東京工業大学
Tokyo Institute of Technology

Protein-Protein Docking on Hardware Accelerators: Comparison of GPU and MIC Architectures

APBC2015, HsinChu, Taiwan, 23 Jan. 2015
Session 15: Protein Function and Mutation (Part II)

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

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

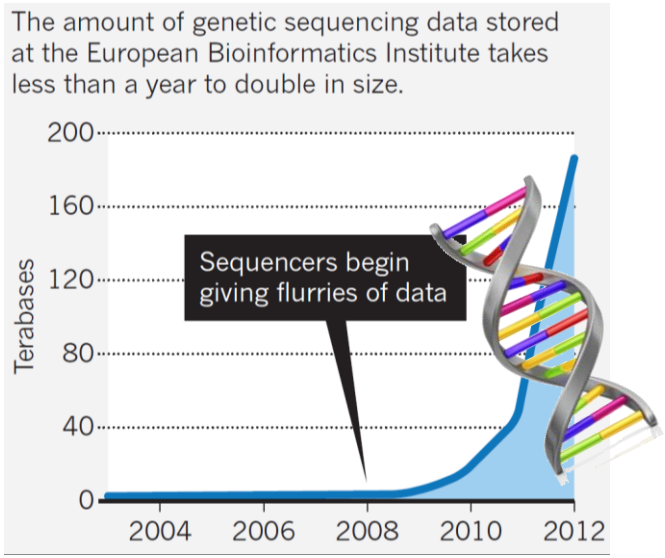
TOKYO TECH Pursuing Excellence
TOKYO INSTITUTE OF TECHNOLOGY

Outline

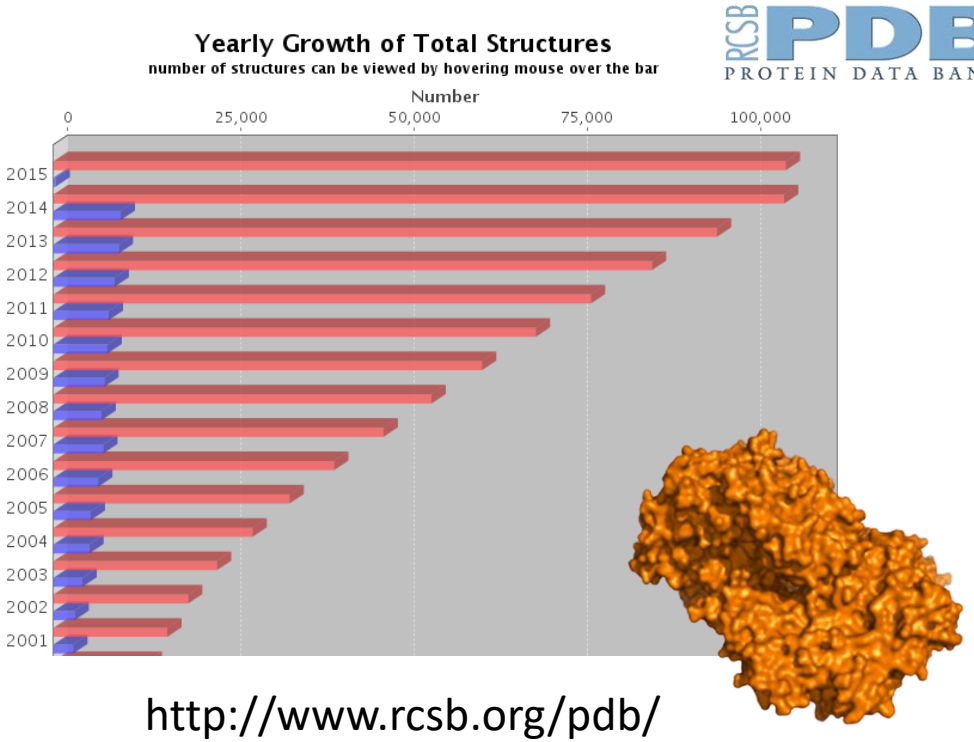
- **Background**
  - Hardware accelerator
  - Protein-protein docking
- **Implementation**
  - GPU implementation
  - MIC implementation
- **Evaluation experiment**
- **Summary**

The Era of Bigdata Bioinformatics

Data explosion



Vivien Marx, *Nature*, 498, 2013



http://www.rcsb.org/pdb/

Require huge computation power!

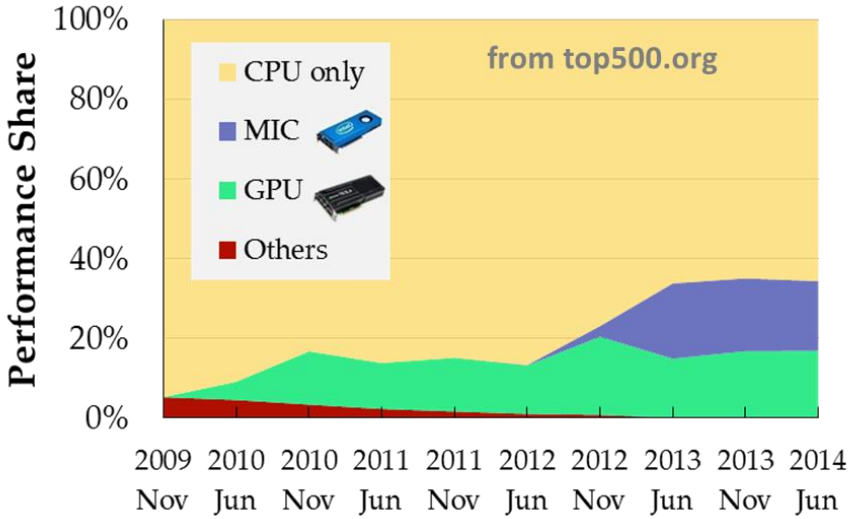
- High Performance Computing (HPC)

TSUBAME (Tokyo Tech)

K computer (RIKEN)

Hardware Accelerator

- **Hardware accelerator**
  - Devices for improving computation performance
  - Mainstream of HPC
    - ~40% performance share in top500 supercomputers



- **Examples of hardware accelerators**
  - Primary accelerators
    - GRAPE (for N-body problem, 1989-)
    - ClearSpeed (for general purpose, 2003-)
    - Cell Accelerator Board (for general purpose, 2006-)
  - Currently most efficient accelerators
    - GPU (for image processing and general purpose)
    - MIC architecture (for general purpose)

GPU (NVIDIA Tesla)

MIC (Intel Xeon Phi)

GPU (Graphics Processing Unit)

- **Features**
  - Parallel computation using many small cores
  - Merit: high computation performance and power efficiency

	CPU	GPU
Product	Intel Xeon E5-2670	NVIDIA Tesla K20X
Number of cores	8	2,688
Theoretical performance [GFLOPS]	332.8	3,935.2
Power consumption [W]	115.0	235.0
Power efficiency [GFLOPS/W]	2.9	16.7

  - Demerit: requires heavy implementation cost
    - Programming of dedicated language, such as CUDA
      - CUDA: development environment provided by NVIDIA
    - Understanding of complex multi-layered memory architecture
    - Consideration of overheads of such data transfer

MIC (Many Integrated Core)

- **Features**
  - Merit 1: high computation performance and power efficiency

	CPU	MIC
Product	Intel Xeon E5-2670	Intel Xeon Phi 5110P
Number of cores	8	60
Theoretical performance [GFLOPS]	332.8	2,021.6
Power consumption [W]	115.0	225.0
Power efficiency [GFLOPS/W]	2.9	9.0

  - Merit 2: lower implementation cost (compared to GPU)
    - Not require programming of dedicated language
    - Only add OpenMP pragma to use
  - Demerit: insufficient related work and references
    - Actual performance is unknown

Acceleration of Bioinformatics Applications by Hardware Accelerators

- **GPU**
  - Genome sequence analysis
    - GPU-BLAST
      - **3-4 times** faster than **1 CPU core** by using GPU

      P. D. Vouzis, *et al*., *Bioinformatics,* 2011.
    - GHOSTM (for Metagenomics)
      - **4.2 times** faster than competitive CPU application by using GPU

      S. Suzuki, *et al*., *PLOS ONE,* 2012.
  - Protein-protein docking
    - PIPER-GPU
      - **16 times** faster than **1 CPU core** by using the GPU

      B. Sukhwani, *et al*., *GPGPU*, 2009.
    - MEGADOCK-GPU
      - **15 times** faster than **1 CPU core** by using the GPU

      T. Shimoda, *et al*., *ACM-BCB*, 2013.
  - Molecular dynamics simulation
    - Conventional MD simulation
      - **40 times** faster than **1 CPU core** by using the GPU

      J.A. van Meel, *et al*., *Mol Sim*, 2008.

