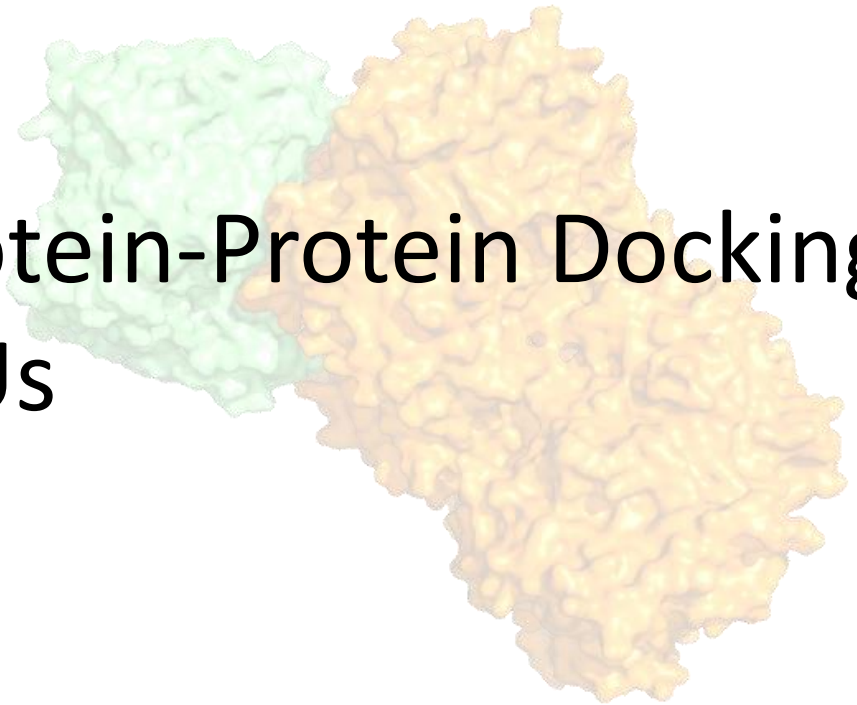
MEGADOCK-GPU: Acceleration of Protein-Protein Docking Calculation on GPUs



Takehiro Shimoda, Takashi Ishida, Shuji Suzuki, Masahito Ohue, Yutaka Akiyama

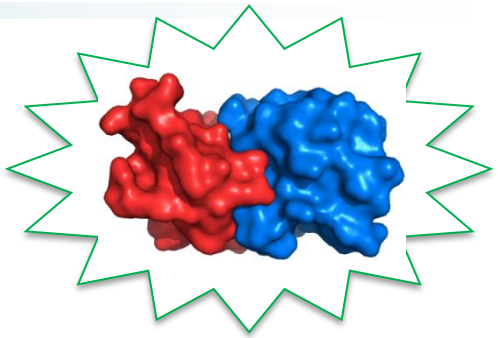
Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

TOKYO TECH Pursuing Excellence

Outline

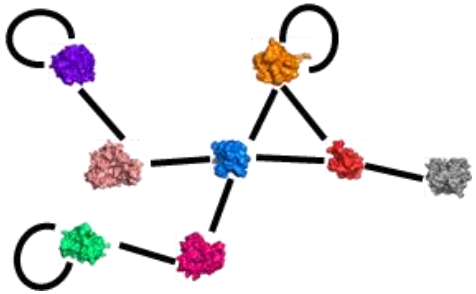
- Background
- MEGADOCK-GPU
- Evaluation of Performance
- Conclusion

Protein-Protein Interaction Network



- Protein-protein interaction (PPI)
  - Proteins interact with each other and make interaction network

- PPI network
  - Important for understanding of cell behaviors
  - Needs a lot of wet experiments
    - Computational prediction method is required



M. N. Wass, *et al.*, *Mol. Syst. Biol.*, 2011.
Y. Matsuzaki, *et al.*, *J. Bioinform. Comput. Biol.*, 2009.

Protein-Protein Interaction Prediction

- Computational PPI prediction method
  - Sequence based method
  - Domain-domain interaction based method
  - Structure based method

J. Shen, *et al.*, *PNAS,* 2007.
Y. Guo, *et al.*, *BMC Research Notes*, 2010.

M. Deng, *et al.*, *Genome Research,* 2002.

- Structure based method
  - Molecular Dynamics (MD)
    - High-definition simulation but very slow
  - Rigid body protein-protein docking
    - Fast but low-definition calculation

Protein-Protein Docking Software

- Protein-protein docking software
  - Non-FFT-based
    - PATCHDOCK D. Duhovny, *et al.*, *Lecture Notes in Computer Science*, 2002.
      - Geometric hashing
  - FFT-based
    - ZDOCK J. Mintseris, *et al.*, *Proteins*, 2007.
      - High precision docking
      - Widely used
    - PIPER D. Kozakov, *et al.*, *Proteins*, 2006.
    - MEGADOCK

