MEGADOCK-GPU: Acceleration of Protein-Protein Docking Calculation on GPUs

Takehiro Shimoda^{1,2,†}

Takashi Ishida^{1,†}

Shuji Suzuki^{1,2,3,†}

Masahito Ohue^{1,2,3,†}

Yutaka Akiyama^{1,2,‡,*}

¹ Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology

W8-76, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, JAPAN

² Education Academy of Computational Life Sciences, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, JAPAN

³ Research Fellow of the Japan Society for the Promotion of Science

[†] {shimoda, t.ishida, suzuki, ohue}@bi.cs.titech.ac.jp

[‡] akiyama@cs.titech.ac.jp

ABSTRACT

Protein-protein docking is a method for predicting the protein complex structure from monomeric protein structures. Because protein structural information has been increased and the application field has been expanded to more difficult ones such as interactome prediction, a faster protein-protein docking method has been eagerly demanded. MEGADOCK is fast protein-protein docking software but more acceleration is demanded for an interactome prediction, which is composed of millions of protein pairs. In this paper, we developed an ultra-fast protein-protein docking software named MEGADOCK-GPU by using general purpose GPU computing techniques. We implemented a system that utilizes all CPU cores and GPUs in a computation node. As results, MEGADOCK-GPU on 12 CPU cores and 3 GPUs achieved a calculation speed that was 37.0 times faster than MEGADOCK on 1 CPU core. The novel docking software will facilitate the application of docking techniques to assist large-scale protein interaction network analyses. MEGADOCK-GPU is freely available at <http://www.bi.cs.titech.ac.jp/megadock/gpu/>.

Categories and Subject Descriptors

D.1.3 [Programming Techniques]: Concurrent Programming; J.3 [Computer Applications]: Life and Medical Sciences

*Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

BCB '13, September 22 - 25, 2013, Washington, DC, USA
Copyright 2013 ACM 978-1-4503-2434-2/13/09 ...\$15.00.

General Terms

Algorithms, Performance

Keywords

protein-protein docking, structural biology, GPGPU, fast Fourier transform

1. INTRODUCTION

Protein-protein docking is a method for predicting the complex structure from two or more monomeric protein structures based on their shape and physicochemical complementarity. Docking calculation is important as a source of structural information on protein complexes that complements experimentally determined structures, and as a way to understand the mechanisms of protein association [1, 2, 3, 4].

In recent days, protein-protein docking methods are used not only for prediction of a single protein complex but also for an interactome prediction of proteins in a biological pathway [5, 6, 7]. Thus, the importance of protein-protein docking in proteomics and systems biology fields has been increased as well as in structural biology field. However, there are several difficulties in applying protein-protein docking methods to interactome predictions. One of the most critical problems is its calculation speed. Current docking methods are fast enough to predict a single protein complex structure but interactome prediction requires docking calculations for millions of protein pairs. Therefore, the acceleration of protein-protein docking calculation is highly required.

Currently, most of popular docking methods are based on rigid-body docking using voxel representation in 3-D grid space and discrete convolution-based scoring function with fast Fourier transform (FFT). The ZDOCK software [8, 9] is now widely used in structural biology community [10] because of its excellent prediction accuracy [11]. However, ZDOCK unfortunately requires long computation time because it employs seven energy terms for its docking score

function calculation. On the other hand, MEGADOCK [6, 12] uses more compact energy terms and is 8.8 times faster than ZDOCK 3.0 with comparable prediction accuracy. However, even using MEGADOCK, it is difficult to perform a large-scale interactome prediction because of its computation cost. Therefore, further acceleration of protein-protein docking is anticipated for a proteomics-scale studies.