Acceleration of Bioinformatics Applications by Hardware Accelerators

- **MIC**
  - Smith-Waterman algorithm for long DNA sequences
    - SWAPHI-LS
      - **29 times** faster than competitive CPU application by using the MIC

    Y. Liu, *et al.*, *IEEE CLUSTER*, 2014.

  - Genome-wide association study
    - SNP-SNP interaction detection
      - **15-20 times** faster than **1 CPU core** by using the MIC

    D. Sluga, *et al.*, *BMC Bioinform*, 2014.

  - Virtual screening in drug discovery
    - Calculation of the non-bonded interactions

    J. Fang, *et al.*, *IWBBIO*, 2014.

Comparison Studies of Hardware Accelerators

- **Comparison studies**
  - Molecular dynamics simulation — W. Qiang, *et al*., *COSMIC*, 2013.
    - GPU Tesla M2050 vs. MIC Xeon Phi 3120P
      - GPU and MIC are almost the same computation time
  - Genome-wide association study — D. Sluga, *et al*., *BMC Bioinform*, 2014.
    - GPU Tesla K20 vs. MIC Xeon Phi 5110P
      - **GPU is 2 times faster** than MIC
  - Virtual screening — J. Fang, *et al*., *IWBBIO*, 2014.
    - GPU Tesla K20X vs. MIC Xeon Phi 5110P
      - **GPU is 2-4 times faster** than MIC

Problems of Hardware Acceleration

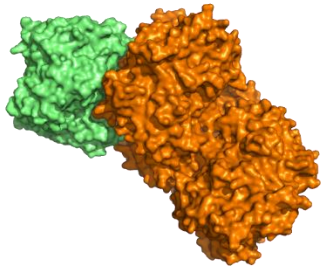
- **Difficult to estimate acceleration effect**
  - Strongly depends on characteristics of hardware accelerators and applications
- **Comparison studies are still insufficient**
  - Due to heavy implementation cost

More comparison studies are required for selecting the best accelerator

Research Purpose

- **Direct comparison of hardware accelerators**
  - Comparison
    - Computation performance
    - Implementation cost
  - Target accelerators:
    - GPU Tesla K20X
    - MIC Intel Xeon Phi 5110P
  - Target application:
    - Real bioinformatics application
      - Protein-protein docking (MEGADOCK)

GPU Tesla K20X	MIC Xeon Phi 5110P
2,688 [core]	60 [core]
0.73 [GHz]	1.05 [GHz]
3,935.2 [GFLOPS]	2,021.6 [GFLOPS]
5,760 [MB memory]	7,697 [MB memory]

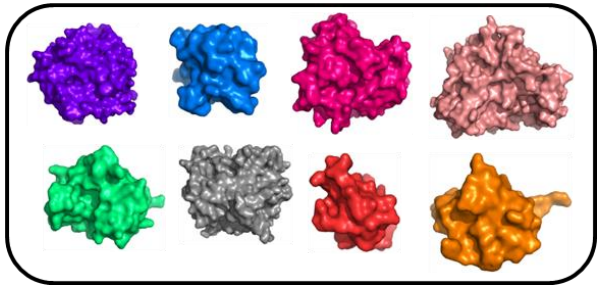


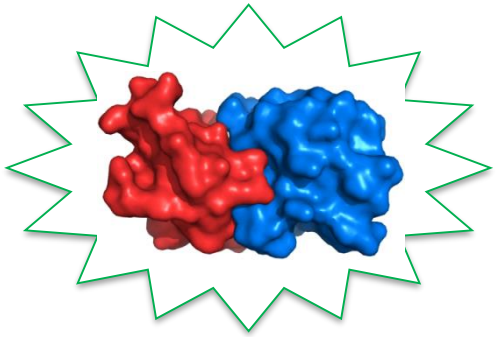
Outline

- **Background**
  - Hardware accelerator
  - Protein-protein docking
- **Implementation**
  - GPU implementation
  - MIC implementation
- **Evaluation experiment**
- **Summary**

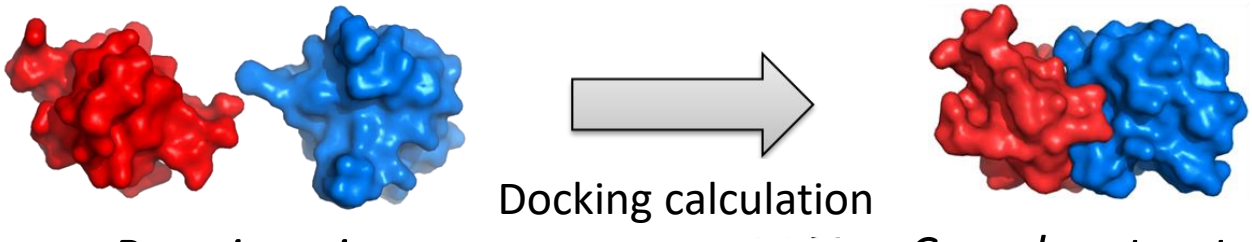
Protein-Protein Docking

- **Proteins interact with each other and perform**
  - Which protein pair interact each other ?





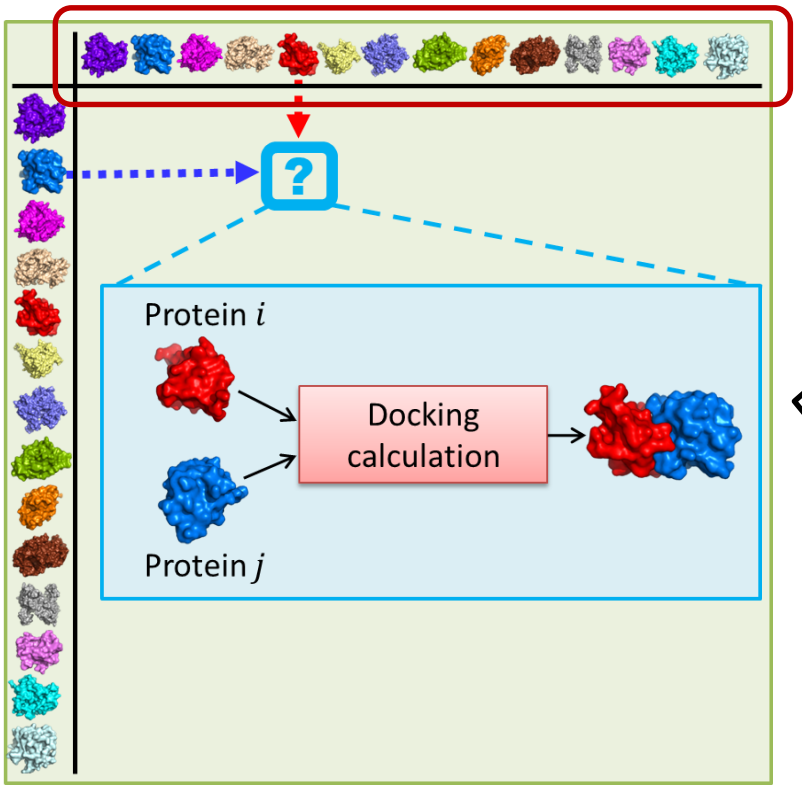
- **Prediction using computer**
  - Protein-protein docking
    - Predict complex structure from protein structure