MEGADOCK

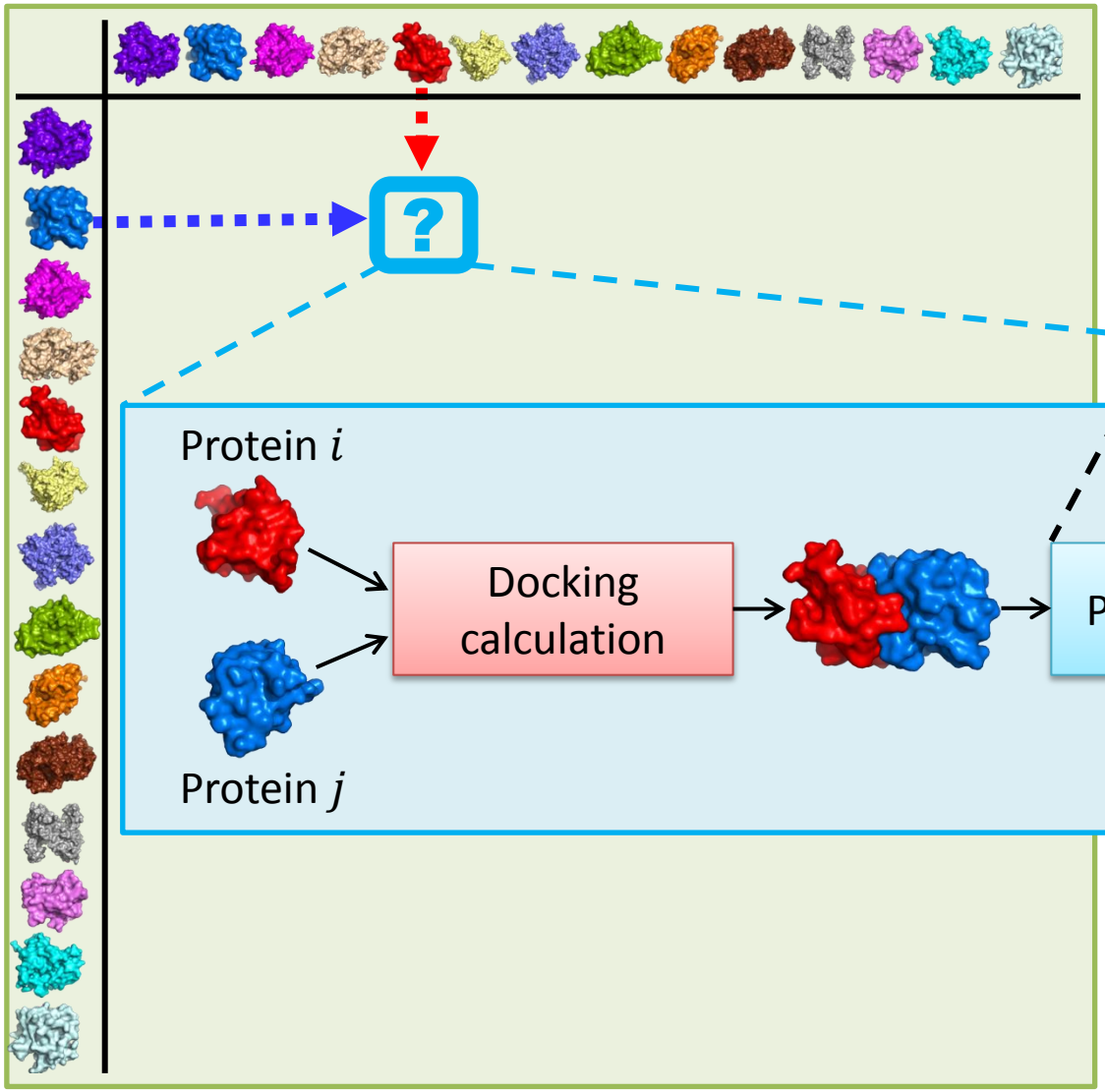
MEGADOCK

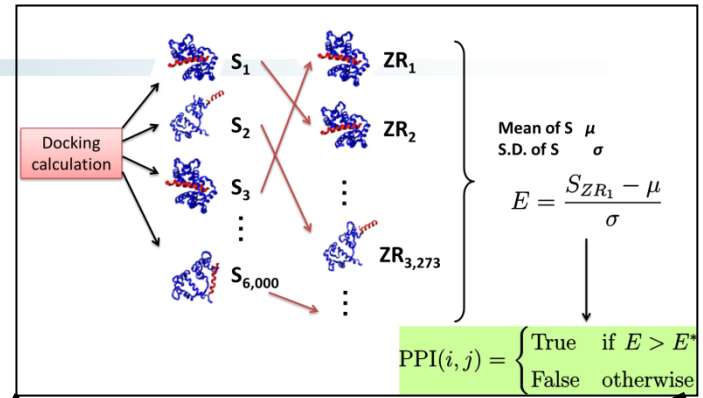
M. Ohue, *et al*. *Protein & Peptide Letters*. (in press)

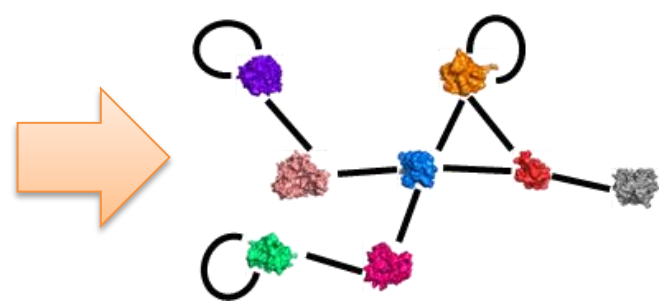
- Protein-protein interaction prediction system
  - For large-scale PPI network
  - Using protein-protein docking