These days, graphics processing units (GPUs) have been transformed into powerful coprocessors for general purpose computing. Current GPUs, such as NVIDIA's Tesla K20, have excellent power efficiency and their computational power supersedes that of CPUs. Also, GPU software development tools, such as NVIDIA's Compute Unified Device Architecture (CUDA) [13], have been developed and they enable us to develop GPU applications much easier. Thus the general purpose GPU (GPGPU) computing techniques have been widely used in various research fields including bioinformatics, such as metagenome sequence mapping [14], molecular dynamics simulation [15] and quantum chemistry calculation [16]. Therefore, the MEGADOCK could be also accelerated using GPU computing techniques.

In this study, we mapped the docking calculation of MEGADOCK onto GPUs and developed fast protein-protein docking software named MEGADOCK-GPU. We implemented almost all processes of MEGADOCK including "FFT", "modulation" and "ligand voxelization". We also implemented the system for utilizing all CPU cores and GPUs in a computation node. As results, MEGADOCK-GPU on 12 CPU cores and 3 GPUs achieved a calculation speed that was 37.0 times faster than MEGADOCK CPU version on 1 CPU core.

2. RELATED WORK

A GPU implementation of FFT-based protein-protein docking was already done by Sukhwani, *et al.* in 2009 [17]. They mapped a FFT-based protein-protein docking tool PIPER [18] onto GPUs, and demonstrated a calculation speed that was 17.7 times faster with 1 GPU than PIPER with 1 CPU core. Sukhwani, *et al.* used cuFFT library [19] for mapping FFT processes onto GPUs, and also mapped several processes onto GPUs. PIPER is a famous docking tool and has shown its good prediction accuracy through international benchmarks [20], but it uses 22 energy terms and is much slower than ZDOCK and MEGADOCK. Therefore, the performance of PIPER is still insufficient for proteomics-scale studies even with GPUs.

3. MEGADOCK

3.1 Overview

MEGADOCK [6] is a protein-protein interaction prediction system using FFT-based protein-protein docking by Katchalski-Katzir algorithm [21]. MEGADOCK is implemented by C++, and distributed under the GPLv3 license. MEGADOCK evaluates each docking pose based on three types of score function, including shape complementarity, electrostatics and desolvation free energy. MEGADOCK calculates these functions with a single FFT calculation, and is much faster than ZDOCK, which requires seven FFT calculations, and PIPER, which requires 22 FFT calculations. In addition, MEGADOCK has been already parallelized by MPI and OpenMP for multiple combinations of

Figure 1: The process flow of FFT-based docking tools.

protein pairs.

Figure 1 shows the flow of the docking processes of MEGADOCK. The flow of MEGADOCK is mainly based on Katchalski-Katzir algorithm. In the Katchalski-Katzir algorithm, the pseudo interaction energy score (called the docking score S) between a receptor protein and a ligand protein is calculated as a convolution of two discrete functions by using N^3 -point forward FFT and inverse FFT (IFFT), as follows:

$$S(\mathbf{t}) = \sum_{\mathbf{v} \in \mathcal{V}} R(\mathbf{v})L(\mathbf{v} + \mathbf{t}) \quad (1)$$

$$= \text{IFFT}[\text{FFT}[R(\mathbf{v})]^* \text{FFT}[L(\mathbf{v})]], \quad (2)$$

where R and L are the discrete score function of the receptor and ligand proteins, $\mathbf{v}$ is a coordinate in a 3-D grid space $\mathcal{V}$, and $\mathbf{t}$ is the parallel translation vector of the ligand protein. $*$ is the complex conjugation operator. N , called the size of FFT, is the double of the size of grid. In order to find the best docking pose, possible ligand orientations are exhaustively examined at n_θ rotation angles for a given stepsize θ . For each rotation, the ligand protein is translated into $N/2 \times N/2 \times N/2$ voxels in the $\mathcal{V}$ grid space (where $N/2$ is a edge of $\mathcal{V}$). The decoy (relative conformation model of receptor and ligand) that yields the highest value of S for each rotation is recorded. In this method, a total of $n_\theta \times N^3$ docking poses are evaluated for one protein pair. To directly execute the simple convolution sums in eq. (1), $\mathcal{O}(N^6)$ calculations are required; however, this is reduced to $\mathcal{O}(N^3 \log N)$ using the FFT in eq. (2). FFT-based docking calculation is composed of these processes; initialization (P1), receptor voxelization (P2), forward FFT of a receptor (P3), ligand voxelization (P4), forward FFT of a ligand (P5), modulation (P6), inverse FFT (P7), finding the best solutions (P8), and post processes (P9) as shown in Figure 1.

