throughout the rest of the paper.

Figure 6 shows the solution quality comparison of non-interactive, periodic interactive and asynchronous dynamic interactive parallel scheme on 16p system, where 16P_N is the non-interactive scheme, 16P_P is the periodic interactive scheme, and 16P_D represents the asynchronous dynamic interactive scheme. With the incorporation of interactions among different processors, the score curve has some irregular behavior, not as smooth as the non-interactive one. As indicated from figure 6, the two interactive parallel implementations are consistently better than the non-interactive scheme, yielding better scores and much shorter convergence time due to solution interactions among multi-processors. Similarly, when comparing the asynchronous dynamic interactive scheme with the periodic interactive scheme, we also find that the asynchronous dynamic interactive scheme is more effective than the periodic interactive one due to asynchronous communication and different start solutions after interaction. It confirms that asynchronous communication of the asynchronous dynamic interaction scheme can obtain much better protein structure than synchronous communication of the periodic interaction.

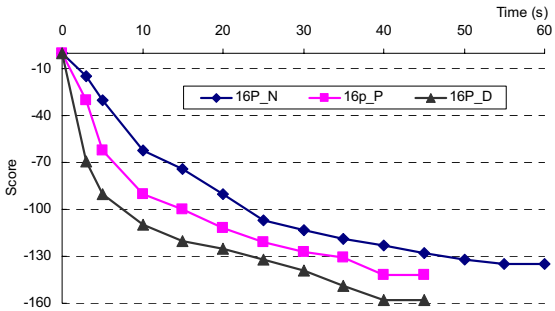


Figure 6. Score vs. time curves of different parallel schemes

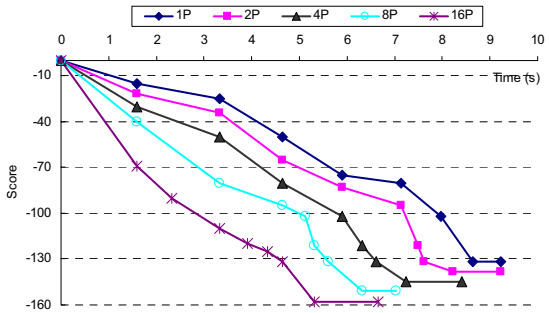


Figure 7. Score vs. time curves of asynchronous dynamic interactive parallel scheme for different processor number

In principle, each processor of parallel Rosetta searches a Markov chain, and multiple chains in Rosetta have a much larger search space than one single Markov chain. As a result, it can achieve lower score, better solution quality and much quicker convergence speed. Figure 7 shows the score-time curves of asynchronous dynamic interactive parallel scheme for different processor number. It uses a $\log_2$ scale on X axis. We can easily observe that to get the same score, Rosetta with one single chain requires much longer time than the parallel one. To summarize, parallel Rosetta achieves lower score with much quicker convergence with the increase of processor number.

4.2 Performance analysis

Figure 8 shows asynchronous dynamic interactive scheme scales well with the increased number of processors, and exhibits almost linear speedup performance. However, non interactive scheme and periodic interactive scheme scale poorly beyond 8 processors.

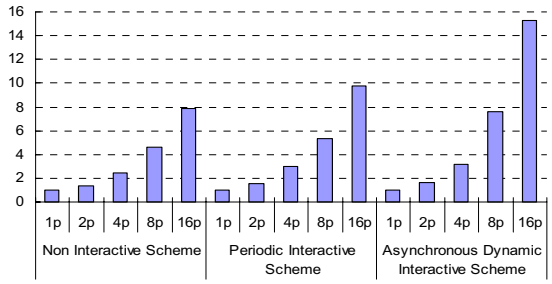


Figure 8. Speedup performance of different parallel schemes

To deeply understand the scalability limiting factors, we characterize the parallel performance from the high level general parallel overheads, e.g., synchronizations penalties, load imbalance, and sequential sections, to the detailed memory hierarchy behavior, e.g., cache miss rates and FSB (front side bus) bandwidth.