- Features
  - FFT-based
  - Fast
  - Open source

PPI Network Prediction Based on Protein-Proteindocking



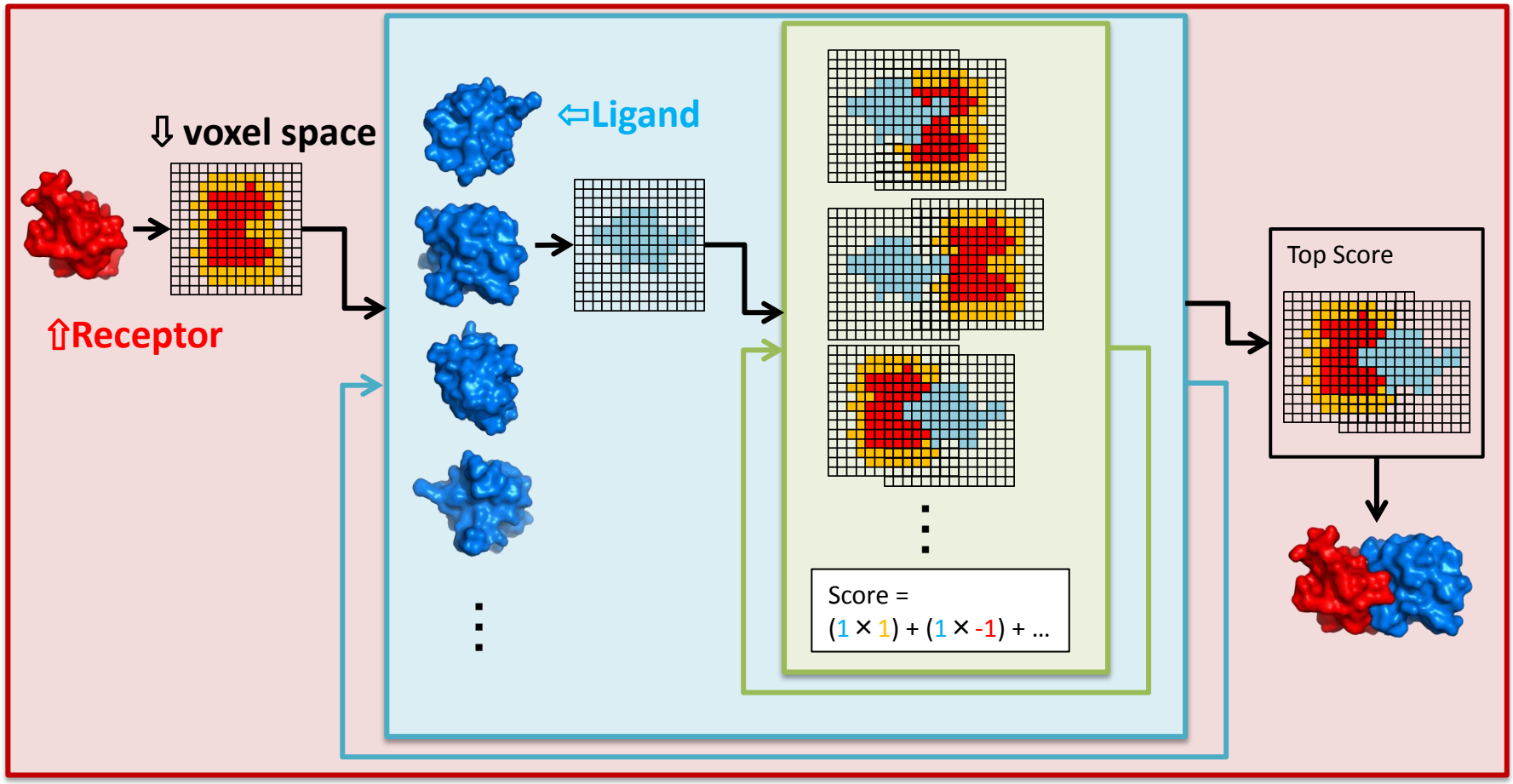




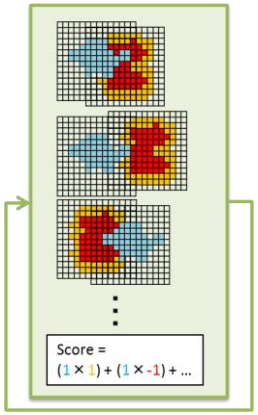
Docking Calculation Algorithm

- Flow of docking calculation
  - Using voxel space

Katchalski-Katzir E, *et al*. *PNAS*, 1992.

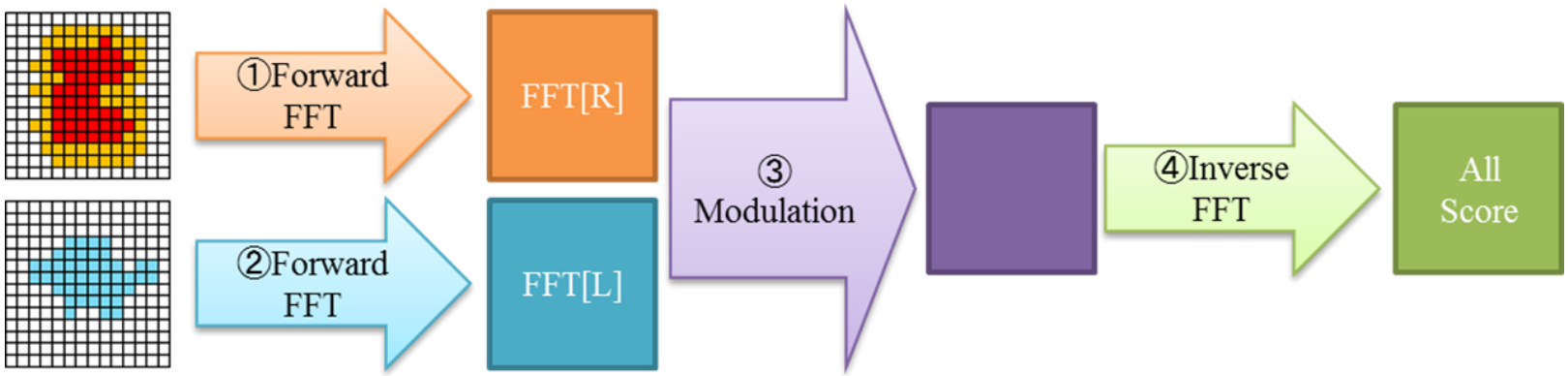


Fast Docking Calculation Using FFT



- Bottleneck: Score calculation
  - 3-D product & 3-D overlap pattern ⇨ $O(N^6)$
    - N is voxel size (about 100 to 300)

- Fast Fourier Transform (FFT)
  - FFT reduces computational complexity ⇨ $O(N^3 \log N)$

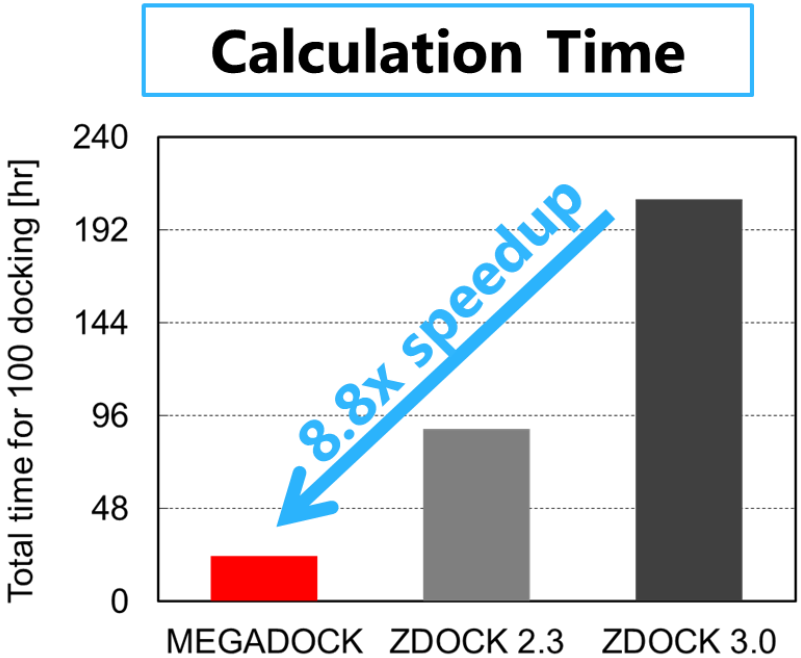


③Modulation: complex conjugate & multiplication

Calculation Time

- MEGADOCK compresses 3 energy terms into only one time FFT calculation
  - 1. Shape complementarity
  - 2. Hydrophobic interaction
  - 3. Electrostatic interaction

- Other docking software needs many time FFT calculation
  - ZDOCK needs 8 times FFT
  - PIPER needs 22 times FFT



Problems: Large calculation time

- Application example

A. Ozbabacan S.E., *et al.*, *J. Struct. Biol.*, 2012.

  – Apoptosis pathway dataset
    - Includes 158 proteins
    - Combination of proteins: 158 × 158 = 24,964 pairs
    - Average docking time of 1 pair in 1 CPU core : 12.5 mins
    - Runtime: 12.5mins × 24,964 pairs = 217 days

- Faster calculation method is required

Research Purpose

- Purpose
  - Acceleration of protein-protein docking calculation of MEGADOCK

- Approaches
  - Acceleration by GPU
    1. GPU Implementation of main processes
    2. Optimization of FFT size
    3. Using full computing resources in a node

Outline

- Background
- MEGADOCK-GPU
- Evaluation of Performance
- Conclusion

Graphics Processing Unit

- GPU (Graphics Processing Unit)
  - Processers for Graphics processing
  - Computational performance of GPUs overtakes that of CPUs
  - High efficiency

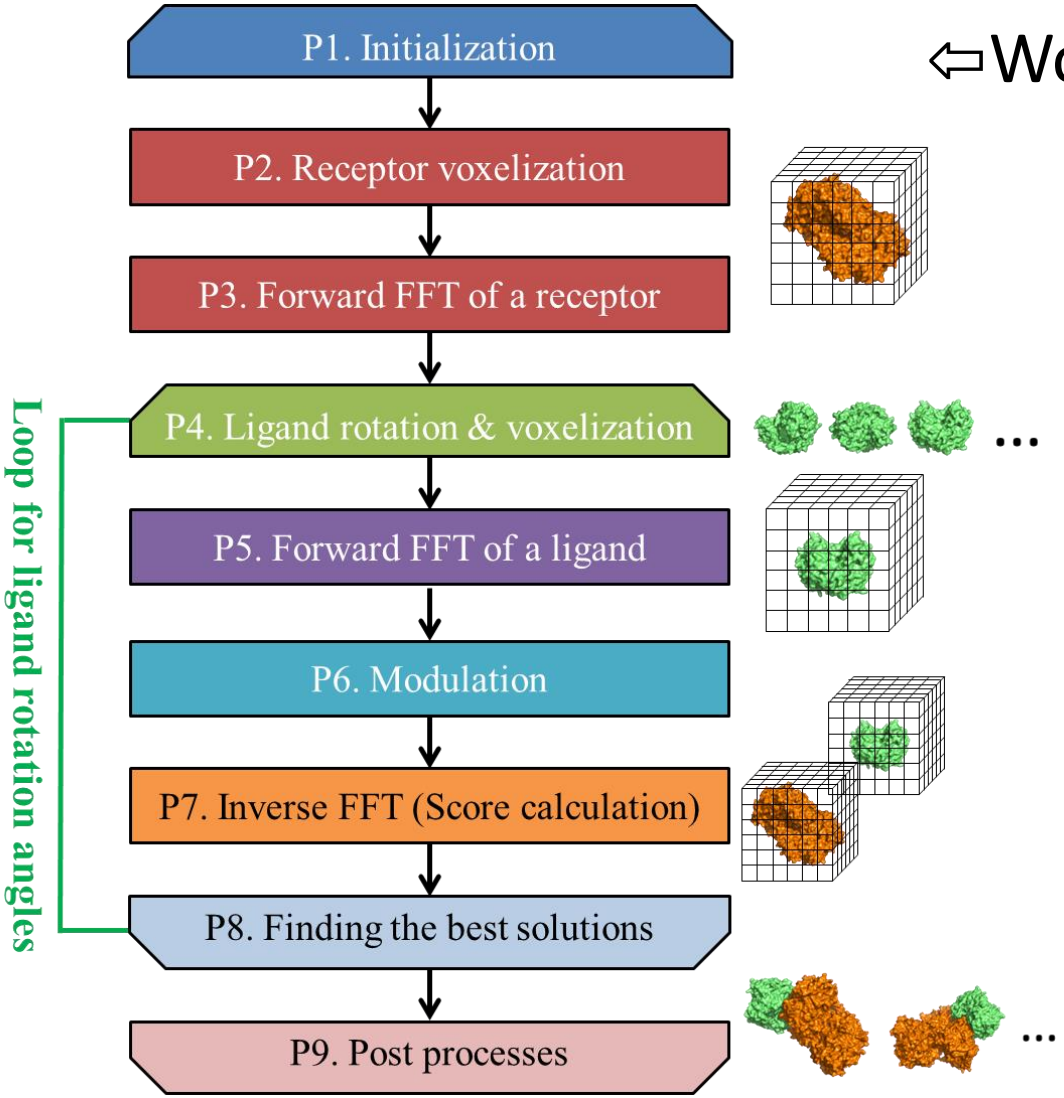
		Performance [GFLOPS]	Power Consumption [W]	Efficiency [GFLOPS/W]
GPU	NVIDIA Tesla M2050	515	225	2.29
CPU	Intel Xeon X5670	70	95	0.74

- CUDA (Compute Unified Device Architecture)
  - Development platform for GPU programming

Related Works

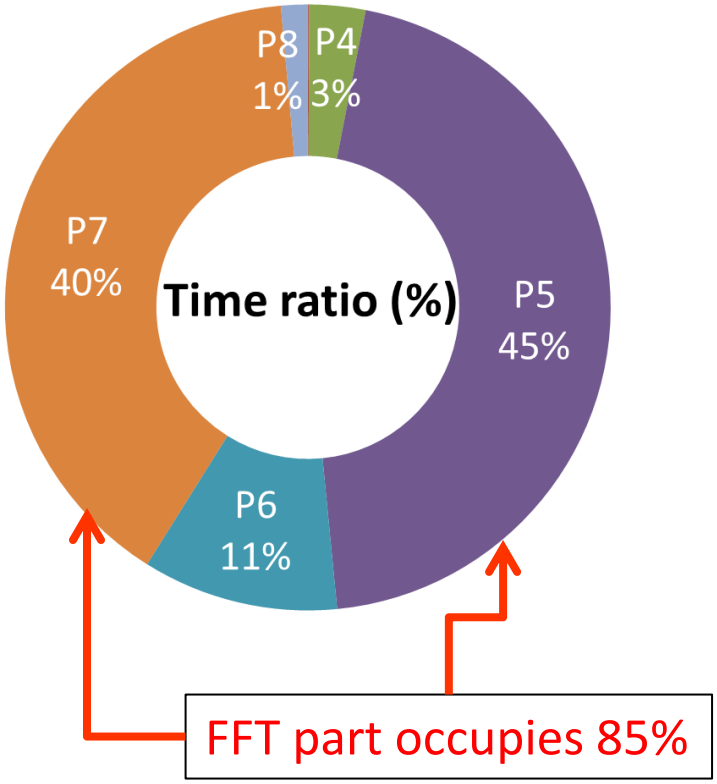
- GPU-accelerated bioinformatics software
  - GPU-BLAST P. D. Vouzis, *et al*., *Bioinformatics,* 2011.
  - GHOSTM S. Suzuki, *et al*., *PLOS ONE,* 2012.
  - PIPER D. Kozakov, *et al*., *Proteins*, 2006.
    - FFT-based protein-protein docking software
    - GPU-accelerated B. Sukhwani, *et al*., *GPGPU-2*, 2006.
      - All processes were not on GPUs

Bottlenecks in MEGADOCK CPU Version



⇦Workflow of MEGADOCK

⇩ Profile of CPU version



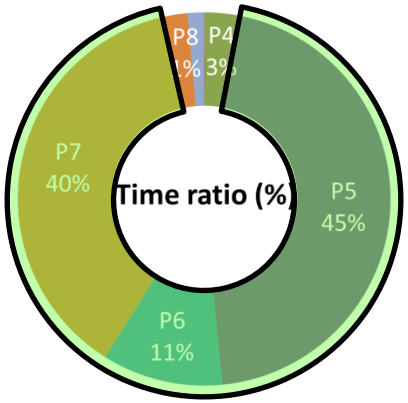
Approach① GPU Implementation of main processes

- P5, P7: Forward FFT & Inverse FFT
  - Accelerated by using NVIDIA CUFFT library

- P6: Modulation
  - Modulation: complex conjugates and multiplication

$$\mathrm{FFT}[\mathrm{R}]^* \times \mathrm{FFT}[\mathrm{L}]$$

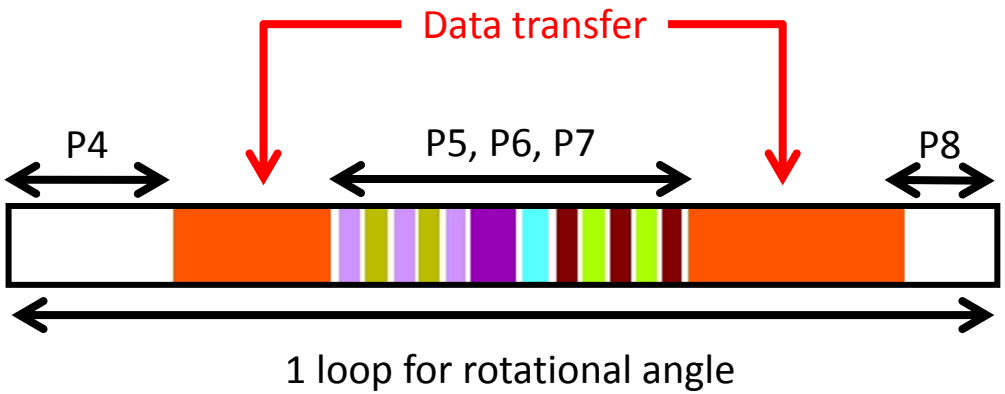
  - Parallelized by voxel element

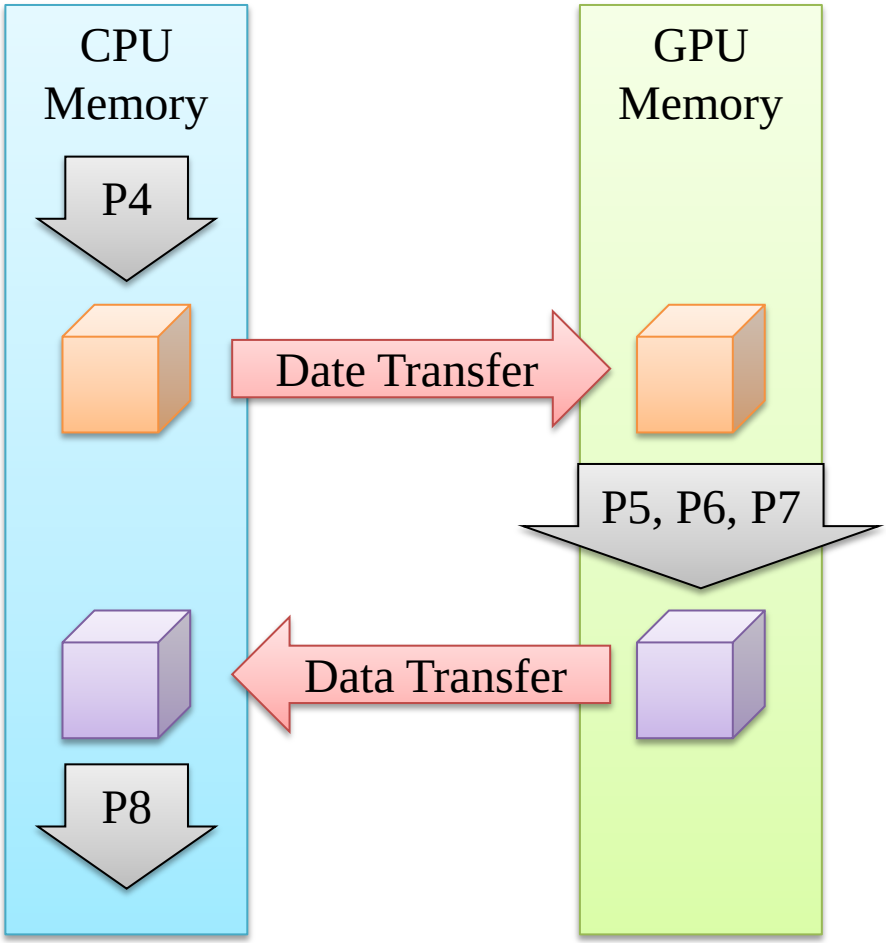


Approach① GPU Implementation of main processes

- P5, P6, P7: Forward FFT, Modulation, Inverse FFT Processes are performed on GPUs

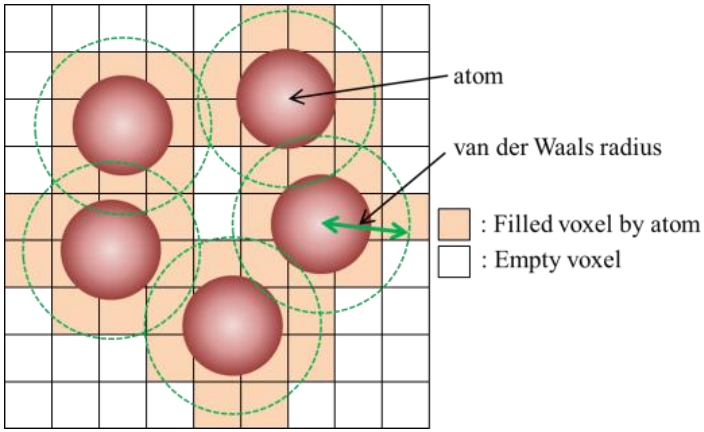
- However large temporary data should be transferred



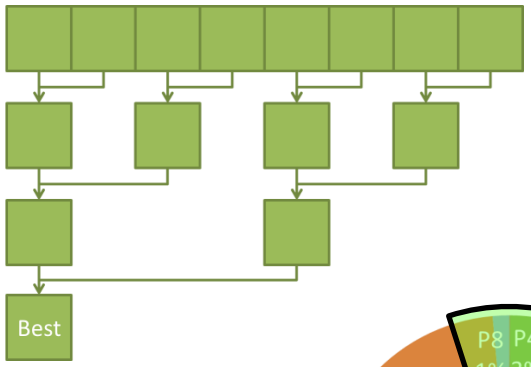


① GPU Implementation of main processes

- P4: Ligand voxelization
  - Voxelization: assigning a value to each voxel based on atom radius
  - Parallelized by atom

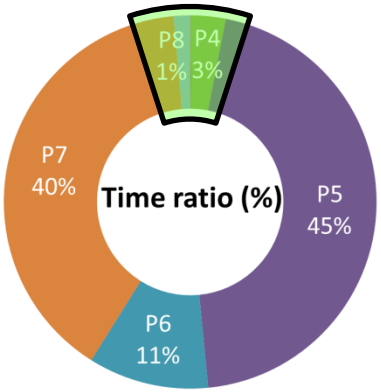


- P8: Finding the best solutions
  - Using reduction method



- All processes (P4) – (P8) are performed on GPU

➡ avoid to transfer large temporary data



② Optimization of FFT Size

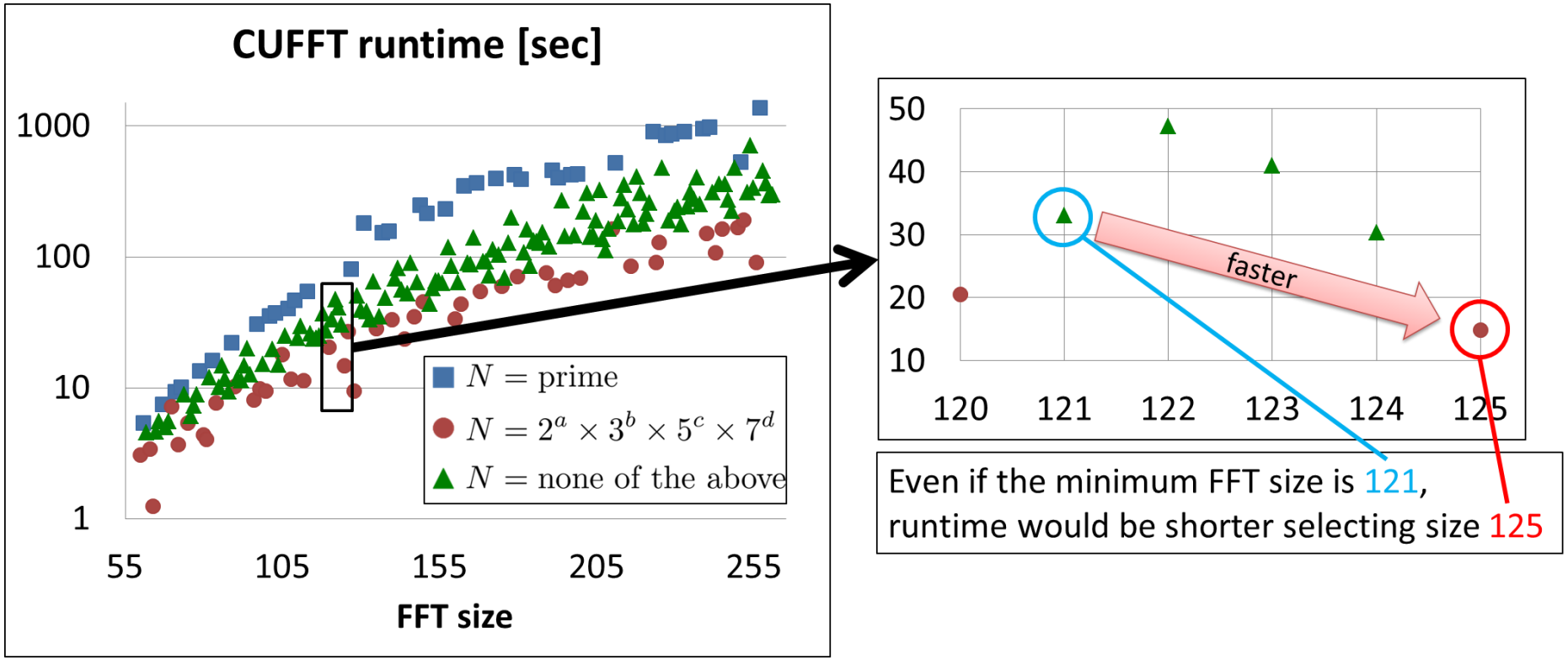
- FFT size is decided based on the protein size
- FFT runtime seems to be proportional to FFT size
- However, CUFFT library may drastically slow down on some FFT sizes
  - Original MEGADOCK uses FFTW library and the influence of this problem is small