In the "receptor voxelization (P2)" process, MEGADOCK

Table 1: The profile of docking calculation on 1 CPU core (PDB ID: 1ACB).

	Time (sec.)	Ratio (%)
P1. Initialization	0.04	0.0
P2+P3. Receptor processes	0.30	0.1
P4. Ligand rotation & voxelization	9.19	2.7
P5. Forward FFT of a ligand	155.99	46.4
P6. Modulation	33.97	10.1
P7. Inverse FFT	133.68	39.7
P8. Finding the best solutions	3.27	1.0
P9. Post processes	0.00	0.0
Total	336.45	100.0

assigns score values for each receptor grid based on the atom coordinates of a receptor protein and three types of score functions, real Pairwise Shape Complementarity (rPSC) [22], electrostatics, and desolvation free energy [12]. Processes (P4)–(P8) are looped for n_θ times. MEGADOCK uses n_θ of 3600 in the default setting.

3.2 Profile of MEGADOCK Processes

Table 1 shows proportion of docking calculation time for each process of MEGADOCK. This profile was obtained from the docking calculation for a protein complex (PDB ID: 1ACB, receptor: chain E, 245 residues, ligand: chain I, 70 residues). The FFT size N of the docking calculation is 108, and it is typical in the current protein structure database. The profile was taken on Intel Xeon 2.93 GHz, 1 CPU core.

FFT processes (P5+P7) occupy majority (86.1%) of total time. On the other hand, other calculations such as voxelization and finding the best solutions still consume considerable time portions. It is because the time of FFT calculation has already been reduced by employing a simplified scoring function compared with the other docking software. Thus, even processes other than the FFT calculation must be accelerated for significant speedup. Assume that 30-fold acceleration is achieved in the FFT and modulation processes (P5–P7), then estimated time consumption of these processes will be approximately 10 seconds and the total computation time will be 23 seconds. As a result, the computation time of ligand voxelization (P4, currently 9.19 seconds consumed) will occupy about 40% of total time. Therefore, the mapping of almost all processes, which include ligand voxelization and finding the best solutions, onto GPUs is obviously crucial for achieving effective acceleration.

4. MEGADOCK-GPU

In this study, we developed MEGADOCK-GPU for further acceleration of a protein-protein docking. For the GPGPU implementation, we used CUDA, which is a platform for GPGPU provided by NVIDIA. The system requires CUDA version 5.0 or later because older cuFFT libraries in the previous version of CUDA have problems in the barrier synchronization. We mapped not only the FFT and modulation processes but also the voxelization and finding the best solutions processes onto GPUs. In the previous work by Sukhwani, *et al.*, the targeted system has only a GPU card

Figure 2: Assignment of voxels filled by atoms.

and their implementation could not utilize multiple GPUs. However, current computing system often has multiple CPU cores and multiple GPU cards in a computing node. It is thus important to make full use of such a computing environment, e.g. 12 CPU cores and 3 GPUs. Therefore, we targeted a computing node with multiple CPU cores and multiple GPU cards.

4.1 Implementation on GPUs

We have implemented the following processes on GPUs, forward FFT of a receptor (P3), ligand rotation and voxelization (P4), forward FFT of a ligand (P5), modulation (P6), inverse FFT (P7) and finding the best solutions (P8). The details of each implementation are described in the following.

