論文 / 著書情報
Article / Book Information

Title	MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers
Authors	Ohue, M., Shimoda, T., Suzuki, S., Matsuzaki, Y., Ishida, T., Akiyama, Y.
Citation	Bioinformatics, Vol. 30, No. 22, pp. 3281-3283
Pub. date	2014, 8
Note	This is a pre-copyedited, author-produced PDF of an article accepted for publication in [Bioinformatics] following peer review. The version of record [Bioinformatics, Vol. 30, No. 22, pp. 3281-3283] is available online at:http://bioinformatics.oxfordjournals.org/content/30/22/3281.full.pdf+html?maxtoshow=&hits=10&RESULTFORMAT=&fulltext=MEGADOCK+4.0%253A+an+ultra%25E2%2580%2593high-performance+protein%25E2%2580%2593protein+docking+software+for+heterogeneous+supercomputersSearch+articles...&searchid=1&FIRSTINDEX=0&resourcetype=HWCIT

Structural bioinformatics

MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers

Masahito Ohue[1,2†], Takehiro Shimoda[1,3†], Shuji Suzuki[1,2,3], Yuri Matsuzaki[3], Takashi Ishida[1], and Yutaka Akiyama[1,3*]

[1]Department of Computer Science, Graduate of Information Science and Engineering, Tokyo Institute of Technology, 2-12-1 W8-76, Ookayama, Meguro-ku, Tokyo 152-8550, Japan; [2]Japan Society for the Promotion of Science (JSPS) Research Fellow; [3]Education Academy of Computational Life Sciences (ACLS), Tokyo Institute of Technology, 2-12-1 W8-93, Ookayama, Meguro-ku, Tokyo 152-8550, Japan,

ABSTRACT

Summary: The application of protein–protein docking in large-scale interactome analysis is a major challenge in structural bioinformatics and requires huge computing resources. In this work, we present MEGADOCK 4.0, an FFT-based docking software that makes extensive use of recent heterogeneous supercomputers and shows powerful, scalable performance of over 97% strong scaling.
Availability and Implementation: MEGADOCK 4.0 is written in C++ with OpenMPI and NVIDIA CUDA 5.0 (or later) and is freely available to all academic and non-profit users at: http://www.bi.cs.titech.ac.jp/megadock.
Contact: akiyama@cs.titech.ac.jp
Supplementary Information: Supplemental information is available at the journal's website.

1 INTRODUCTION

Protein–protein interactions can provide valuable insights for understanding the principles of biological systems and for elucidating causes of incurable diseases. Although many structures of interacting proteins have been determined by X-ray crystallography and Nuclear Magnetic Resonance (NMR) spectroscopy, the structures of many protein complexes have still not been determined experimentally due to cost and technical limitations. Protein–protein docking, a computational method for predicting the structure of a protein complex from known component structures, is a powerful approach that facilitates the discovery of otherwise unattainable protein complex structures.

A number of fast Fourier transform (FFT)-based rigid-body initial protein-protein docking tools have been developed for predicting protein complex structures (Cheng *et al*., 2007; Ritchie and Venkatraman, 2010; Pierce *et al*., 2011). However, faster docking tools are still required in order to perform large-scale interactome predictions. Some applications also require a huge number of dockings, such as ensemble docking techniques using multiple conformations for flexible docking (Grünberg *et al*., 2004; Król *et al*., 2007), cross-docking for identification of protein interaction partners (Matsuzaki *et al*., 2009; Wass *et al*., 2011; Lopes *et al*., 2013; Zhang *et al*., 2014), and multiple docking (Karaca and Bonvin, 2011). To achieve these large-scale analyses, use of the supercomputing environment has become absolutely necessary.

On the other hand, 35% of computing performance of supercomputers ranked in top500.org (June 2014) is currently achieved by hardware accelerators, such as graphics processing units (GPUs), and this percentage is increasing. Therefore, tools that can be used with such "heterogeneous" supercomputers are necessary. While some docking tools are accelerated by GPUs on a node (Sukhwani and Herbordt, 2009; Ritchie and Venkatraman, 2010), "heterogeneous" supercomputers, which have massive numbers of nodes including multiple CPU cores and GPU cards, have not yet been used for acceleration of docking tool performance.

Here, we present ultra–high-performance docking software, "MEGADOCK 4.0," which makes extensive use of supercomputers equipped with GPUs.