Figure 9 depicts the general parallel profiling metrics, where "Parallel" means the running time inside the parallel region, and "Imbalance" represents time spent waiting for other threads to reach the end of a parallel region. The higher parallel region, the potential better speedup can be achieved on highly-threaded architectures. The profiling information suggests that for the asynchronous dynamic interactive scheme, there is almost no synchronization overhead in parallel Rosetta, the sequential area and load imbalance goes up steadily with the increase of processor number, but maintains at a relatively low percentage. While for the periodic interactive scheme and non interactive scheme, as figure 9 indicates, the load imbalance is more prominent than the asynchronous dynamic interactive one, limiting its scaling performance especially on 8 and 16 processors. Overall speaking, parallel Rosetta with asynchronous dynamic interactive scheme is a highly parallel application, and the general parallel limiting factors are insignificant and will not hinder its scalability performance on 16-way system.

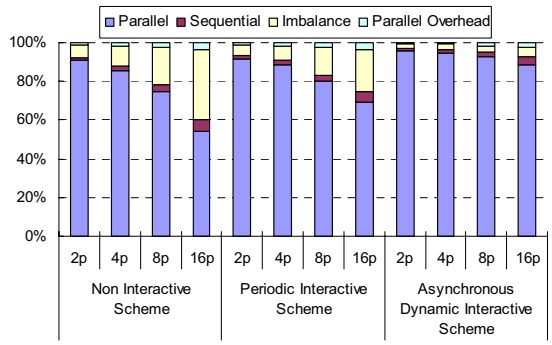


Figure 9. Breakdown of the general parallel limiting factors

Since asynchronous dynamic interactive scheme outperforms the other two in terms of speed, protein 3D structure prediction quality, and scalability performance, we will only study this scheme in the following sections.

Besides the general scalability performance factors, memory subsystem also plays an important role in identifying the scaling performance bottlenecks. We profile the application with VTune, and performance metrics are chosen to be different level cache misses and system memory bandwidth. From figure 10, it is interesting to see that the cache miss rates vary little with the number of processors. Though Rosetta has several large data structures (private to thread) not to fit in L2 and L3 cache, it only works on a small portion of them. The data can be reused in difference phases of score calculation, and smaller enough to fit in the L3 cache. The regular data access pattern and well-organized data structure delivers very good cache performance though it has a relatively large memory footprint.

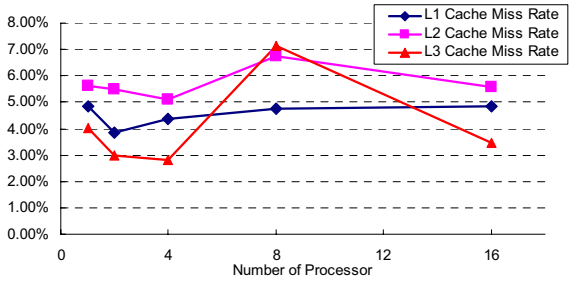


Figure 10. Cache Miss Rates

Generally speaking, memory bandwidth is a key factor which may potentially limit the speedup on more processors, especially for the shared memory system with snoopy based cache coherency model. However, due to the intrinsic data independencies among all search paths searching threads, the cache coherency traffics are tremendously reduced; coupled with the high cache locality performance on the L3 cache, it has a much lower memory bandwidth requirement. Figure 11 shows how the bus bandwidth utilization varies with the number of processors. For all data inputs, the bus bandwidth increases linearly with the number of processors, but far from the saturation (3.2GB/s) even with 16 processors.