Ligand Voxelization

MEGADOCK sets adequate rPSC score, electrostatics values and desolvation free energy score on the ligand voxel model in this process. Ligand voxelization is a process that calculates the distance between the coordinates of an atom and each grid and assigns a value to each grid within van der Waals radius of the atom (Figure 2). The assignment process can be parallelized for each atom. Because the rPSC score and desolvation free energy score of a ligand has only binary states (0 or 1) and the electrostatics value of a grid is calculated as accumulative sum of the values of all adjacent atoms, the calculation order for each atom can be freely exchanged. Therefore, we could process atoms in parallel and mapped them onto GPUs. Thus, multiple atoms are simultaneously processed on different GPU cores in this process.

Forward and Inverse FFT

For mapping FFT calculations onto GPUs, we used the NVIDIA cuFFT library [19]. Since cuFFT is optimized for FFT bases $\{2, 3, 5, 7\}$, MEGADOCK-GPU uses FFT size N as a multiple of $\{2, 3, 5, 7\}$. This issue will be discussed again on section 7.3.

Finding the Best Solutions

In this process, the best docking pose is selected according to the docking score. This reduction process is also implemented on GPUs. Current MEGADOCK-GPU only reports a single pose with the best score in a ligand rotation step while original MEGADOCK can report the n -best poses.

However, the other protein-protein docking software, such as ZDOCK and PIPER, does not have such option, and it is not so crucial in a practical case.

Rotation of Ligand

In this process, the atom coordinates of a ligand are updated according to a given rotation matrix. The process for each atom is independent and can be fully parallelized. We mapped them onto GPUs.

Modulation

The modulation can be independent for each grid, thus we mapped them onto GPUs.

4.2 Data Transfer

In the previous work by Sukhwani, *et al*, they implemented the voxelization process on a CPU because the system needs to perform many FFT calculations and it did not become a bottleneck. Thus, a voxelized protein structure data had to be transferred from a host system to GPUs. Previously, we had also performed the voxelization on a CPU, but the data transfer from a host to GPUs became a bottleneck. Indeed, only one FFT calculation is required in our system and the data transfer then occupies large portion of computation time. Thus, we tried to minimize the data transfer. In our implementation, the transfer of large data from a host to GPUs takes place only once. The data includes the original atom coordinates of a ligand and Fourier transformed receptor grid information are transferred at first. In the loop for each ligand rotation angle, only trivial data transfer is required (12 bytes angular information and 8 bytes calculation result) because all processes are performed on GPUs.

4.3 Using Multiple CPU Cores and Multiple GPUs

The latest computing systems tend to have a powerful computing node composed of several multicore CPUs and multiple GPU cards. Thus, we implemented our system to fully use such heterogeneous computing resources. As we mentioned above, the processes for each ligand rotation are parallelized in our system. For utilizing all computing resources, we assign the decomposed jobs to multiple GPUs and CPU cores dynamically using OpenMP. The same number of CPU cores as GPUs is used for controlling GPU processes but the remaining cores perform docking calculation by themselves. Thus, 3 CPU cores are used for controlling 3 GPUs and the remaining 9 CPU cores are used for docking calculation, when we use a system with 12 CPU cores and 3 GPUs. Algorithm 1 shows parallel algorithm of docking calculation.

5. EVALUATION OF PERFORMANCE

5.1 Computation Environment

All the calculations were conducted on the TSUBAME 2.0 supercomputing system, Tokyo Institute of Technology, Japan. We used its thin nodes in all experiments. The specifications of the node are shown in Table 2.