- According to the manual, CUFFT library shows the best performance on condition that:

$$\text{FFT size } N = 2^a \times 3^b \times 5^c \times 7^d$$

Approach② Optimization of FFT Size

- Relation between FFT size and runtime

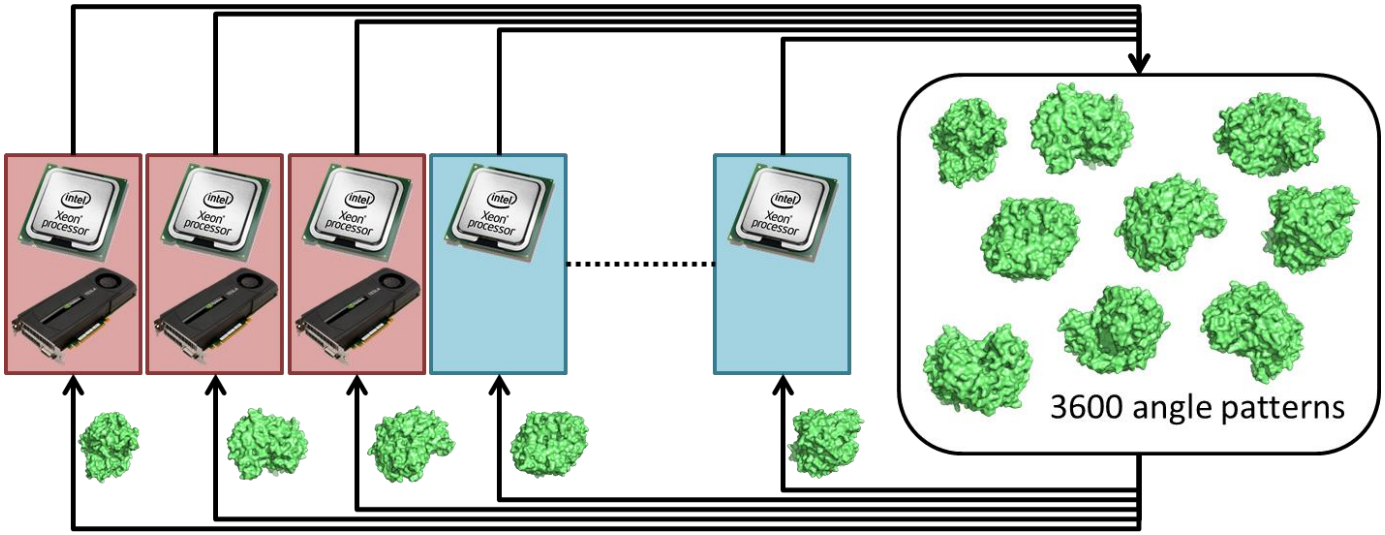


⇨Select FFT size from only FFT sizes that CUFFT library can process efficiently

③Using full computing resources in a node

- Our computing systems TSUBAME 2.0
  - Multiple CPU cores and GPUs

- TSUBAME 2.0 thin node: 12 CPU cores and 3 GPUs
  - Assign decomposed works to multiple CPU cores and GPUs dynamically
    - 3 CPU cores & 3 GPUs: used as GPU version
    - 9 CPU cores: used as CPU version



Outline

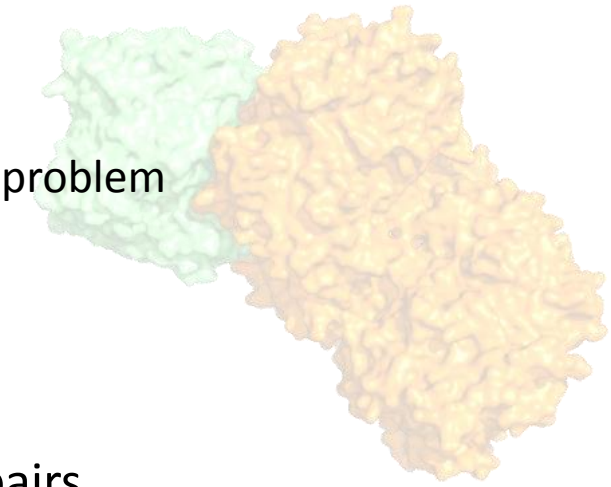
- Background
- MEGADOCK-GPU
- Evaluation of Performance
- Conclusion

Experiment Environment

- Computation Environment

	Tokyo Tech TSUBAME 2.0 Thin Node
CPU	Intel Xeon X5670, 2.93[GHz] (6 cores) × 2
GPU	NVIDIA Tesla M2050, 1.15[GHz] (448 cores) × 3
Memory	54[GB]
FFT library	FFTW (CPU), CUFFT (GPU)

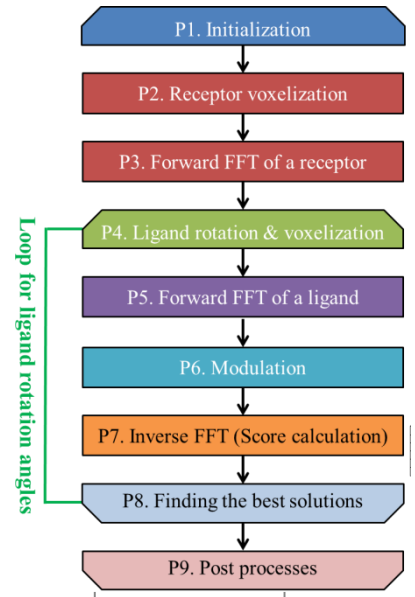
- Dataset
  - Protein-Protein Docking Benchmark 4.0
    - Typical benchmark for protein-protein docking problem
    - 352 protein pairs