Protein pair *Complex structure*

All-to-All Protein-Protein Docking

- **FFT-based rigid-body docking**
  - Takes about 5 minutes for one protein pair
- **If target group has 2000 types of proteins**
  - 5 min * $_{2000}C_2$ → about 20 years

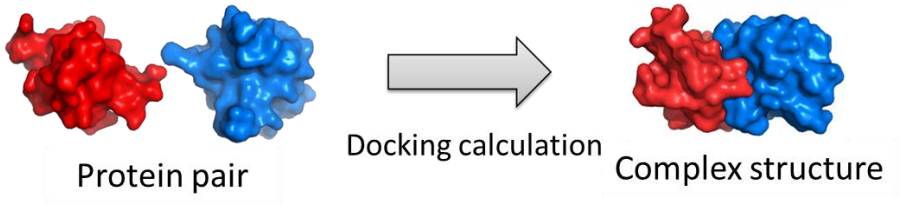


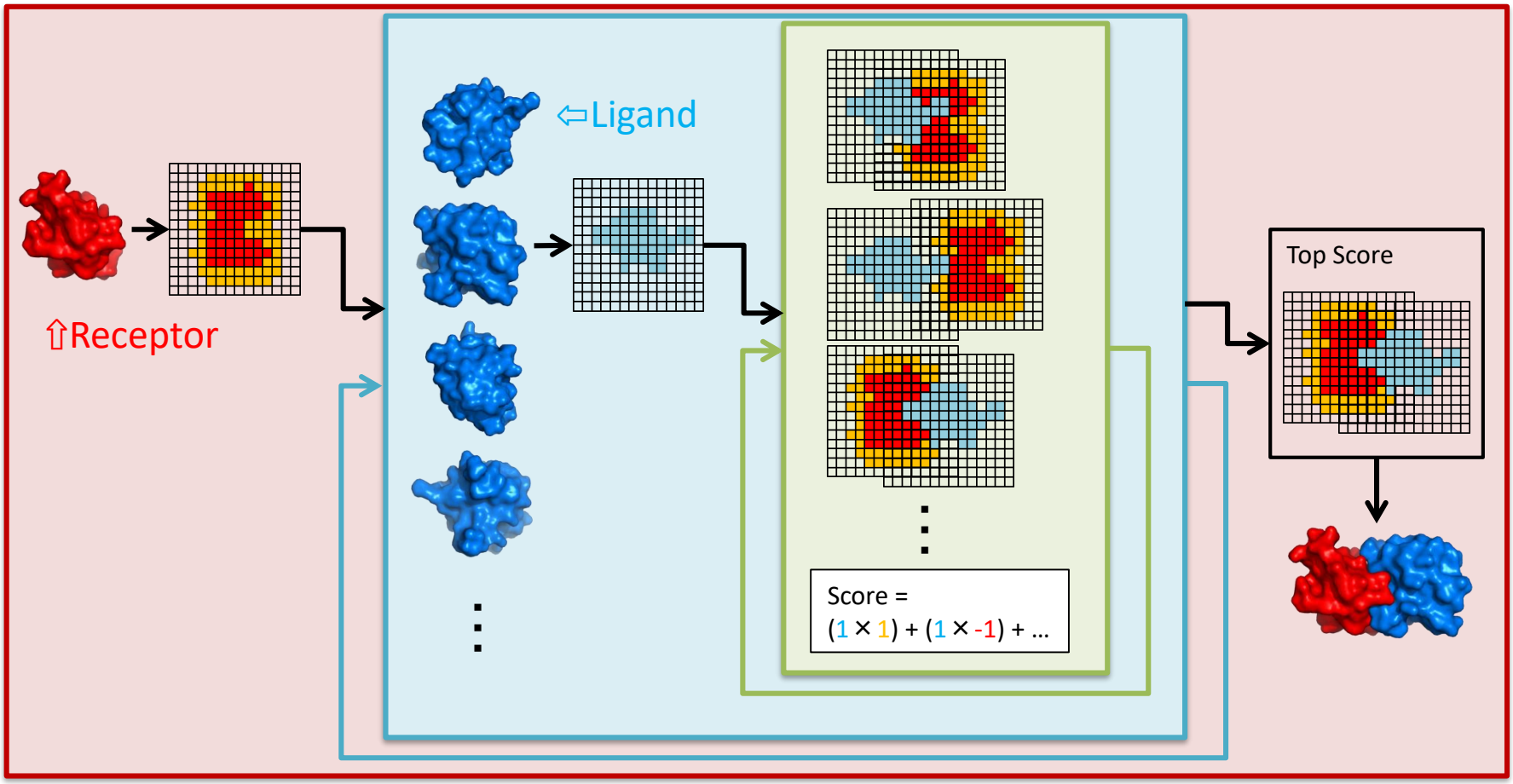
In order to know which protein pair interact each other,
docking calculations of all pairs are required.

Detail of Docking Calculation

- **Flow of docking calculation**

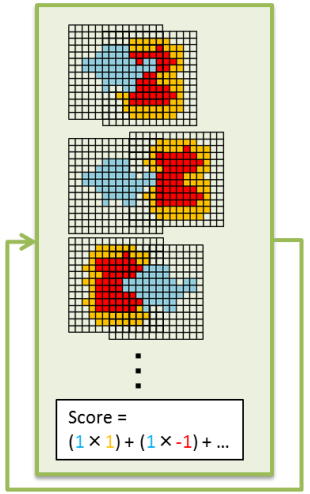
Katchalski-Katzir E, *et al*. *PNAS*, 1992.