5.2 Datasets

We used 352 protein complex structures retrieved from a standard protein-protein docking benchmark set (ZLAB

Algorithm 1 Parallel algorithm of docking calculation

Input: atom coordinates of receptor R and ligand L , rotational angles of ligand $\{\theta\}$, number of CPU cores C , number of GPUs G

Output: List of high-score docking poses $\mathcal{H}$

- 1: Initialization
- 2: Data transfer to GPU (R, L)
- 3: $R' \leftarrow \text{Voxelization}(R)$
- 4: $\mathcal{F}[R'] \leftarrow \text{FFT}(R')$
- 5: **parallel for** each θ by C threads **do**
- 6: **if** thread number $< G$ **then**
- 7: processed on GPU
- 8: **else**
- 9: processed on CPU
- 10: **end if**
- 11: $L_\theta \leftarrow \text{Rotation}(L, \theta)$
- 12: $L'_\theta \leftarrow \text{Voxelization}(L_\theta)$
- 13: $\mathcal{F}[L'_\theta] \leftarrow \text{FFT}(L'_\theta)$
- 14: $M_\theta \leftarrow \text{Modulation}(\mathcal{F}[R'], \mathcal{F}[L'_\theta])$
- 15: $\text{Score}_\theta \leftarrow \text{IFFT}(M_\theta)$
- 16: $\mathcal{H} \leftarrow \mathcal{H} \cup \text{Max}(\text{Score}_\theta)$
- 17: **end parallel for**
- 18: Post processes

Table 2: Computation Environment

CPU	Intel Xeon 2.93 [GHz] (6 cores)×2
Memory	54 [GB]
OS	SUSE Linux Enterprise Server 11 SP1
GPU	NVIDIA Tesla M2050×3 (3 GPUs / 1 node)
Compiler	Intel C++ Compiler 13.0.0
FFTW	FFTW 3.2.2
CUDA	CUDA 5.0
cuFFT	cuFFT 5.0

Benchmark 4.0) [23], which contains protein structures in both bound and unbound forms. The sizes of the proteins in the dataset are distributed widely and it is a fairly-sampled subset of the current known protein structure complexes.

5.3 Evaluation Method

For evaluating calculation time of each system, we performed same docking calculation for 352 protein pairs three times, and took their average. We used `gettimeofday()` function to measure calculation time. There is no difference between the CPU version of MEGADOCK and the GPU version in the scoring function. Although MEGADOCK-GPU seldom returned different results from those of the CPU version due to the different precision of numeric calculation between a CPU and a GPU, the difference was less than 0.001% and negligibly small.

6. RESULTS

6.1 Comparison of Total Docking Runtime

Table 3 shows the results of total and average docking calculation time for 352 protein complexes. MEGADOCK-GPU using 1 CPU core and 1 GPU was 13.9 times faster than MEGADOCK using 1 CPU core. Also, MEGADOCK-GPU using 12 CPU cores and 3 GPUs was 37.0 times faster than MEGADOCK using 1 CPU core. MEGADOCK has

Table 3: The results of total and averaged docking calculation time for 352 protein complexes. “C” and “G”, respectively, denote number of CPU cores and GPUs.

	1C	1C1G	12C	12C3G
Total (hour)	73.5	5.3	8.2	2.0
Average (sec.)	751.4	54.1	84.1	20.3
vs. 1C	1.0	13.9	8.9	37.0

Figure 3: The distribution of the speedup ratio of MEGADOCK-GPU using 1 CPU core and 1 GPU compared to MEGADOCK using 1 CPU for different FFT size N . Horizontal axis shows FFT size N and vertical axis shows the averaged speedup ratio in protein complexes with same FFT size.

been already parallelized using OpenMP and it achieved 8.9-fold speedup using 12 CPU cores. Furthermore, MEGADOCK-GPU using 12 CPU cores and 3 GPUs was approximately 4.2 times faster than MEGADOCK using 12 CPU cores by fully using computer resources of a node. One may expect that MEGADOCK-GPU using 12 CPU cores and 3 GPUs may achieve not 37.0-fold but at least 41.7-fold ($= 13.9 \times 3$) speedup compared to 1 CPU core, because of using three GPUs. However, the speedup is not fully proportional because the initialization of GPU cannot be parallelized and becomes a bottleneck.