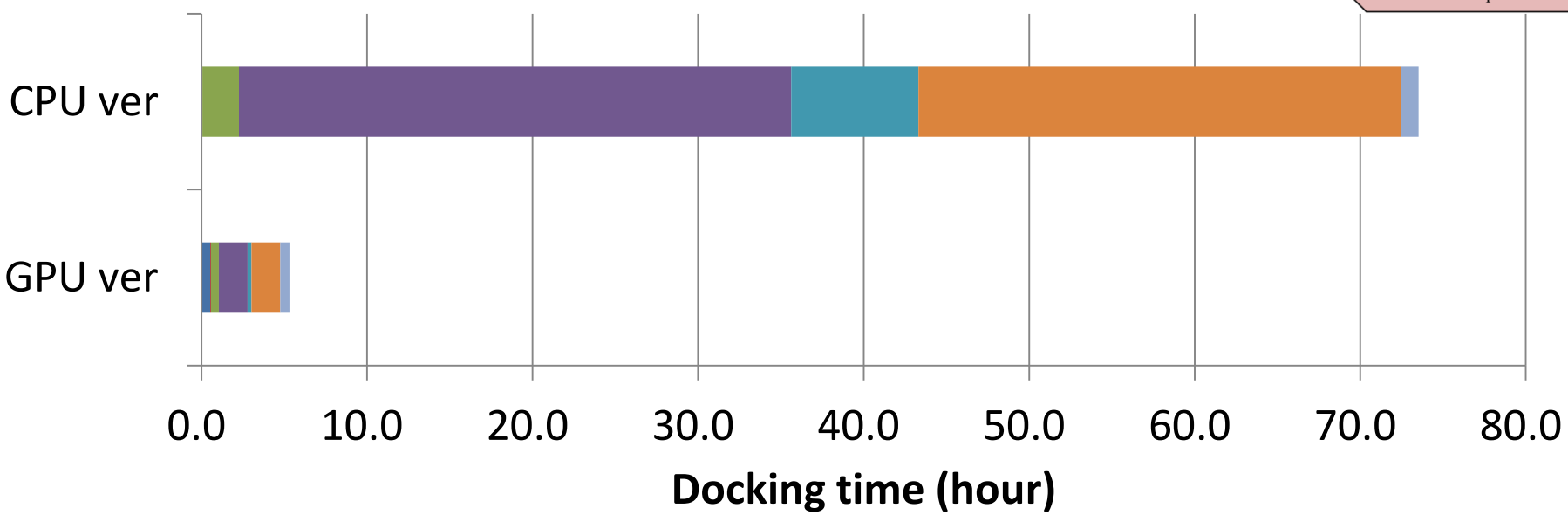


- Measurement
  - Total docking calculation time of 352 protein pairs

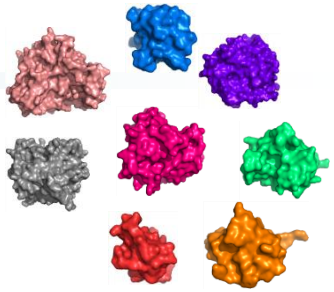
Comparison of each process (1 CPU core vs. 1 CPU core and 1 GPU)

- Comparison of CPU version and GPU version
  1. FFT, Modulation: 20-30-fold faster
  2. Voxelization,
     Finding the best solutions: 2-6-fold faster
  3. Only initialization process slows down
     because of GPU initialization

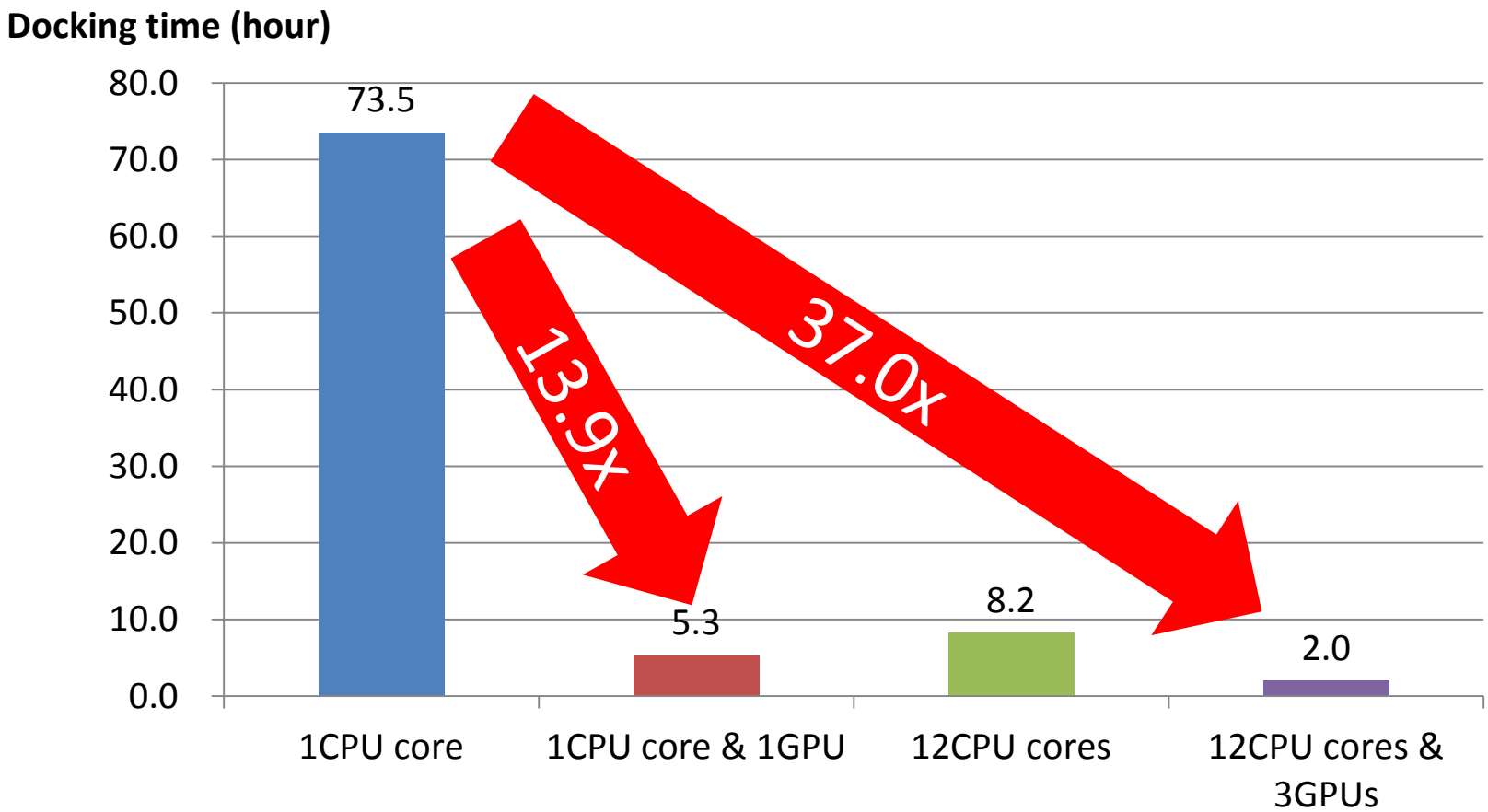




Comparison of total docking runtime



- Comparison of CPU version and GPU version



Conclusion

- We have accelerated docking calculation of MEGADOCK
  - 1 CPU core & 1 GPU: 13.9-fold acceleration
  - 12 CPU cores & 3 GPUs: 37.0-fold acceleration

- Ex.) Prediction for an apoptosis pathway
  - Runtime in 1 CPU core: 217 days
  - Runtime in 12 CPU cores & 3 GPUs : 6 days

MEGADOCK-GPU is freely available at www.bi.cs.titech.ac.jp/megadock/gpu/