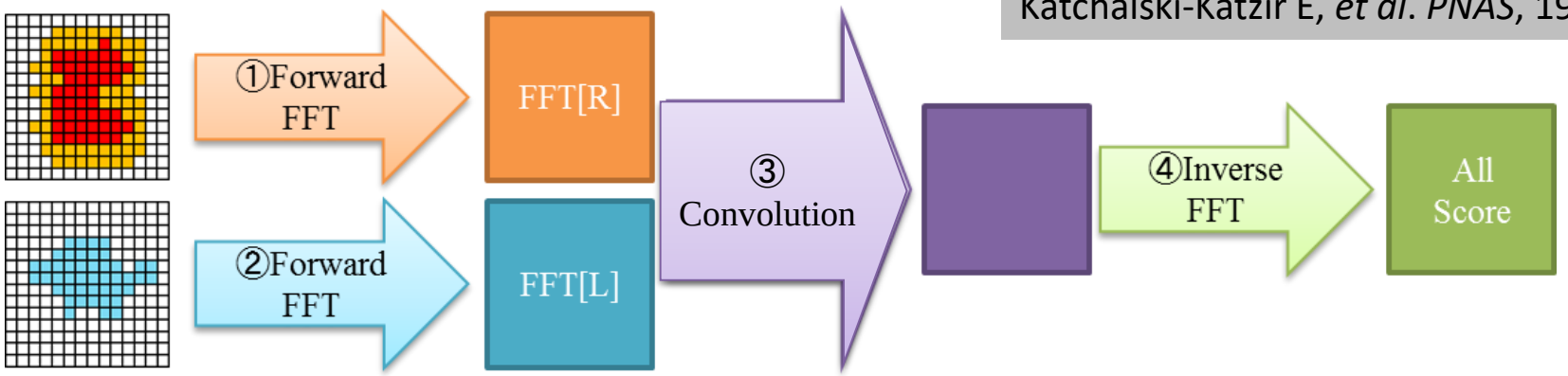
Faster Docking Method Using FFT

- **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 ⇨ $O(N^3 \log N)$

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



FFT-based Protein-Protein Docking Applications

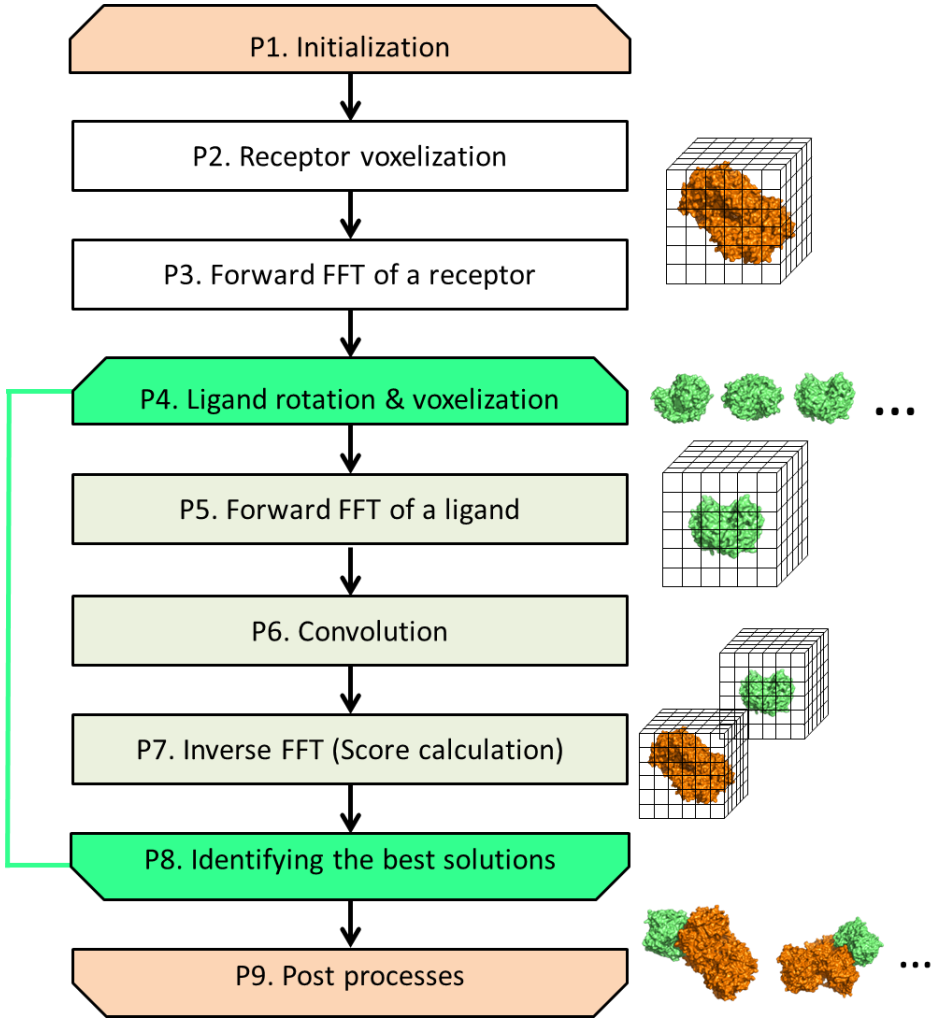
- **PIPER** D. Kozakov, *et al*., *Proteins*, 2006.
  - GPU supported (only in part)
  - 22 times FFT calculation

- **ZDOCK** J. Mintseris, *et al*. *Proteins,* 2007.
  - High accuracy of docking
  - *De facto* standard of protein-protein docking application
  - 7 times FFT calculation

- **MEGADOCK** M. Ohue, *et al*. *Protein Pept Lett,* 2014.
  - Faster docking
    - Only 1 time FFT calculation
    - Docking speed of about 10 times compared with ZDOCK
  - OpenMP supported (Parallelization on multi-CPU cores)

Workflow and Time Profile of Docking

Workflow of docking



Time profile of docking (on CPU core)

	Time [sec.]	Ratio [%]
P1. Initialization	0.0	0.0
P2. Receptor voxelization	0.3	0.2
P3. Forward FFT of receptor	0.1	0.0
P4. Ligand rotation & voxelization	12.9	6.9
P5. Forward FFT of ligand	69.8	37.5
P6. Convolution	27.4	14.7
P7. Inverse FFT	71.5	38.4
P8. Identifying best solutions	4.3	2.3
P9. Post processes	0.0	0.0
Total	186.4	100.0

Target processes of acceleration

Outline

- **Background**
  - Hardware accelerator
  - Protein-protein docking
- **Implementation**
  - GPU implementation
  - MIC implementation
- **Evaluation experiment**
- **Summary**

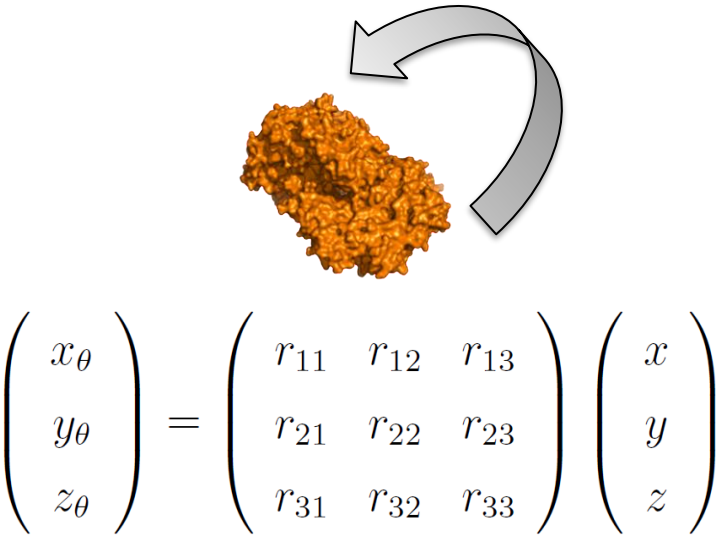
GPU Implementation:

Rotation and Voxelization

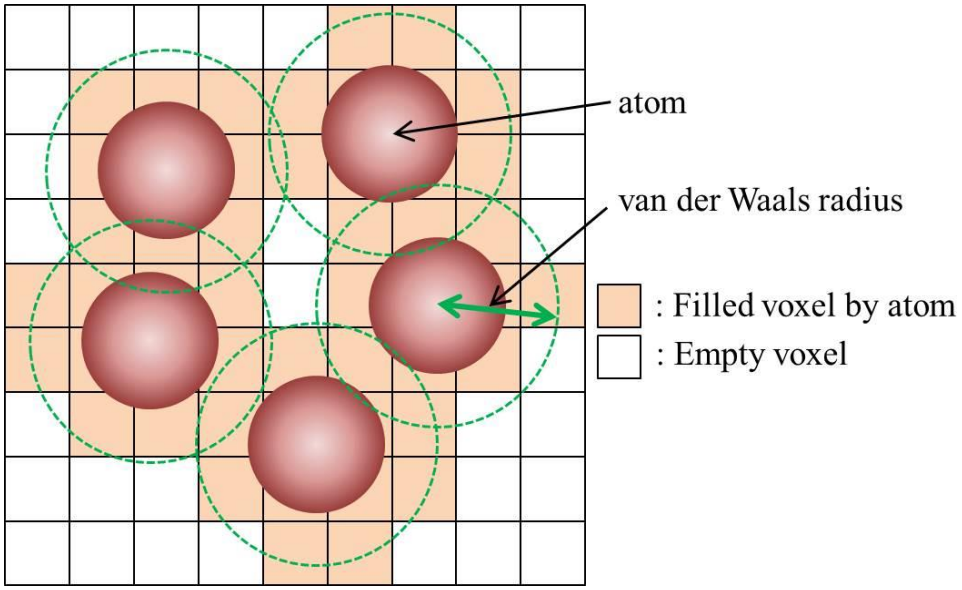
- **P4. Ligand rotation & voxelization**