6.2 Distribution of Computation Time for FFT Size

Figure 3 shows the distribution of the speedup ratio of MEGADOCK-GPU with 1 CPU core and 1 GPU compared to MEGADOCK using 1 CPU core, for each FFT size. In the figure, the horizontal axis is FFT size N and the vertical axis is an averaged speedup ratio in the complexes whose FFT size are same. The ratio generally increases in proportion to the size of FFT. Because, the FFT requires $\mathcal{O}(N^3 \log N)$ calculation but the processes that are hard to be mapped onto GPUs basically take only $\mathcal{O}(N)$. Thus, the speedup is not large for small N . Within this experiments, the best speedup ratio was obtained with FFT $N = 192$ and is 33.7-fold against 1 CPU core.

6.3 Speedup on Each Process

Table 4 shows the speedup of docking calculation from MEGADOCK using 1 CPU core to MEGADOCK-GPU using 1 CPU core and 1 GPU, for each process. In the FFT and modulation processes, better acceleration is achieved. Speedups on the ligand voxelization and finding the best solutions processes are moderate because they include reduction processes. However, we succeeded to reduce data transfer drastically by mapping these processes onto GPUs. Thus the GPU implementation of these processes is practically effective. The time of initialization highly increased using a GPU. This is because the startup time is demanded for initializing a GPU and it is difficult to be decreased.

7. DISCUSSION

7.1 Data Transfer Time

In this work, we mapped almost all processes of a protein-protein docking calculation onto GPUs. Also, we succeeded to reduce the amount of data transfer. As results, the time of data transfer in now only 270 milliseconds is approximately 1.3% of total docking time in the case of a docking for PDB ID: 1ACB. This is a large advantage to calculate all processes in a ligand rotation loop on GPUs.

7.2 Initialization of GPU

We achieved to reduce computation time of almost all processes in a protein-protein docking significantly by using GPGPU techniques. As results, the initialization of a GPU has now become one of the bottlenecks. As shown in Table 4, the initialization of a GPU requires approximately 5 seconds and it occupies more than 20% of whole computation time, because we generate a new process and initialize GPU for each pair now. However, in the practical application like a protein-protein interaction network prediction, we have to perform large number of docking calculations for many protein pairs. Therefore, we have only to initialize GPU once if we modify the system to deal many docking calculations for multiple protein pairs in a single computing process. Assume that the time of GPU initialization can be ignorable; the average calculation time of MEGADOCK-GPU using 12 CPU cores and 3 GPUs will be approximately 15 seconds and it is 50-times faster than that of using 1 CPU core.

7.3 Optimization of FFT Size

An FFT-based docking tool firstly reads the atom coordinates of a receptor and a ligand, and determines the grid size fitted for the receptor and ligand. The FFT size N is proportional to the grid size, which was automatically calculated from the single grid unit size and the size of proteins. FFTW algorithms [24], which is used in MEGADOCK, are optimized for sizes that represented as a multiple of $\{2, 3, 5, 7, 11, 13\}$. Thus, our algorithm to decide the grid size searches the smallest composite number consisted of those prime factors. However, cuFFT algorithms [19], which is used in MEGADOCK-GPU, are optimized for sizes that represented as a multiple of $\{2, 3, 5, 7\}$. For the other sizes, slower algorithm is used. Therefore, we should adjust the grid size to optimal one for the cuFFT library. However, in case of using multiple cores and GPUs, MEGADOCK-GPU uses both the FFTW library for a CPU and the cuFFT library for a GPU. The best set of prime numbers consisting

Table 4: Acceleration ratio for each calculation part (PDB ID: 1ACB).