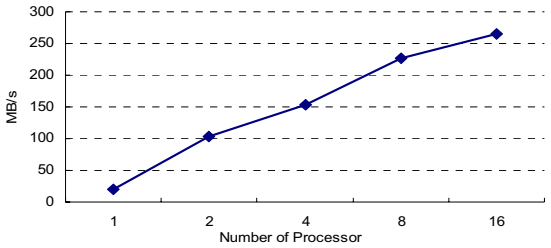


Figure 11. FSB Bandwidth

To summarize, parallel Rosetta exhibits very good scaling performance on symmetric multiple processor systems. It is very promising to scale well on more than one hundred processors.

5. Conclusion

In this paper, we presented a novel asynchronous dynamic interactive parallel scheme to accelerate the 3D protein structure prediction of Rosetta. Experiments show that it has much better performance than several candidate parallel schemes, i.e, non-interactive and periodic interactive parallel schemes, in both the computation time and predicted 3D structure quality. In addition, the exploration of different parallel scheme study, in another aspect, reveals the effectiveness of parallel simulated annealing with interactions among multiple processors.

Besides its good performance, it also delivers nearly linear speedup in the shared memory multi-processor system. Detailed performance analysis indicates that parallel Rosetta is a compute-bound application, displaying very low LLC miss rate and memory bandwidth requirement. The general scaling limiting factors, e.g., barrier and parallel overhead are almost negligible even with up to 16 processors. As a whole, parallel Rosetta exhibits a very nice scalability performance, which can be expected to scale well on even more processors.

Our future work will include extending parallel Rosetta to more than 64 processors to characterize its performance on many-core machines, and improving parallel scheme by other optimization approaches, e.g. parallel simulated tempering and genetic algorithm.

6. References

[1] J Xu, M Li, D Kim, and Y Xu, RAPTOR: Optimal Protein Threading by Linear Programming, Journal of Bioinformatics and Computational Biology, Vol. 1(1):95-117, 2003.

[2] Y Xu, D Xu, Protein threading using PROSPECT: Design and evaluation. Proteins: Structure, Function, and Genetics, Vol.40(3):343-354, 2000.

[3] A Fiser, PK Do, and A Sali, Modeling of loops in protein structures, Protein Science, Vol. 9:1753-1773, 2000.

[4] DT Jones, JJ Ward, Protein secondary structure prediction based on position-specific scoring matrices. J Mol. Biol, Vol.292:195-202, 1999.

[5] KT Simons, C Kooperberg, E Huang, and D Baker, Assembly of protein tertiary structures from fragments with similar local sequences using simulate annealing and Bayesian scoring functions. J Mol Biol Vol.268:209-25, 1997.

[6] KT Simons, I Ruczinski, C Kooperberg, B Fox, C Bystroff, and D Baker. Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. Proteins Vol.34(1) 82-95, 1999.

[7] R Bonneau, J Tsai, I Ruczinski, D Chivian, C Rohl, CE Strauss, and D Baker, Rosetta in CASP4: progress in ab initio protein structure prediction. Proteins Suppl Vol.5:119-26, 2001.

[8] S Kirkpatrick, CD Gelatt, and MP Vecchi, Optimization by Simulated Annealing. Science, Vol.220, 1983.

[9] N Metropolis, AW Rosenbluth, MN Rosenbluth, and AH Teller, Equation of state calculations by fast computing machines, J. Chemical Physics, Vol 21(6):1087-1092, 1953.

[10] TM Nabhan, AY Zomaya, A Parallel Simulated Annealing algorithm with low communication overhead, IEEE Trans on Parallel and Distributed Systems, vol.6(12), 1995.

[11] SY Lee, KG Lee, Synchronous and asynchronous parallel simulated annealing with multiple markov chains, IEEE Trans on Parallel and Distributed Systems, vol.7(10):993-1008, 1996.

[12] The Intel Vtune Performance Analyzer. http://developer.intel.com/software/products/vtune/.

[13] OpenMP Architecture Review Board: ”OpenMP C and C++ Application Program Interface,” Version 2.0, March 2002, http://www.openmp.org.