2 IMPLEMENTATION

2.1 MEGADOCK scheme

MEGADOCK uses a Katchalski-Katzir algorithm known as a traditional FFT-based rigid-docking scheme (Katchalski-Katzir *et al*., 1992). Its original scoring function, based on shape complementarity, electrostatics, and desolvation free energy, is calculated by only one correlation function (Ohue *et al*., 2012; Ohue *et al*., 2014). This is advantageous for faster calculation because multiple correlation functions and thus multiple FFT calculations are used to evaluate multiple effects in previous methods (Kozakov *et al*., 2006; Pierce *et al*., 2011). (see Supplementary Text S1 for details)

2.2 GPU implementation

MEGADOCK has been implemented on multiple GPUs using the CUDA library (Shimoda *et al*., 2013). A previous study (Sukhwani and Herbordt, 2009) mapped only FFT processes onto a GPU, and its implementation could not utilize multiple GPUs. We mapped the whole docking process (voxelization, ligand rotation, FFTs, and finding solutions) onto GPUs, and our implementation was able to utilize multiple GPUs and CPU cores (Shimoda *et al*., 2013).

2.3 Hybrid CUDA, MPI, and OpenMP parallelization

For extensive execution of docking jobs, an implementation that can be performed among many computing nodes is required. We previously parallelized the calculation of each docking processes using MPI and OpenMP with the master/worker model (Matsuzaki *et al*., 2013). On cluster computers, a master process acquires a list of protein pairs and distributes the docking jobs to worker processes on available nodes. This implementation guarantees fault tolerance in that the master process surveys all docking jobs.

[†]These authors contributed equally to this work.

The proposed software, MEGADOCK 4.0, is implemented by hybrid CUDA, MPI, and OpenMP parallelization. Reducing the usage of memory space is important with systems that have many CPU cores, multiple GPUs per node, and relatively little memory (e.g., there is only 6 GB memory on an NVIDIA Tesla K20X GPU). We assigned one docking job to each node and then distributed the calculations of ligand rotation by thread parallelization with CPU cores and GPUs. This implementation model manages one node as the master and the other nodes as workers. The master node distributes the docking jobs to worker nodes and a worker node executes distributed docking jobs with multiple GPUs by CUDA and all CPU cores by OpenMP thread parallelization. This implementation also guarantees fault tolerance similar to the CPU version.

3 RESULTS AND DISCUSSION

In order to check the performance of MEGADOCK 4.0, we used the ZLAB benchmark 4.0 dataset (Hwang *et al*., 2010). Speed measurement experiments were conducted on the TSUBAME 2.5 supercomputing system (Tokyo Institute of Technology, Japan). We used its "thin nodes" with a reservation service of exclusive use (up to 420 nodes). Each "thin" node contained two Intel Xeon X5670 (six cores, 2.93 GHz) and three NVIDIA Tesla K20X (GK110) GPUs. The specifications of the environment are shown in Supplementary Text S2 and Table S1.

Figure 1 shows the average of five measurements of computation time and the parallel scalability of MEGADOCK 4.0 on 30,976 protein pairs from combinations between 176 receptors and 176 ligands, assuming a cross-docking study. The observed calculation acceleration was close to ideal. Strong scaling values from 35 nodes were over 97% for all numbers of nodes measured here (Supplementary Table S2). Notably, a high scalability (98%) was obtained with the largest number of nodes (420 nodes).

We also measured docking time on a half million and a million protein pairs for simulation of large-scale interactome analyses using averaged-sized proteins (FFT size of 108, see Supplementary Table S3). In this simulation, a half million docking jobs required 5.71 hours while a million jobs required 11.51 hours. The epidermal growth factor receptor (EGFR)-related pathway, which we are studying in non-small cell lung cancer, required approximately a quarter million dockings. This analysis could be completed in only 3 hours with MEGADOCK 4.0 using 420 nodes, whereas solving the same problem requires several days with an older version of MEGADOCK.

4 CONCLUSIONS

MEGADOCK 4.0 is a docking software for heterogeneous supercomputing environments and shows excellent scalability. Heterogeneous supercomputers equipped with hardware accelerators, such GPUs, will become common in the future. Fully utilizing such computers is crucial for bioinformatics research, which must analyze massive amounts of data. MEGADOCK 4.0 can serve as a tool to promote analysis of the whole interactome within a reasonable time.