T. Shimoda, *et al.*, *ACM-BCB*, 2013.

  - Coordinate transformation of atoms corresponding to the rotation angle
  - Voxelization from coordinates and types of atoms
  - Both processes are independent for each atom
    - Assign to 1 GPU thread on 1 atom



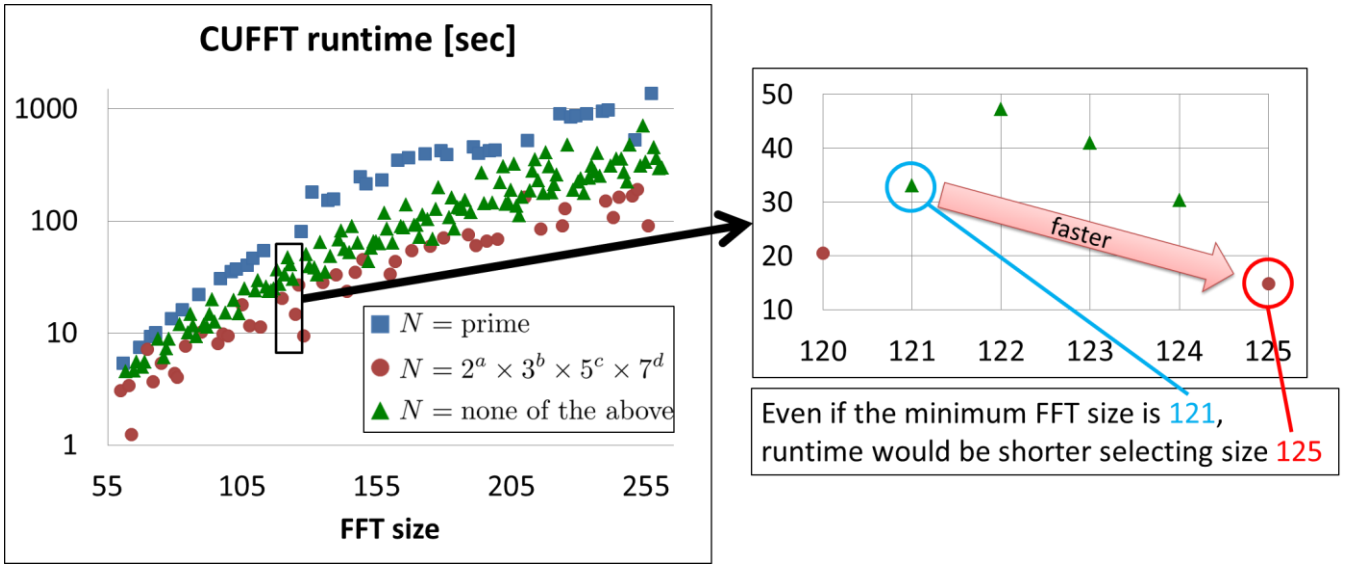
$$\begin{pmatrix} x_\theta \\ y_\theta \\ z_\theta \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



GPU Implementation:

Forward and Inverse FFT

- **P5 & P7. Forward & Inverse FFT**
  - 3-dimensional ($N \times N \times N$) complex FFT using NVIDIA CUFFT library
  - Optimization of FFT size *N*
    - FFT efficiency depends on bases of FFT size
      - CUFFT may drastically slow down on some FFT sizes (especially when *N* is prime)
      - FFT efficiency is higher when $\text{FFT size } N = 2^a \times 3^b \times 5^c \times 7^d$



- Decide FFT size *N* only from the most efficient FFT size set

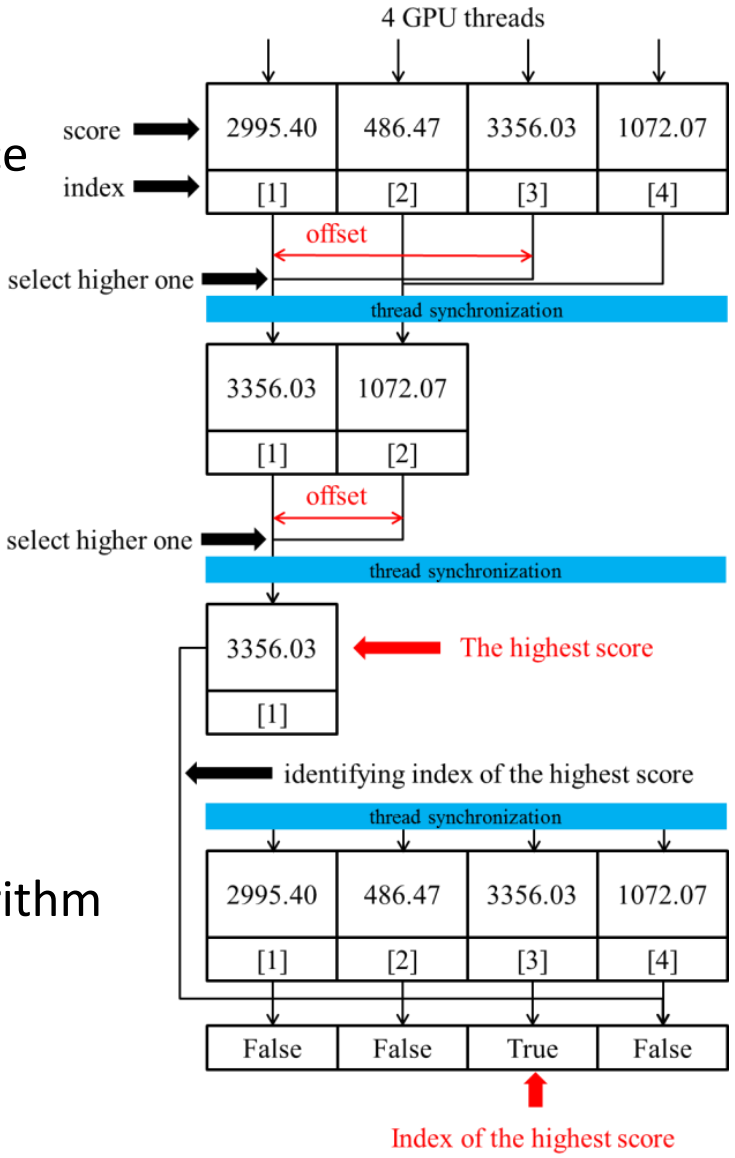
GPU Implementation:

Convolution and Identifying the Best Solutions

- **P6. Convolution**
  - Convolution operation on the Fourier space
    - complex conjugate-multiplied by each element of the FFT output
  - Independently for each element
    - Assign the 1 GPU thread on 1 element

$$\mathrm{Conv}[R,L](x,y,z) = \mathrm{FFT}[R](x,y,z) \times \mathrm{FFT}[L](x,y,z) *$$

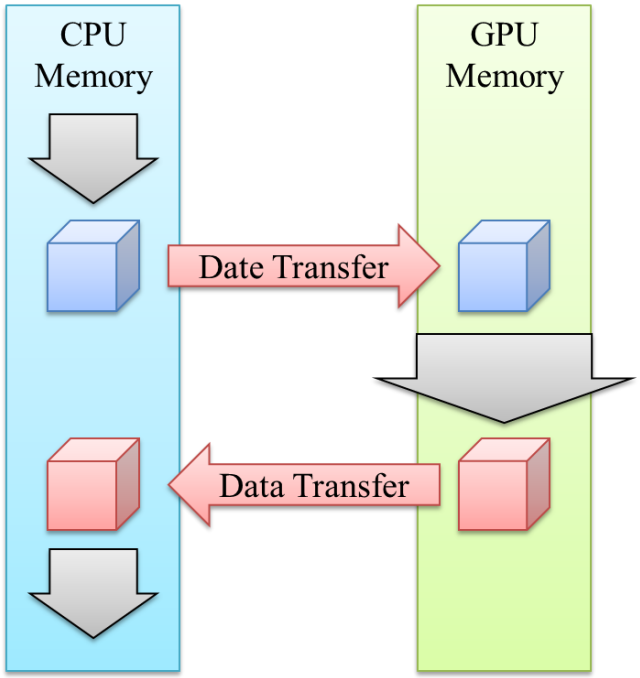
- **P8. Identifying the best solutions**
  - Identify the highest scores from output of inverse FFT
  - Parallel execution by using reduction algorithm



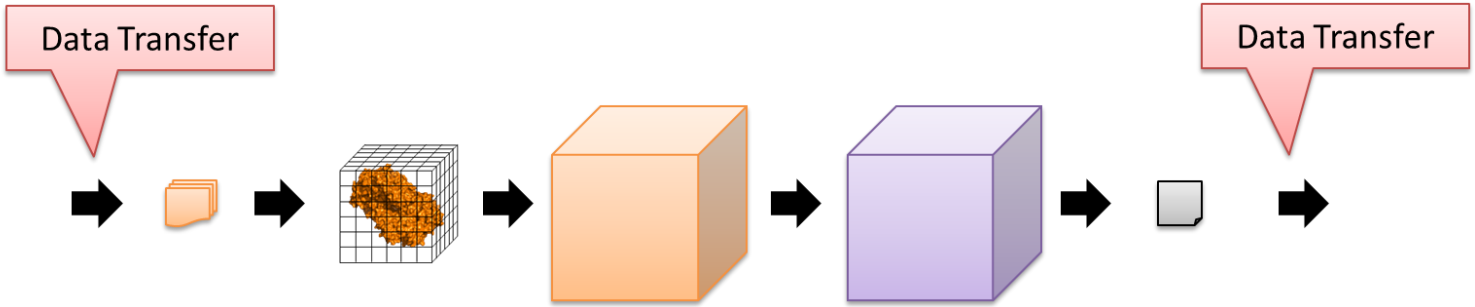
GPU Implementation:

Reduction of Data Transfer

- **Minimization of data transfer time**
  - GPU calculation requires data transfer



  - Avoid to transfer large temporary data
    - CPU to GPU: Atom data of protein
    - GPU to CPU: Best score data



	Atom data	Protein grid	FFT input data	FFT output data	Best score
Size (example)	100 KB	13.8 MB	110 MB	110 MB	1 KB

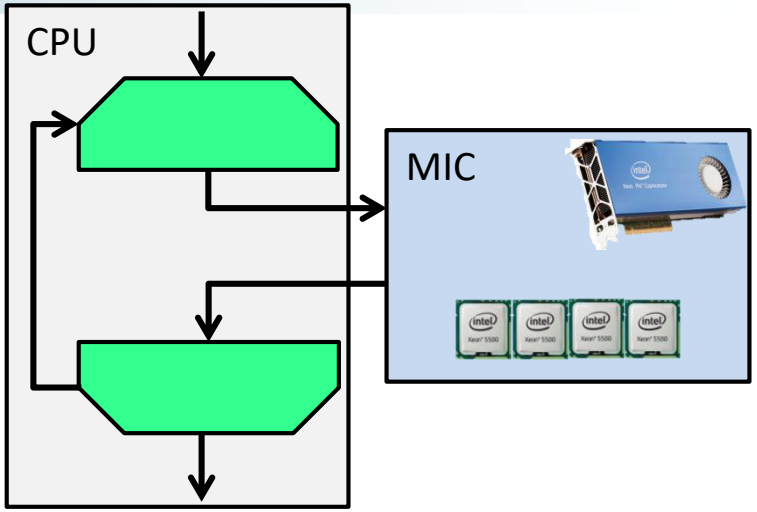
Outline

- **Background**
  - Hardware accelerator
  - Protein-protein docking
- **Implementation**
  - GPU implementation
  - MIC implementation
- **Evaluation experiment**
- **Summary**

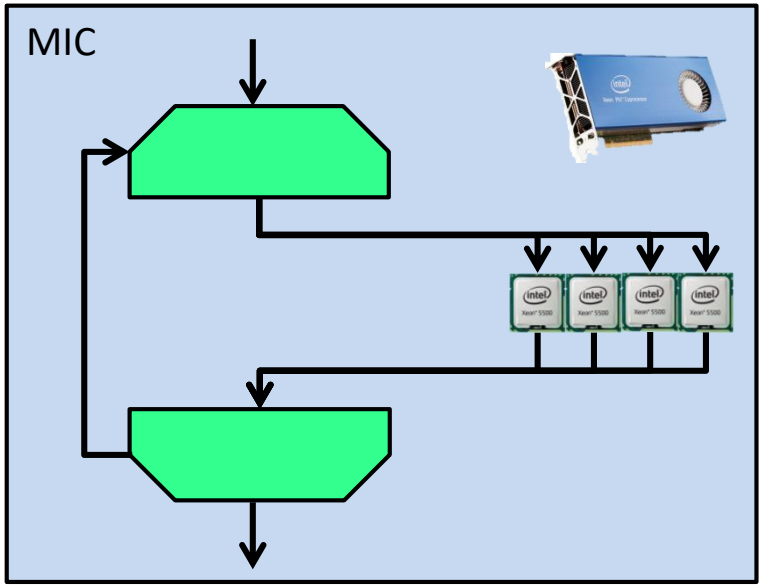
MIC Implementation:

MIC Operation Modes (Offload/Native)

- **MIC has 2 operation modes**
  - Offload mode
    - CPU calls MIC for specified process
      - Like GPU usage
    - Requires description with OpenMP
      - Parallelization
      - Data transfer

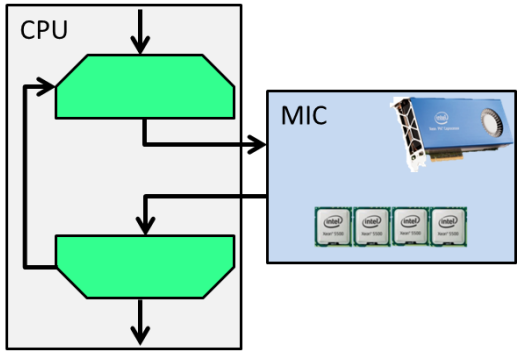


  - Native mode
    - All processes run on MIC
    - Available without any code change if program is parallelized with OpenMP

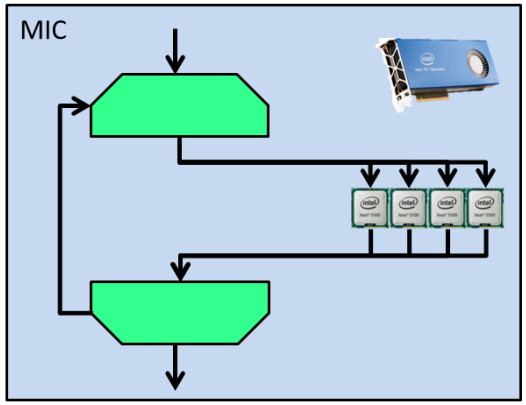


MIC Operation Modes (Offload/Native)

- **Offload mode**
  - Implementation like GPU
  - **240 MIC threads** used for 1 protein pair



- **Native mode**
  - No change of code
    - MEGADOCK is already OpenMP supported
  - **1 MIC thread** used for 1 protein pair
  - Memory problem
    - Many FFT calculations concur



MIC Implementation:

Memory Limitation of MIC Native Mode

- **Memory capacity and number of MIC threads**
  - Intel Xeon Phi 5110P has **8 GB** memory
  - $N \times N \times N$ 3D-FFT requires **$16N^3$ Byte** in each threads
  - All MIC threads: **240 threads**
- **If you use all MIC threads (240 threads):**

  3D-FFT size *N* is up to

$$N = \sqrt[3]{8 \times 10^9 \text{ bytes}/240 \text{ threads}/16} = 127.7...$$

  - Although *N* is about 100 to 300 in protein-protein docking
- **In other words, if you input *N*=200 of FFT size:**

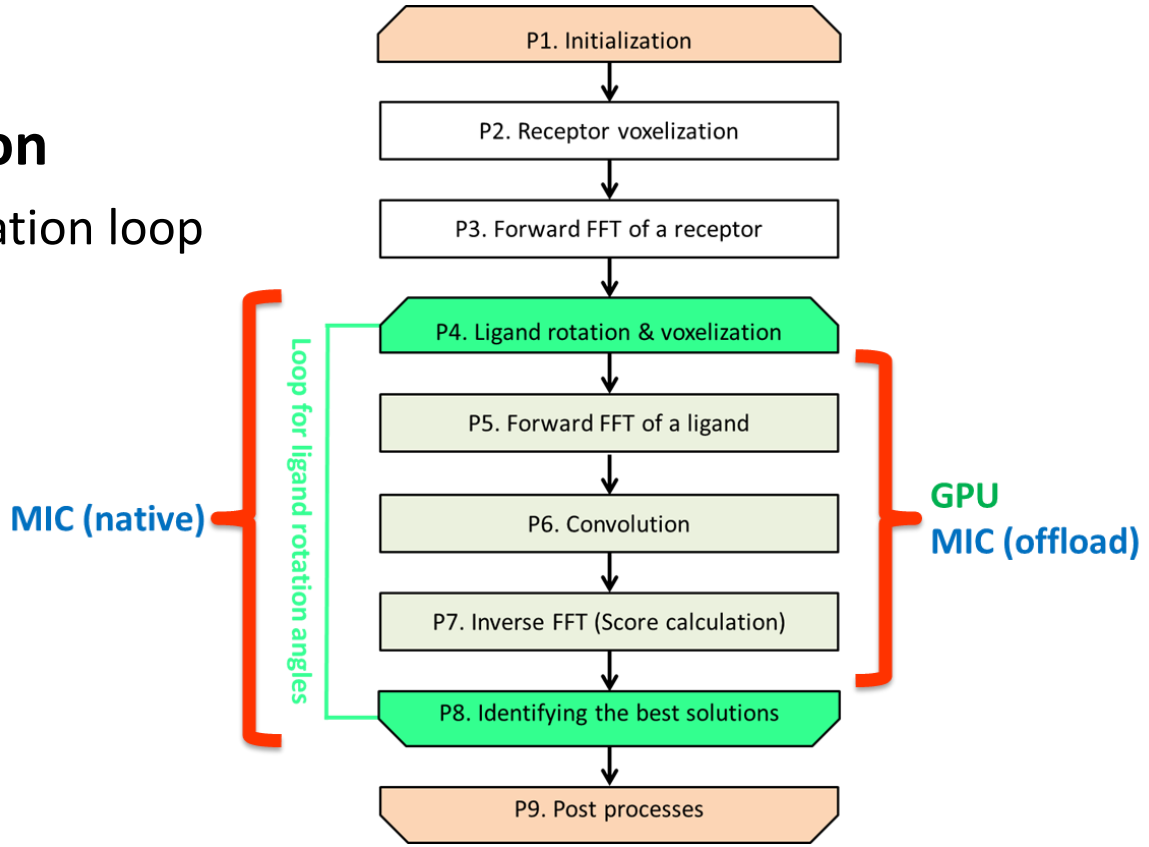
  The number of available MIC threads is up to

$$\#\text{Threads} = 8 \times 10^9 \text{bytes}/(16 \times 200^3) = 62.5$$

Summary of Implementation

- **3 implementations**
  - GPU
  - MIC (offload)
  - MIC (native)

- **Scope of parallelization**
  - One step of ligand rotation loop
    - GPU
    - MIC (offload)
  - Ligand rotation loop
    - MIC (native)



Outline

- **Background**
  - Hardware accelerator
  - Protein-protein docking
- **Implementation**
  - GPU implementation
  - MIC implementation
- **Evaluation experiment**
- **Summary**

Experiment Environment

- **5 computation environments**
  1. 1 CPU core
  2. 8 CPU cores (1 socket)
  3. GPU
  4. MIC (offload)
  5. MIC (native)

	CPU Xeon E5-2670	GPU Tesla K20X	MIC Xeon Phi 5110P
Release date	2012 Q1	2013 Q1	2013 Q1
Number of cores	8	2,688	60
Clock [GHz]	2.60	0.73	1.05
Theoretical performance [GFLOPS]	332.8	3,935.2	2,021.6
Accelerator memory [MB]		5,760	7,697

- **Dataset**
  - Protein-Protein Docking Benchmark 4.0

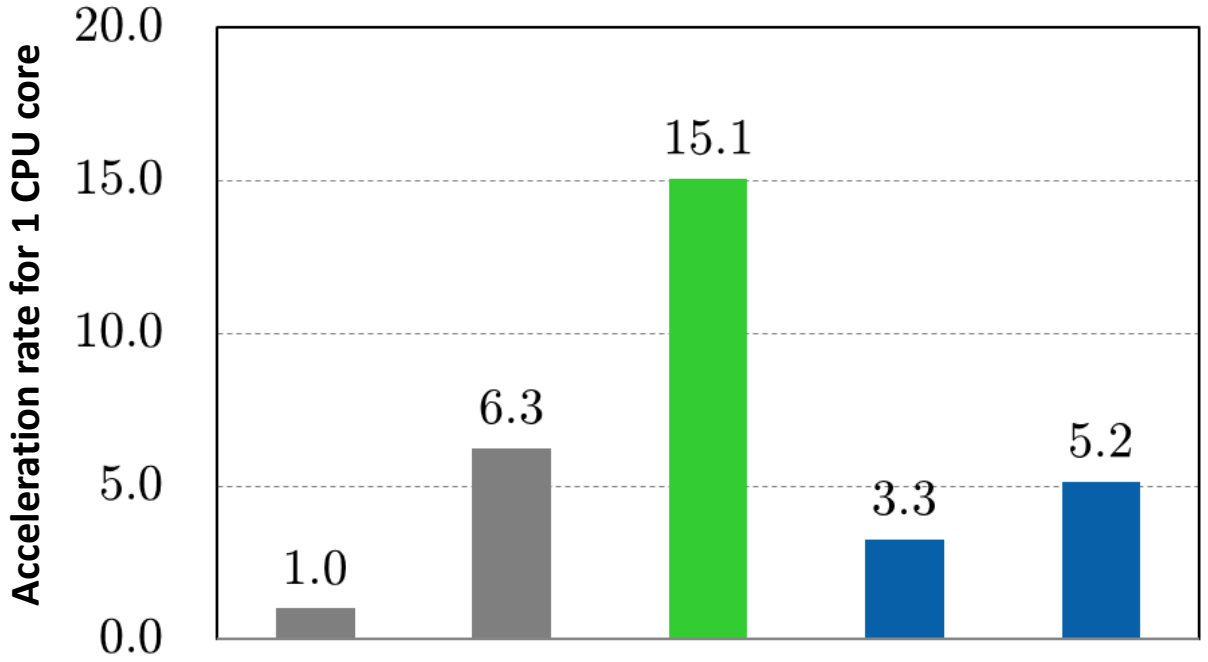
H. Hwang, *et al.*, *Proteins*, 2012.