	1 CPU core (sec.)	1 CPU core & 1 GPU (sec.)	Speedup Ratio
P1. Initialization	0.04	5.34	-
P2+P3. Receptor processes	0.30	0.26	1.2
P4. Ligand rotation & voxelization	9.19	3.72	2.5
P5. Forward FFT of a ligand	155.99	5.92	26.4
P6. Modulation	33.97	1.09	31.3
P7. Inverse FFT	133.68	5.92	22.6
P8. Finding the best solutions	3.27	2.58	1.3
P9. Post processes	0.00	0.00	1.0
Total	336.45	24.83	13.5

FFT size N is different between them. We conducted an investigation of a set of prime numbers as a previous experiment. For MEGADOCK-GPU, the best results were obtained on a set $\{2, 3, 5, 7\}$ in both using 1 CPU core and 1 GPU, and using 12 CPU cores and 3 GPUs. For evaluating the performances, we used $\{2, 3, 5, 7\}$ for MEGADOCK-GPU and $\{2, 3, 5, 7, 11, 13\}$ for MEGADOCK CPU version.

8. CONCLUSION

In this work, we developed MEGADOCK-GPU and mapped almost all processes of a protein-protein docking onto GPUs. As a result, the system achieved 13.9-fold acceleration using 1CPU and 1 GPU, and 37.0-fold acceleration using 12 CPU cores and 3 GPUs by making full use of heterogeneous computing resources.

An application to large-scale protein-protein interaction network prediction that needs a lot of docking calculations is one of the important future work. In the case of application to pathway analysis (e.g. human apoptosis pathway [25]) that have many related proteins, the GPU starting time cited in section 6.3 will be hidden effectively.

9. ACKNOWLEDGMENTS

This work was supported in part by NVIDIA CUDA Center of Excellence (CCOE), a Grant-in-Aid for JSPS Fellows (23-8750, 24-8766), a Grant-in-Aid for Research and Development of The Next-Generation Integrated Simulation of Living Matter (ISLiM) and by the Education Academy of Computational Life Sciences (ACLS) at Tokyo Institute of Technology. The authors would like to thank the TSUB-AME supercomputer system at the Global Scientific Information and Computing Center, Tokyo Institute of Technology.

10. REFERENCES

- [1] R. Mosca, A. Céol and P. Aloy, Interactome3D: adding structural details to protein networks, *Nature Methods*, 10(1):47–53, Jan. 2013.
- [2] I. A. Vakser, Low-resolution structural modeling of protein interactome, *Curr. Opin. Struct. Biol.*, 23(2):198–205, Apr. 2013.
- [3] M. N. Wass, A. David and M. J. E. Sternberg, Challenges for the prediction of macromolecular interactions, *Curr. Opin. Struct. Biol.*, 21(3):382–390, Jun. 2011.
- [4] A. Stein, R. Mosca and P. Aloy, Three-dimensional modeling of protein interactions and complexes is going 'omics, *Curr. Opin. Struct. Biol.*, 21(2): 200–208, Apr. 2011.
- [5] Y. Matsuzaki, Y. Matsuzaki, T. Sato and Y. Akiyama, *In silico* screening of protein-protein interactions with all-to-all rigid docking and clustering: an application to pathway analysis, *J. Bioinform. Comput. Biol.*, 7(6): 991–1012, Dec. 2009.
- [6] M. Ohue, Y. Matsuzaki, N. Uchikoga, T. Ishida and Y. Akiyama, MEGADOCK: an all-to-all protein-protein interaction prediction system using tertiary structure data, *Prot. Pept. Lett.* (in press).
- [7] M. N. Wass, G. Fuentes, C. Pons, F. Pazos and A. Valencia, Towards the prediction of protein interaction partners using physical docking, *Mol. Syst. Biol.*, 7:469, Feb. 2011.
- [8] J. Mintseris, B. Pierce, K. Wiehe, R. Anderson, R. Chen and Z. Weng, Integrating statistical pair potentials into protein complex prediction, *Proteins*, 69(3):511–520, Nov. 2007.
- [9] B. G. Pierce, Y. Hourai and Z. Weng, Accelerating protein docking in ZDOCK using an advanced 3D convolution library, *PLoS ONE*, 6(9):e24657, Sep. 2011.
- [10] K. Tharakaraman, L. N. Robinson, A. Hatas, Y.-L. Chen, L. Siyue, S. Raguram, V. Sasisekharan, G. N. Wogan and R. Sasisekharan, Redesign of a cross-reactive antibody to dengue virus with broad-spectrum activity and increased in vivo potency, *Proc. Natl. Acad. Sci. USA.*, 110(7):E1555–1564, Apr. 2013.
- [11] H. Hwang, T. Vreven, B. G. Pierce, J.-H. Hung and Z. Weng, Performance of ZDOCK and ZRANK in CAPRI rounds 13–19, *Proteins*, 78(15):3104–3110, Nov. 2010.
- [12] M. Ohue, Y. Matsuzaki, T. Ishida and Y. Akiyama, Improvement of the protein-protein docking prediction by introducing a simple hydrophobic interaction model : an application to interaction pathway analysis, *Lecture Notes in Bioinformatics*, 7632:178–187, Nov. 2012.
- [13] NVIDIA CUDA - NVIDIA developer zone, <http://developer.nvidia.com/cuda/>.
- [14] S. Suzuki, T. Ishida, K. Kurokawa and Y. Akiyama, GHOSTM: a GPU-accelerated homology search tool for metagenomics, *PLoS ONE*, 7(5):e36060, Jan. 2012.
- [15] J. A. van Meel, A. Arnold, D. Frenkel, S. F. Portegies Zwart and R. G. Belleman, Harvesting graphics power for MD simulations, *Mol. Simul.*, 34(3):259–266, May