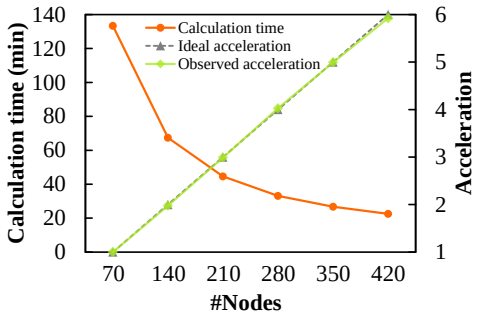

Fig. 1. Calculation time and acceleration by parallelization among nodes on 30,976 docking jobs.

ACKNOWLEDGEMENTS

This research used computational resources of the TSUBAME 2.5 supercomputer provided by Tokyo Institute of Technology through the HPCI System Research project (Project ID: hp140173).
Funding: this work was supported by a Grant-in-Aid for JSPS Fellows (238750 and 2630002) from the Ministry of Education, Culture, Sports, Science, and Technology of Japan (MEXT).

REFERENCES

Cheng,T.M.-K. *et al.* (2007) pyDock: electrostatics and desolvation for effective scoring of rigid-body protein–protein docking. *Proteins*, 68, 503-515.
Grünberg,R. *et al.* (2004) Complementarity of structure ensembles in protein-protein binding. *Structure*, 12, 2125-2136.
Hwang,H. *et al.* (2010) Protein–protein docking benchmark version 4.0. *Proteins*, 78, 3111-3114.
Karaca,E. and Bonvin,A.M.J.J. (2011) A multidomain flexible docking approach to deal with large conformational changes in the modeling of biomolecular complexes. *Structure*, 19, 555-565.
Katchalski-Katzir,E. *et al.* (1992) Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proc. Natl. Acad. Sci. U. S. A.*, 89, 2195-2199.
Kozakov,D. *et al.* (2006) PIPER: an FFT-based protein docking program with pairwise potentials. *Proteins*, 65, 392-406.
Król,M. *et al.* (2007) Flexible relaxation of rigid-body docking solutions. *Proteins*, 68, 159-169.
Lopes,A. *et al.* (2013) Protein–protein interactions in a crowded environment: an analysis via cross-docking simulations and evolutionary information. *PLOS Comput. Biol.*, 9, e1003369.
Matsuzaki,Y. *et al.* (2009) *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, 991-1012.
Matsuzaki,Y. *et al.* (2013) MEGADOCK 3.0: a high-performance protein–protein interaction prediction software using hybrid parallel computing for petascale supercomputing environments. *Source Code Biol. Med.*, 8, 18.
Ohue,M. *et al.* (2012) Improvement of the protein–protein docking prediction by introducing a simple hydrophobic interaction model : an application to interaction pathway analysis. *Lect. Notes Comput. Sci.*, 7632, 178-187.
Ohue,M. *et al.* (2014) MEGADOCK: An all-to-all protein–protein interaction prediction system using tertiary structure data. *Protein Pept. Lett.*. 21, 766-778.
Pierce,B.G. *et al.* (2011) Accelerating protein docking in ZDOCK using an advanced 3D convolution library. *PLOS ONE*, 6, e24657.
Ritchie,D.W. and Venkatraman,V. (2010) Ultra-fast FFT protein docking on graphics processors. *Bioinformatics*, 26, 2398-2405.
Shimoda,T. *et al.* (2013) MEGADOCK-GPU: Acceleration of protein–protein docking calculation on GPUs. In, *Proc. ACM-BCB'13*. ACM Press, NY, pp. 883-889.
Sukhwani,B. and Herbordt,M.C. (2009) GPU acceleration of a production molecular docking code. In, *Proc. GPGPU-2*. ACM Press, NY, pp. 19-27.
Wass,M.N. *et al.* (2011) Towards the prediction of protein interaction partners using physical docking. *Mol. Syst. Biol.*, 7, 469.
Zhang,C. *et al.* (2014) Discovery of binding proteins for a protein target using protein–protein docking-based virtual screening. *Proteins*. (early access).