- **3 performance comparisons**
  1. Docking runtime of 352 protein pairs (total runtime)
  2. Docking runtime of one protein pair (by problem size)
  3. Docking runtime of one protein pair (by process)

- **Implementation cost comparison**

Performance Comparison 1: Total Runtime

- **Total docking runtime of 352 protein pairs**
  - GPU is 5 times faster than MIC offload mode
  - GPU is 3 times faster than MIC native mode

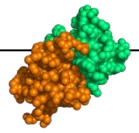


	1 CPU core	8 CPU cores	GPU	MIC offload	MIC native
Calculation time [hour]	30.8	4.9	2.0	9.4	6.0
vs. 1 CPU core	(1.0x)	(6.3x)	(15.1x)	(3.3x)	(5.2x)

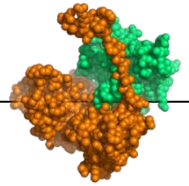
Calculation time [sec.] (Acceleration rate for 1 CPU core)

Performance Comparison 2: Runtime for Each Size of Protein Pairs

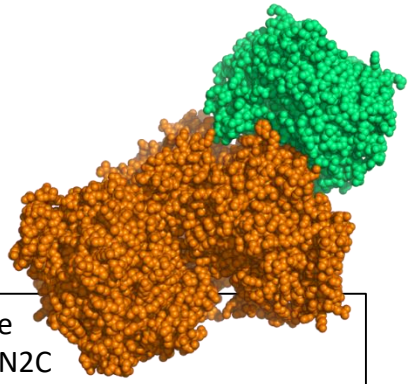
- **Acceleration rate of each system for each size of protein pair**
  - GPU: generally higher performance
  - MIC (offload): lower performance
  - MIC (native): As protein size is larger, performance is lower
    - Only limited threads are available due to memory limitation



Size: Small
PDB ID: 1GCQ
Receptor (Number of residue):
GRB2 C-ter SH3 domain (57)
Ligand (Number of residue):
Vav N-ter SH3 domain (69)
FFT size: **80 × 80 × 80**



Size: Middle
PDB ID: 1JK9
Receptor (Number of residue):
CCS metallochaperone (243)
Ligand (Number of residue):
SOD1 superoxide dismutase (153)
FFT size: **128 × 128 × 128**



Size: Large
PDB ID: 1N2C
Receptor (Number of residue):
Nitrogenase Mo-Fe protein (2000)
Ligand (Number of residue):
Nitrogenase Fe protein (538)
FFT size: **216 × 216 × 216**

	Small			Middle			Large		
1 CPU core	38.3	(1.0x)		186.4	(1.0x)		1105.6	(1.0x)	
8 CPU cores	8.4	(4.6x)		38.5	(4.8x)		177.5	(6.2x)	
GPU	5.8	(6.6x)		10.8	(17.3x)		62.2	(17.8x)	
MIC offload	58.7	(0.7x)	240 threads	77.0	(2.4x)	240 threads	180.6	(6.1x)	240 threads
MIC native	7.6	(5.0x)	240 threads	26.8	(7.0x)	171 threads	310.5	(3.6x)	38 threads

Calculation time [sec.] (Acceleration rate for 1 CPU core)

Performance Comparison 3: Runtime for Each Process

- **Acceleration rate of each system in each process**
  - Target: middle size protein pair

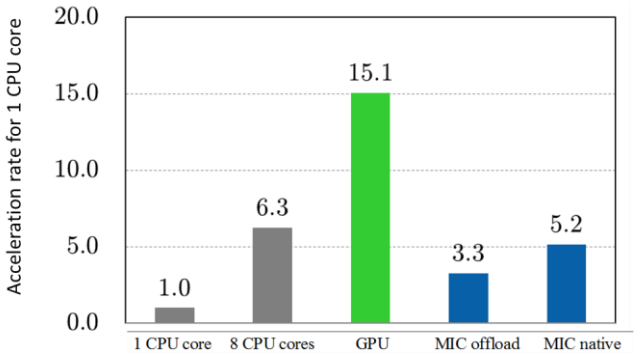
	1 CPU core	8 CPU cores		GPU		MIC offload		MIC native	
P1. Initialization	0.0	0.0		0.8		4.0		0.7	
P2. Receptor voxelization	0.3	0.3	(1.1x)	0.3	(1.1x)	0.3	(1.1x)	4.4	(0.1x)
P3. Forward FFT of receptor	0.1	0.1	(1.0x)	0.0	(1.7x)	1.0	(0.1x)	0.3	(0.2x)
P4. Ligand rotation & voxelization	12.9	3.4	(3.8x)	2.3	(5.5x)	7.4	(1.7x)	1.2	(11.1x)
P5. Forward FFT of ligand	69.8	14.2	(4.9x)	2.2	(31.1x)	15.1	(4.6x)	7.9	(8.9x)
P6. Convolution	27.4	4.6	(5.9x)	1.1	(25.6x)	13.9	(2.0x)	3.6	(7.7x)
P7. Inverse FFT	71.5	14.1	(5.1x)	2.2	(31.8x)	15.2	(4.7x)	8.3	(8.6x)
P8. Finding best solutions	4.3	1.7	(2.5x)	1.7	(2.5x)	9.8	(0.4x)	0.3	(12.5x)
P9. Post processes	0.0	0.0		0.0		0.0		0.0	
Data transfer				0.6		10.1			
Total	186.4	38.5	(4.8x)	10.8	(17.3x)	77.0	(2.4x)	26.8	(7.0x)

Calculation time [sec.] (Acceleration rate for 1 CPU core)

  - FFT processes:
    - GPU: 31 times faster
    - MIC (offload): 5 times faster
    - MIC (native): 9 times faster
  - Data transfer:
    - MIC (offload)'s data transfer takes more time than GPU's one

Summary of Performance Comparison

- **GPU**
  - Highest performance throughout experiment
- **MIC (offload)**
  - Lower performance
  - Cause 1: lower FFT efficiency
    - GPU: 31x, MIC (offload): 5x
    - FFT part occupies the majority of calculation time
  - Cause 2: heavier overheads
    - Data transfer takes a lot of time
- **MIC (native)**
  - Moderate performance
  - As problem size is larger, performance is lower
    - Only limited threads are available due to memory limitation



GPU Tesla K20X	MIC Xeon Phi 5110P
3935.2 [GFLOPS]	2021.6 [GFLOPS]

Comparison of Implementation Cost

- **GPU**
  - Heaviest implementation cost
  - GPU initialization, memory allocation, data transfer
    kernel function (~1,000 lines written in CUDA)
- **MIC (offload)**
  - Lower cost because not require new programming language
  - Implementation similar to GPU
    (~500 lines written in C++ and OpenMP pragma)
- **MIC (native)**
  - Lowest implementation cost
  - Only change compile option, no additional code
    - Because MEGADOCK is already supported by OpenMP parallelization

Conclusion

- **Comparison of acceleration performance and cost**
  - Target application: FFT-based protein-protein docking
  - Target accelerators: GPU, MIC (offload), MIC (native)

	GPU	MIC (offload)	MIC (native)
Theoretical performance [GFLOPS]	3935.2	2021.6	2021.6
Real acceleration rate (vs. 1 CPU core)	15.1x	3.3x	5.2x
primary factor	Higher FFT efficiency	Lower FFT effciency Data transfer overheads	Limitation of number of threads due to memory
Implementation cost	Heaviest Implementation in CUDA	Medium Implementation in C++ and OpenMP	Lower No additional code

- **Theoretical difference is about 1.9-fold**
- **Actual difference**
  - GPU vs. MIC Offload: 4.6-fold
  - GPU vs. MIC Native: 2.9-fold
    - If memory limitation problem will be solved, MIC will reach more high-performance.

Conclusion

- **In acceleration of the FFT-based protein-protein docking application, GPU is superior than MIC architecture**
  - GPU showed higher performance than the theoretical Flops difference.
  - Easiness of MIC native mode is attractive.
  - We want 100-fold memory equipped Xeon Phis for protein-protein dockings!

Acknowledgements

- **Akiyama Lab. Members**

TECH chan

- **This work was supported in part by**
  - JSPS KAKENHI (238750, 248766, 2630002)
  - The Next-Generation Integrated Life Simulation Software Project (ISLiM)
  - HPCI System Research Project (hp140173)
  - Educational Academy of Computational Life Sciences (ACLS), Tokyo Tech