2008.

- [16] I. S. Ufimtsev and T. J. Martínez, Quantum chemistry on graphical processor units. 1. Strategies for two-electron integral evaluation. *J. Chem. Theory Comput.*, 4(2):222–231, Jan. 2008.
- [17] B. Sukhwani and M. C. Herborgt, GPU acceleration of a production molecular docking code, In *Proc. of 2nd Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-2)*, 19–27, Mar. 2009.
- [18] D. Kozakov, R. Brenke, S. R. Comeau and S. Vajda, PIPER: an FFT-based protein docking program with pairwise potentials, *Proteins*, 65(2):392–406, Nov. 2006.
- [19] CUFFT - NVIDIA developer zone, <http://developer.nvidia.com/cufft/>.
- [20] D. Kozakov, D. R. Hall, D. Beglov, R. Brenke, S. R. Comeau, Y. Shen, K. Li, J. Zheng, P. Vakili, I. C. Paschalidis and S. Vajda, Achieving reliability and high accuracy in automated protein docking: ClusPro, PIPER, SDU, and stability analysis in CAPRI rounds 13-19, *Proteins*, 78(15):3124–3130, Nov. 2010.
- [21] E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. A. Friesem, C. Aflalo and I. A. Vakser, Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques, *Proc. Natl. Acad. Sci. USA.*, 89(6):2195–2199, Mar. 1992.
- [22] M. Ohue, Y. Matsuzaki and Y. Akiyama, Docking-calculation-based method for predicting protein-RNA interactions *Genome Inform.*, 25(1):25–39, Aug. 2011.
- [23] H. Hwang, T. Vreven, J. Janin and Z. Weng, Protein-protein docking benchmark version 4.0, *Proteins*, 78(15):3111–3114, Nov. 2010.
- [24] M. Frigo and S. G. Johnson, The design and implementation of FFTW3, In *Proc. of IEEE*, 93(2):216–231, Mar. 2005.
- [25] M. Ohue, Y. Matsuzaki, T. Shimoda, T. Ishida and Y. Akiyama, Highly precise protein-protein interaction prediction based on consensus between template-based and *de novo* docking methods, In *Proc. of 8th Great Lakes Bioinformatics Conference (GLBIO2013)*, 100–109, May. 2013.