Supplementary Information

MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers

Masahito Ohue[1,2†], Takehiro Shimoda[1,3†], Shuji Suzuki[1,2,3], Yuri Matsuzaki[3], Takashi Ishida[1], and Yutaka Akiyama[1,3*]

[1]Department of Computer Science, Graduate of Information Science and Engineering, Tokyo Institute of Technology, 2-12-1 W8-76, Ookayama, Meguro-ku, Tokyo 152-8550, Japan

[2]Japan Society for the Promotion of Science (JSPS) Research Fellow

[3]Education Academy of Computational Life Sciences (ACLS), Tokyo Institute of Technology, 2-12-1 W8-93, Ookayama, Meguro-ku, Tokyo 152-8550, Japan

[†]These authors contributed equally to this work

[*]Corresponding author (akiyama@cs.titech.ac.jp)

Contents

Supplementary Material 1 — Text S1

Scoring function with correlation functions on protein docking

One of the major docking methods is the 3-D grid-based docking technique with the fast Fourier transform (FFT) correlation approach [1]. In this method, also used on MEGADOCK 4.0, the protein structure is projected onto a 3-D grid, and the scoring function is calculated by discrete Fourier transform (DFT) and inverse discrete Fourier transform (IDFT) using the correlation of two discrete functions (protein grids), as follows:

$$\begin{aligned} S(\alpha,\beta,\gamma) &= \sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{n=1}^{N} R(l,m,n)L(l+\alpha,m+\beta,n+\gamma) \\ &= \mathrm{IDFT}[\mathrm{DFT}[R(l,m,n)]^{*}\ \mathrm{DFT}[L(l,m,n)]] \end{aligned}$$

where R and L are the discrete function of the receptor (R) and ligand (L) proteins, respectively, (l, m, n) is a coordinate in the 3-D grid space, and (α, β, γ) is the parallel translation vector of the ligand protein. The asterisk operator $*$ indicates the complex conjugate of a complex number. To directly execute the simple convolution sums in $S(\alpha, \beta, \gamma)$, $O(N^6)$ calculations are required; however, this is reduced to $O(N^3 \log N)$ using the FFT.

The discrete functions R and L usually take into account multiple effects, such as shape complementarity, electrostatic interaction, desolvation free energy, and so on (e.g. ZDOCK [2], PIPER [3], and SDOCK [4]). The total scoring function is the weighted sum of the partial scoring functions, according to the following example:

$$\begin{aligned} S_{\mathrm{total}}(\alpha,\beta,\gamma) &= w_{\mathrm{shape}}S_{\mathrm{shape}} + w_{\mathrm{elec}}S_{\mathrm{elec}} + w_{\mathrm{desol}}S_{\mathrm{desol}} \\ S_{\mathrm{shape}}(\alpha,\beta,\gamma) &= \mathrm{IDFT}[\mathrm{DFT}[R_{\mathrm{shape}}(l,m,n)]^{*}\ \mathrm{DFT}[L_{\mathrm{shape}}(l,m,n)]] \\ S_{\mathrm{elec}}(\alpha,\beta,\gamma) &= \mathrm{IDFT}[\mathrm{DFT}[R_{\mathrm{elec}}(l,m,n)]^{*}\ \mathrm{DFT}[L_{\mathrm{elec}}(l,m,n)]] \\ S_{\mathrm{desol}}(\alpha,\beta,\gamma) &= \mathrm{IDFT}[\mathrm{DFT}[R_{\mathrm{desol}}(l,m,n)]^{*}\ \mathrm{DFT}[L_{\mathrm{desol}}(l,m,n)] \end{aligned}$$

In this example, the total scoring function is calculated based on three correlation functions[1].

On the other hand, MEGADOCK requires only one correlation function, despite that the total scoring function take into account shape complementarity, electrostatic interaction, and desolvation free energy, like ZDOCK. We developed the original shape complementarity function (S_{rPSC}) and desolvation free energy function (S_{RDE}). The total scoring function of MEGADOCK is represented by these functions as follows:

$$\begin{aligned} S_{\mathrm{total}}(\alpha,\beta,\gamma) &= \Re\left[\ \mathrm{IDFT}[\mathrm{DFT}[R(l,m,n)]^{*}\ \mathrm{DFT}[L(l,m,n)]]\right] \\ R(l,m,n) &= R_{\mathrm{rPSC}}(l,m,n) + w_{\mathrm{RDE}}R_{\mathrm{RDE}}(l,m,n) + iR_{\mathrm{elec}}(l,m,n) \\ L(l,m,n) &= L_{\mathrm{rPSC\&RDE}}(l,m,n) - iw_{\mathrm{elec}}L_{\mathrm{elec}}(l,m,n) \end{aligned}$$

where S_{total} consisted of only one correlation function. More details of the partial functions are described in previous reports [5, 6].

[1] In actuality, the desolvation free energy function S_{desol} also often comprises multiple correlation functions, e.g., ZDOCK uses six correlation functions and PIPER uses nine correlation functions for the calculation of S_{desol}.

Supplementary Material 2 — Text S2 and Table S1

Hardware specifications of the TSUBAME 2.5 supercomputer

The TSUBAME 2.5 supercomputer is an advanced high-performance computing resource provided by the Global Scientific Information and Computing Center (GSIC) at the Tokyo Institute of Technology (http://tsubame.gsic.titech.ac.jp/en). Currently, TSUBAME 2.5 provides production services for which peak performance reaches 5.7 petaflops in double precision and 17 petaflops in single precision by aggressively exploiting the higher performance per watt. This is the machine achieving petaflop speed in Japan, and it ranks 13th in the top500 list and 8th in the green500 list announced at the International Supercomputing Conference 2014 (http://www.top500.org and http://www.green500.org, June 2014).

TSUBAME 2.5 consists of more than 1,400 compute nodes interconnected by high-bandwidth full-bisection Infiniband networks. There are three node types: *Thin*, *Medium*, and *Fat* nodes, which differ in their equipped memory capacity among, other specifications. All compute nodes share scalable storage systems that provide 7 PB of capacity. In this study, we used *Thin* nodes; hardware specifications are shown in Supplementary Table S1.

Supplementary Table S1

Hardware specifications of TSUBAME 2.5 *Thin* nodes

CPU	Intel Xeon X5670 (2.93 GHz) (6 cores) × 2
Memory	54 GB
OS	SUSE Linux Enterprise Server 11 SP1
GPU	NVIDIA Tesla K20X (GK110) × 3
GPU Memory	6 GB / GPU
Compiler	Intel C++ Compiler 14.0.2.144
FFT library (CPU)	FFTW 3.2.2
CUDA	CUDA 5.5
FFT library (GPU)	cuFFT 5.5

Supplementary Material 3 — Table S2

Supplementary Table S2

Benchmarking results (strong scaling) of 30,976 docking jobs

#Nodes n	35	70	105	140	210	280	350	420
#CPU cores	420	840	1,260	1,680	2,520	3,360	4,200	5,040
#GPUs	105	210	315	420	630	840	1,050	1,260
Time T_n (min)	264.4	133.3	90.6	67.4	44.6	33.1	26.7	22.5
Strong Scaling$_n$ [a]	-	0.991	0.973	0.981	0.988	0.997	0.990	0.980

[a] Strong scaling value from 35 nodes (Strong Scaling$_n$ = (T_{35} / T_n) / (n / 35)).

Supplementary Material 4 — Table S3

Supplementary Table S3

The average-sized protein (FFT size $N = 108$) dataset described in the main text. This dataset was used in the computational experiment of a million docking calculations (Section 3 in the main text). These protein structures are available in the ZLAB protein–protein docking benchmark 4.0 (http://zlab.umassmed.edu/benchmark) and Protein Data Bank (PDB) (http://www.rcsb.org).

PDB ID:Chain	ZLAB Benchmark Code	#Residues
1ATN:A	1ATN_r	372
1AZS:AB	1AZS_r	353
1BGX:HL	1BGX_r	423
1BJ1:VW	1BJ1_l	189
1BUH:A	1BUH_r	294
1BVN:P	1BVN_r	495
1F34:B	1F34_l	127
1FC2:D	1FC2_l	414
1FQ1:B	1FQ1_l	292
1FQJ:A	1FQJ_r	317
1GXD:A	1GXD_l	182
1I9R:HL	1I9R_r	438
1JK9:A	1JK9_r	221
1K4C:C	1K4C_l	394
1K5D:AB	1K5D_r	339
1K74:DR	1K74_l	284
1KXP:A	1KXP_r	372
1N2C:EF	1N2C_l	575
1OC0:A	1OC0_r	373
1RLB:ABCD	1RLB_r	456
1XU1:ABD	1XU1_r	404
1YVB:A	1YVB_r	241
1ZHH:B	1ZHH_l	210
2A5T:B	2A5T_l	278
2G77:A	2G77_r	322
2OT3:B	2OT3_r	245
2Z0E:A	2Z0E_r	319

Supplementary Material 5 — Text S3

Installation of MEGADOCK 4.0

Requirements:

- FFTW3 - http://www.fftw.org
  `--enable-float` flag must be specified when you compile FFTW3
- OpenMPI - http://www.open-mpi.org (use MPI)
- CUDA Toolkit ver. ≥ 5.0 - https://developer.nvidia.com/cuda-zone (use GPU)
- GPU Computing SDK code samples (same version as CUDA Toolkit) - https://developer.nvidia.com/cuda-zone (use GPU)

Installation:

You can use MEGADOCK 4.0 on (a) *GPU cluster*, (b) *CPU cluster*, (c) *GPU single node*, and (d) *CPU single node*. Please see the appropriate instructions below.

(a) *GPU cluster* (GPU, MPI, & OpenMP hybrid parallelization)

Extract tarball contents

```
$ tar xzf megadock-4.0.tgz
$ cd megadock-4.0
```

Edit Makefile

```
CUDA_INSTALL_PATH ?= your/cuda/toolkit/install/path
CUDA_SAMPLES_PATH ?= your/cuda/sdk/install/path
FFTW_INSTALL_PATH ?= your/fftw/library/install/path
CPPCOMPILER       ?= icpc, g++ or others
MPICOMPILER       ?= mpicxx or others
OPTIMIZATION      ?= -O3
OMPFLAG           ?= -openmp (intel) or -fopenmp (g++)
```

Compile

```
$ make
```

A binary file `megadock-gpu-dp` will be generated.

(b) *CPU cluster* (MPI & OpenMP hybrid parallelization)

Extract tarball contents

```
$ tar xzf megadock-4.0.tgz
$ cd megadock-4.0
```

Edit Makefile

```
FFTW_INSTALL_PATH ?= your/fftw/library/install/path
CPPCOMPILER       ?= icpc, g++ or others
MPICOMPILER       ?= mpicxx or others
```

```
OPTIMIZATION      ?= -O3
OMPFLAG           ?= -openmp (intel) or -fopenmp (g++)
USE_GPU := 0
```

Compile

```
$ make
```

A binary file `megadock-dp` will be generated.

(c) *GPU single node* (GPU parallelization)

Extract tarball contents

```
$ tar xzf megadock-4.0.tgz
$ cd megadock-4.0
```

Edit Makefile

```
CUDA_INSTALL_PATH ?= your/cuda/toolkit/install/path
CUDA_SAMPLES_PATH ?= your/cuda/sdk/install/path
FFTW_INSTALL_PATH ?= your/fftw/library/install/path
CPPCOMPILER       ?= icpc, g++ or others
OPTIMIZATION      ?= -O3
OMPFLAG           ?= -openmp (intel) or -fopenmp (g++)
USE_MPI := 0
```

Compile

```
$ make
```

A binary file `megadock-gpu` will be generated.

(d) *CPU single node* (OpenMP thread parallelization)

Extract tarball contents

```
$ tar xzf megadock-4.0.tgz
$ cd megadock-4.0
```

Edit Makefile

```
FFTW_INSTALL_PATH ?= your/fftw/library/install/path
CPPCOMPILER       ?= icpc, g++ or others
OPTIMIZATION      ?= -O3
OMPFLAG           ?= -openmp (intel) or -fopenmp (g++)
USE_MPI := 0
USE_GPU := 0
```

Compile

```
$ make
```

A binary file `megadock` will be generated.

References

[1] Katchalski-Katzir E, Shariv I, Eisenstein M, Friesem AA, Aflalo C and Vakser IA (1992). Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, **89**(6): 2195–2199.

[2] Mintseris J, Pierce B, Wiehe K, Anderson R, Chen R and Weng Z (2007). Integrating statistical pair potentials into protein complex prediction. *Proteins*, **69**(3): 511–520.

[3] Kozakov D, Brenke R, Comeau SR and Vajda S (2006). PIPER: an FFT-based protein docking program with pairwise potentials. *Proteins*, **65**(2): 392–406.

[4] Zhang C and Lai L (2011). SDOCK: a global protein-protein docking program using stepwise force-field potentials. *Journal of Computational Chemistry*, **32**(12): 2598–2612.

[5] Ohue M, Matsuzaki Y, Ishida T and Akiyama Y (2012). Improvement of the protein–protein docking prediction by introducing a simple hydrophobic interaction model: an application to interaction pathway analysis. *Lecture Notes in Computer Science*, **7632**: 178–187.

[6] Ohue M (2014). Protein–protein interaction network prediction based on tertiary structure data. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology.