

**Please turn your camera
and microphone OFF !**



RIKEN International HPC Summer School 2022 -Toward Society 5.0- (IHPCSS)

Overview of the Class

<https://www.r-ccs.riken.jp/en/outreach/schools/20220912-1/>

Kengo Nakajima

RIKEN Center for Computational Science (R-CCS)

- Target
 - Parallel FEM
 - Introduction to AI/DNN (Deep Neural Network)
- Supercomputers and Computational Science
- Overview of the Class
- Future Issues

This 5-day intensive course provides introduction to large-scale scientific computing using the most advanced massively parallel supercomputers. Topics cover:

- Parallel Finite-Element Method (FEM)
 - Finite-Element Method (FEM)
 - Message Passing Interface (MPI)
 - Parallel FEM using MPI and OpenMP
 - Parallel Numerical Algorithms for Iterative Linear Solvers
 - Parallel Numerical Libraries (e.g. PETSc)
- Introductions to AI/DNN (Deep Neural Network)

Several sample programs will be provided and participants can review the contents of lectures through hands-on-exercise/practices using **the Fugaku Supercomputer** at RIKEN R-CCS.

Finite-Element Method is widely-used for solving various types of real-world scientific and engineering problems, such as structural analysis, fluid dynamics, electromagnetics, and etc. This lecture course provides brief introduction to procedures of FEM for 1D/3D steady-state heat conduction problems with iterative linear solvers and to parallel FEM. **Lectures for parallel FEM will be focused on design of data structure for distributed local mesh files, which is the key issue for efficient parallel FEM.** Introduction to MPI (Message Passing Interface), which is widely used method as "de facto standard" of parallel programming, is also provided.

Solving large-scale linear equations with sparse coefficient matrices is the most expensive and important part of FEM and other methods for scientific computing, such as Finite-Difference Method (FDM) and Finite-Volume Method (FVM). Recently, families of Krylov iterative solvers are widely used for this process. In this course, details of implementations of parallel Krylov iterative methods are provided along with parallel FEM.

Moreover, lectures on programming for multicore architectures will be also given along with brief introduction to OpenMP and OpenMP/MPI Hybrid Parallel Programming Model.

Finally, lectures and hands-on for using PETSc library will be provide in the morning of the 5th day.

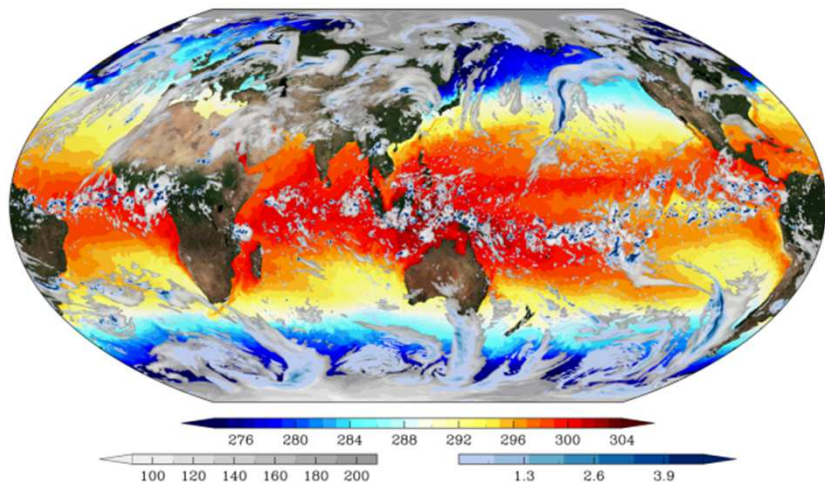
Rapid progress has been made in recent years in artificial intelligence technology, supported by the evolution of the Deep Neural Network (DNN) and the growing use of AI-specific hardware to support this technology. Understanding the basic mathematical background of DNNs, including the concept of Neural Networks as the basis of DNN, their capabilities as high-performance nonlinear approximation functions, and how to determine themselves efficiently, is not necessarily essential for AI users but is regarded as significant for researchers who are further advancing the field.

In this seminar, we will start with the basics of Neural Networks in a short lecture and then move on to approximating functions by backpropagation (i.e., the learning process in artificial intelligence), which is the underlying technology for the current DNNs. In addition, the seminar aims to deepen understanding through practical exercises such as image categorization using a simple program and the standard MNIST image database, identification of hand-written weird kana characters (namely Hentai-Gana; 「変体仮名」 in Japanese), and bridge to the use of higher-level AI frameworks and toolkits.

The following part of this material mainly focuses on the Parallel FEM in first 4-days

Motivation for Parallel Computing (and this class)

- Large-scale parallel computer enables fast computing in large-scale scientific simulations with detailed models. Computational science develops new frontiers of science and engineering.



Physics The Nobel Prize in Physics 2021 Syukuro Manabe - Facts


The Nobel Prize in Physics 2021

Syukuro Manabe
Klaus Hasselmann
Giorgio Parisi

Share this

[Facebook](#) [Twitter](#) [LinkedIn](#) [Email](#)

Syukuro Manabe Facts



Syukuro Manabe
The Nobel Prize in Physics 2021

Born: 21 September 1931, Shinga, Ehime Prefecture, Japan

Affiliation at the time of the award: Princeton University, Princeton, NJ, USA

Prize motivation: "for the physical modelling of Earth's climate, quantifying variability and reliably predicting global warming."

Prize share: 1/4

© Nobel Prize Outreach

Motivation for Parallel Computing (and this class)

- Large-scale parallel computer enables fast computing in large-scale scientific simulations with detailed models. Computational science develops new frontiers of science and engineering.
- Why parallel computing ?
 - faster & larger
 - “larger” is more important from the view point of “new frontiers of science & engineering”, but “faster” is also important.
 - + more complicated
 - Ideal: Scalable
 - Weak Scaling, Strong Scaling

Scalable, Scaling, Scalability

- Solving N^x scale problem using N^x computational resources during same computation time
 - for large-scale problems: Weak Scaling, Weak Scalability
 - e.g. CG solver: more iterations needed for larger problems
- Solving a problem using N^x computational resources during $1/N$ computation time
 - for faster computation: Strong Scaling, Strong Scalability

Scientific Computing = SMASH

Science

Modeling

Algorithm

Software

Hardware

- You have to learn many things.
- Collaboration (or Co-Design) will be important for future career of each of you, as a scientist and/or an engineer.
 - You have to communicate with people with different backgrounds.
 - It is more difficult than communicating with foreign scientists from same area.
- (Q): Your Department ?
 - Science (Physics, Chemistry, Bio etc.)
 - Engineering
 - Math/Applied Math
 - Computer Science

This Class ...

Science

Modeling

Algorithm

Software

Hardware

- Parallel FEM using MPI and OpenMP
- Science: Heat Conduction
- Modeling: FEM
- Algorithm: Iterative Solvers etc.
- You have to know many components to learn FEM, although you have already learned each of these in undergraduate and high-school classes.

Road to Programming for “Parallel” Scientific Computing

Programming for Parallel
Scientific Computing
(e.g. Parallel FEM/FDM)

Programming for Real World
Scientific Computing
(e.g. FEM, FDM)

Programming for Fundamental
Numerical Analysis
(e.g. Gauss-Seidel, RK etc.)

Unix, Fortan, C etc.

Big gap here !!

The third step is important !

- How to parallelize applications ?
 - How to extract parallelism ?
 - If you understand methods, algorithms, and implementations of the original code, it's easy.
 - “Data-structure” is important
- How to understand the code ?
 - Reading the application code !!
 - It seems primitive, but very effective.
 - In this class, “reading the source code” is encouraged.

4. Programming for Parallel Scientific Computing
(e.g. Parallel FEM/FDM)

3. Programming for Real World Scientific Computing
(e.g. FEM, FDM)

2. Programming for Fundamental Numerical Analysis
(e.g. Gauss-Seidel, RK etc.)

1. Unix, Fortan, C etc.

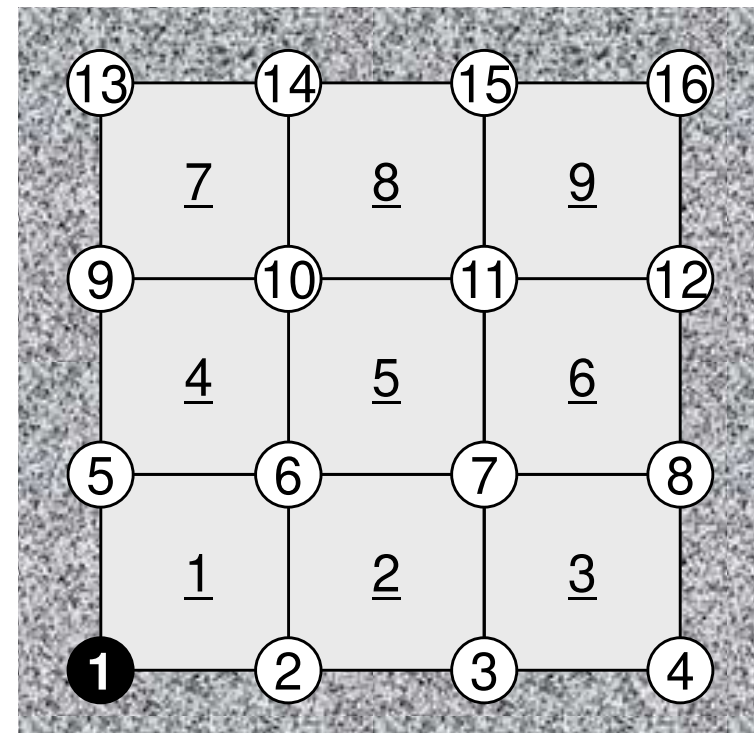


Finite-Element Method (FEM)

- One of the most popular numerical methods for solving PDE's.
 - elements (meshes) & nodes (vertices)
- Consider the following 2D heat transfer problem:

$$\lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + Q = 0$$

- 16 nodes, 9 bi-linear elements
- uniform thermal conductivity ($\lambda=1$)
- uniform volume heat flux ($Q=1$)
- $T=0$ at node 1
- **Insulated boundaries**





Galerkin FEM procedures

- Apply Galerkin procedures to each element:

where $T = [N]\{\phi\}$ in each elem.

$$\int_V [N]^T \left\{ \lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + Q \right\} dV = 0$$

$\{\phi\}$: T at each vertex

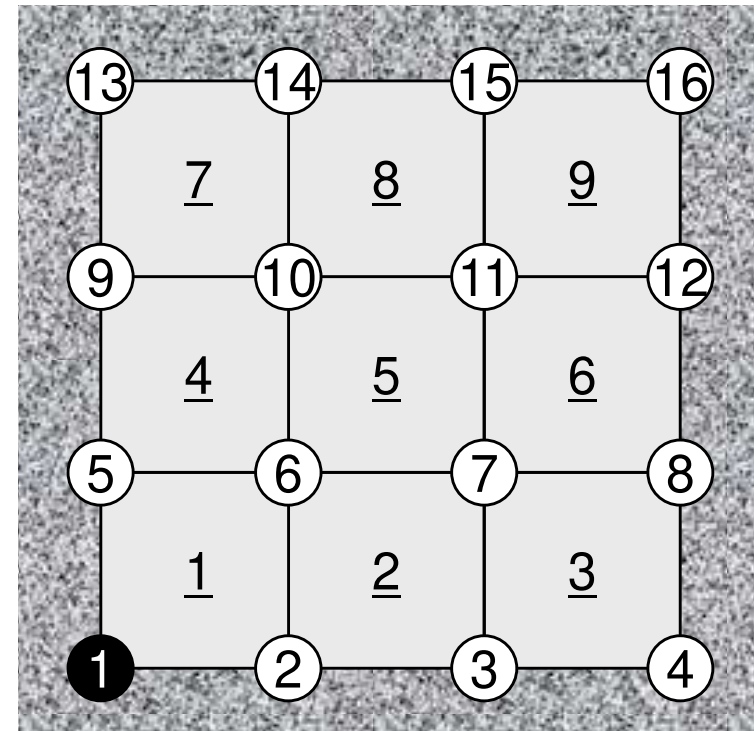
$[N]$: Shape function

(Interpolation function)

- Introduce the following “weak form” of original PDE using Green’s theorem:

$$- \int_V \lambda \left(\frac{\partial [N]^T}{\partial x} \frac{\partial [N]}{\partial x} + \frac{\partial [N]^T}{\partial y} \frac{\partial [N]}{\partial y} \right) dV \cdot \{\phi\}$$

$$+ \int_V Q [N]^T dV = 0$$



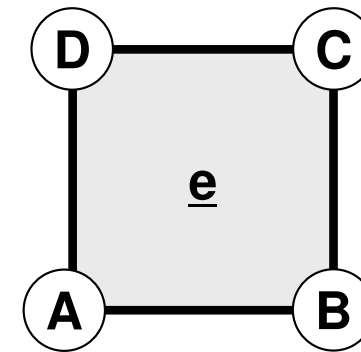


Element Matrix

- Apply the integration to each element and form “element” matrix.

$$-\int_V \lambda \left(\frac{\partial [N]^T}{\partial x} \frac{\partial [N]}{\partial x} + \frac{\partial [N]^T}{\partial y} \frac{\partial [N]}{\partial y} \right) dV \cdot \{\phi\}$$

$$+ \int_V Q [N]^T dV = 0$$



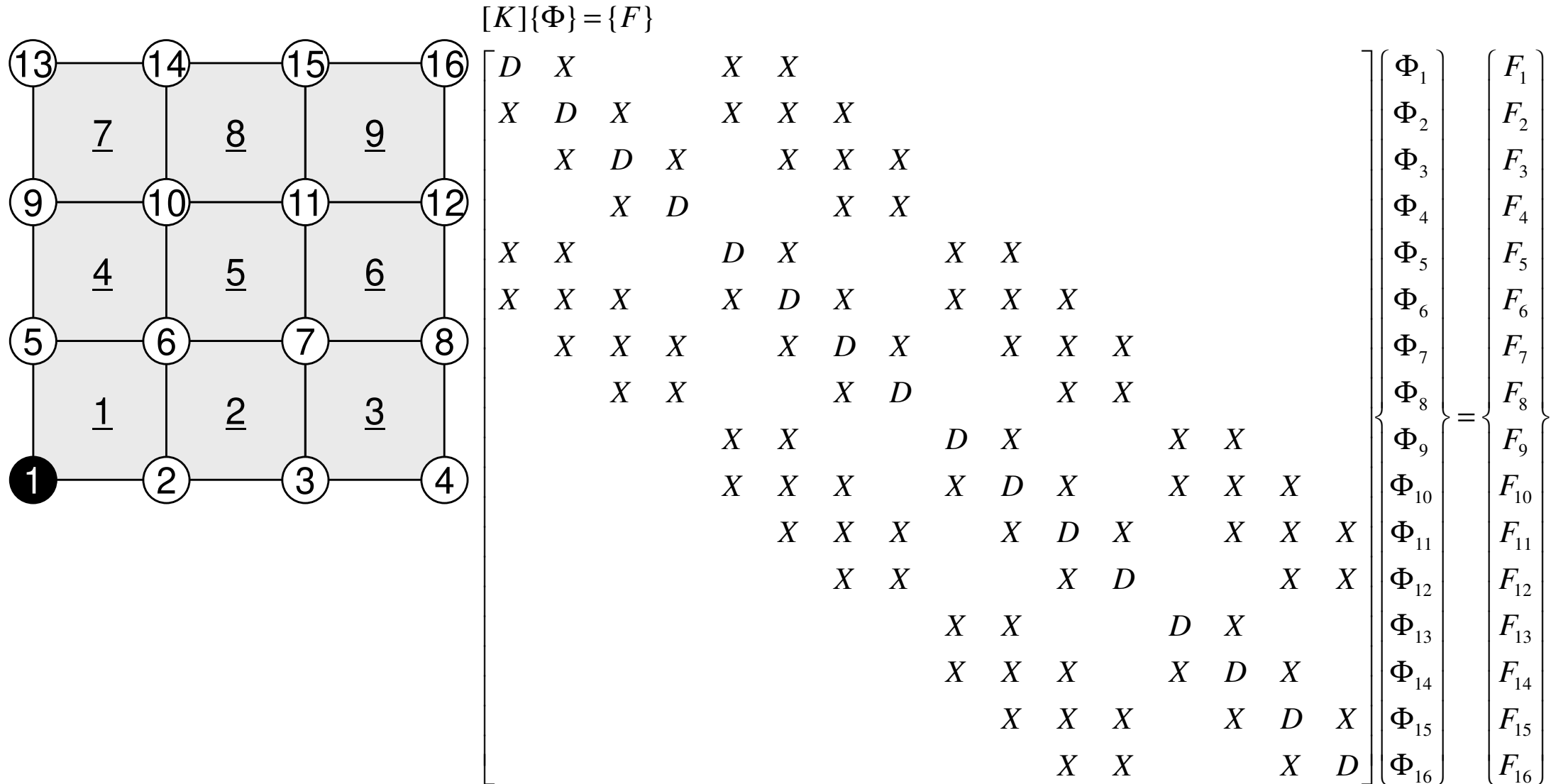
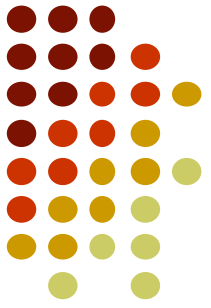
$$[k^{(e)}] \{\phi^{(e)}\} = \{f^{(e)}\}$$

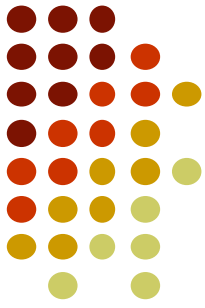
$$\begin{bmatrix} k_{AA}^{(e)} & k_{AB}^{(e)} & k_{AC}^{(e)} & k_{AD}^{(e)} \\ k_{BA}^{(e)} & k_{BB}^{(e)} & k_{BC}^{(e)} & k_{BD}^{(e)} \\ k_{CA}^{(e)} & k_{CB}^{(e)} & k_{CC}^{(e)} & k_{CD}^{(e)} \\ k_{DA}^{(e)} & k_{DB}^{(e)} & k_{DC}^{(e)} & k_{DD}^{(e)} \end{bmatrix} \begin{Bmatrix} \phi_A^{(e)} \\ \phi_B^{(e)} \\ \phi_C^{(e)} \\ \phi_D^{(e)} \end{Bmatrix} = \begin{Bmatrix} f_A^{(e)} \\ f_B^{(e)} \\ f_C^{(e)} \\ f_D^{(e)} \end{Bmatrix}$$



Global (Overall) Matrix

Accumulate each element matrix to “global” matrix.





To each node ...

Effect of surrounding elem's/nodes are accumulated.

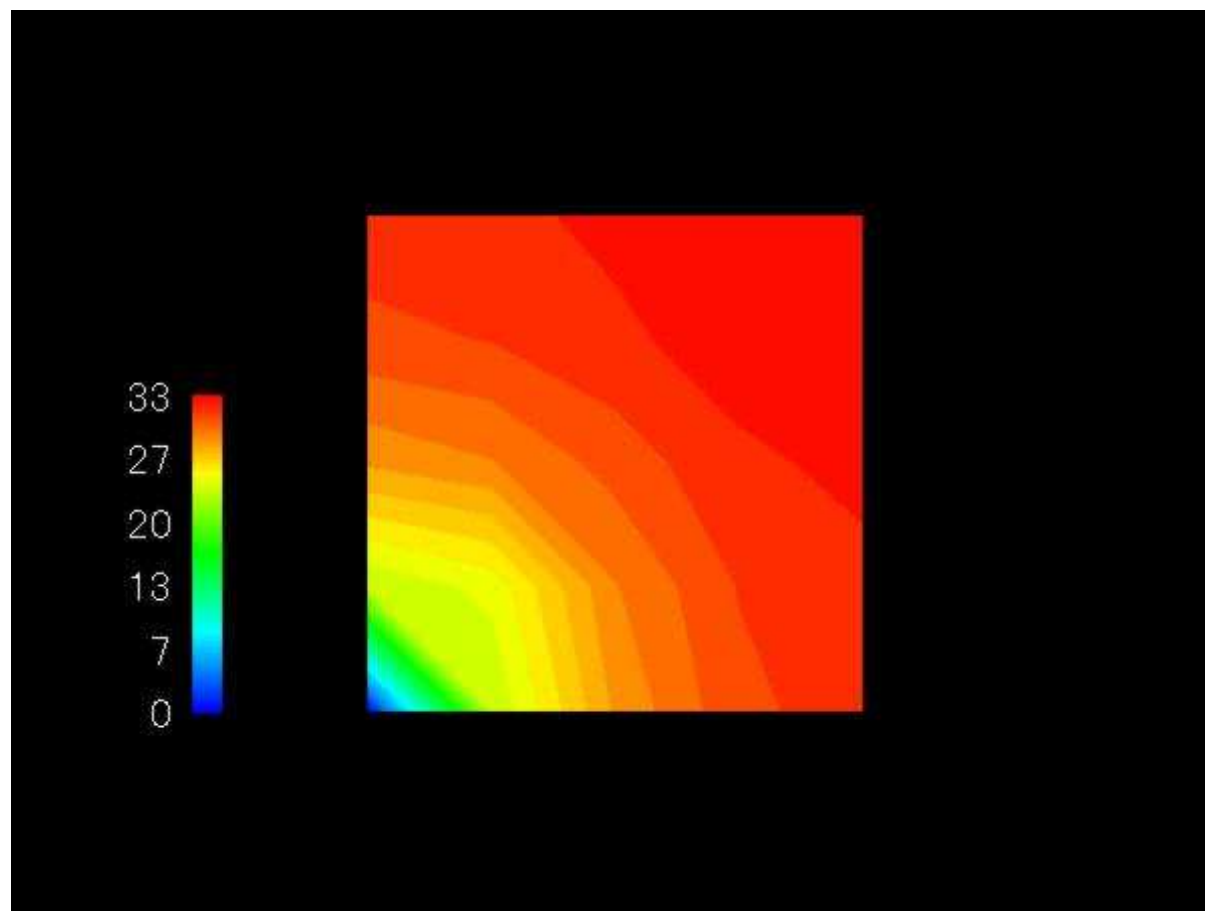
$[K]\{\Phi\} = \{F\}$

	D	X		X	X													
	X	D	X		X	X	X											
		X	D	X		X	X	X										
			X	D			X	X										
	X	X			D	X			X	X								
	X	X	X		X	D	X		X	X	X							
	X	X	X		X	D	X		X	X	X							
			X	X			X	D			X	X						
			X	X	X		X	D	X		X	X	X					
				X	X	X		X	D	X		X	X	X				
					X	X				D	X							
						X	X		X	D	X							
							X	X			D	X						

}	Φ_1	=	F_1
	Φ_2		F_2
	Φ_3		F_3
	Φ_4		F_4
	Φ_5		F_5
	Φ_6		F_6
	Φ_7		F_7
	Φ_8		F_8
	Φ_9		F_9
	Φ_{10}		F_{10}
	Φ_{11}		F_{11}
	Φ_{12}		F_{12}
	Φ_{13}		F_{13}
	Φ_{14}		F_{14}
	Φ_{15}		F_{15}
	Φ_{16}		F_{16}

Coefficient matrix is very "sparse",
many "0's"
Contributions from only neighbors

Result ...



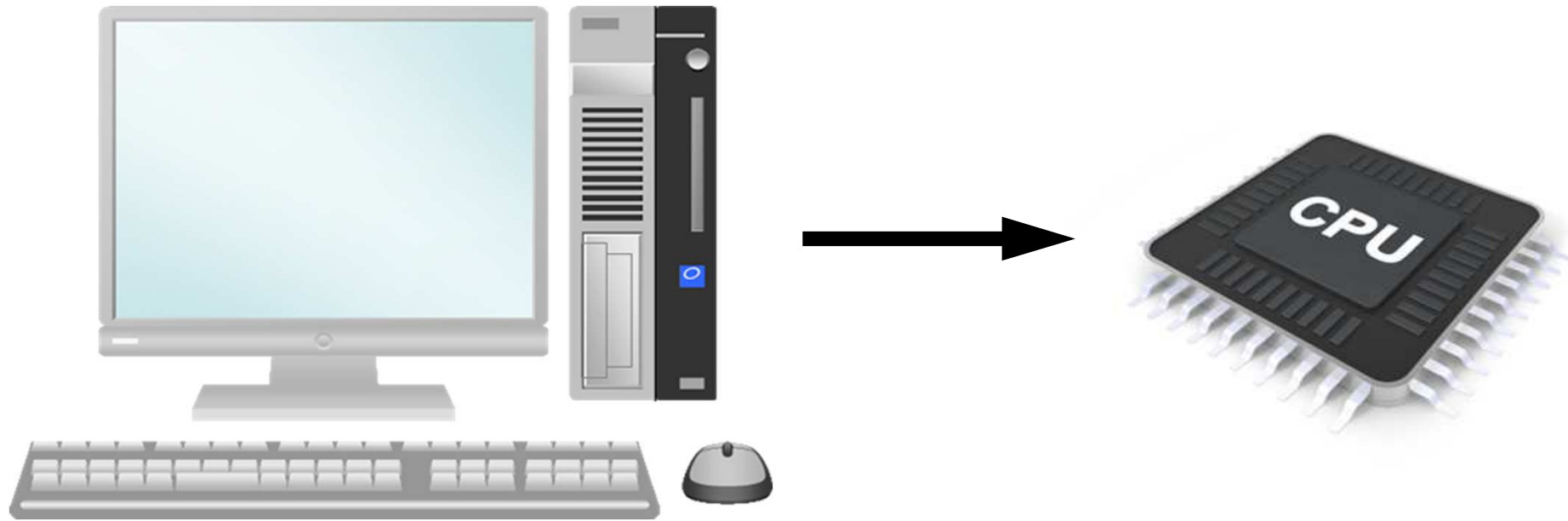


Features of FEM applications

- Typical Procedures for FEM Computations
 - Input/Output
 - Matrix Assembling
 - Linear Solvers for Large-scale Sparse Matrices
 - Most of the computation time is spent for matrix assembling/formation and solving linear equations.
- **HUGE** “indirect” accesses
 - memory intensive
- Local “element-by-element” operations
 - sparse coefficient matrices
 - suitable for parallel computing
- Excellent modularity of each procedure

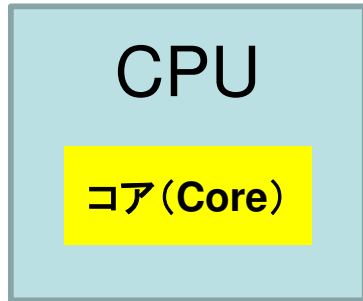
- Target
 - Parallel FEM
 - Introduction to AI/DNN (Deep Neural Network)
- **Supercomputers and Computational Science**
- Overview of the Class
- Future Issues

Computer & CPU

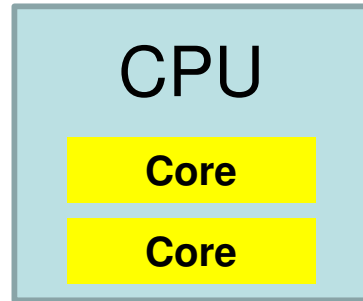


- Central Processing Unit (中央处理装置): CPU
- CPU's used in PC and Supercomputers are based on same architecture
- GHz: Clock Rate
 - Frequency: Number of operations by CPU per second
 - GHz -> 10^9 operations/sec
 - Simultaneous 4-8 (or more) instructions per clock

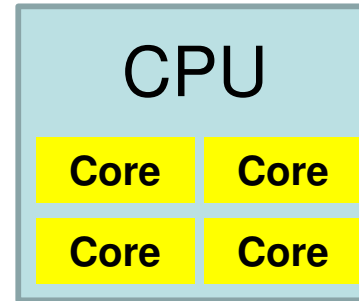
Multicore CPU



Single Core
1 cores/CPU



Dual Core
2 cores/CPU

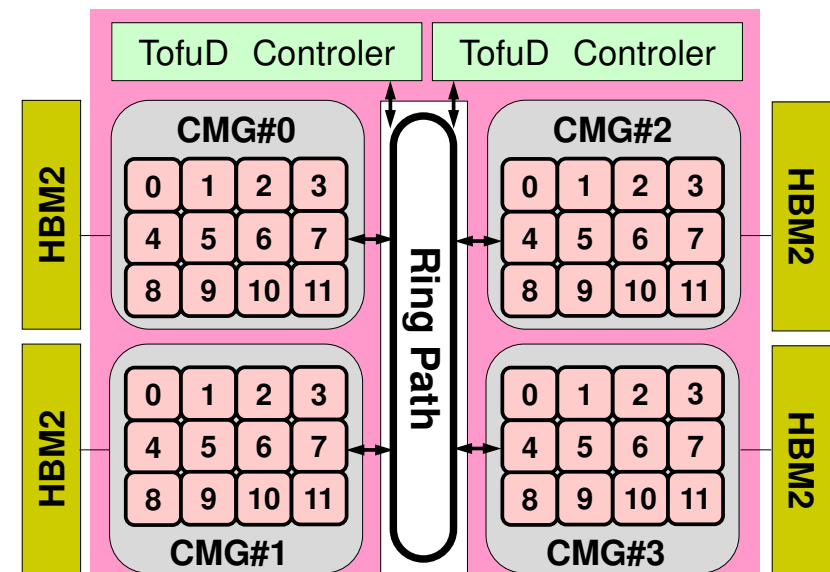


Quad Core
4 cores/CPU

- Core= Central part of CPU
- Multicore CPU's with 4-8 cores are popular
 - Low Power

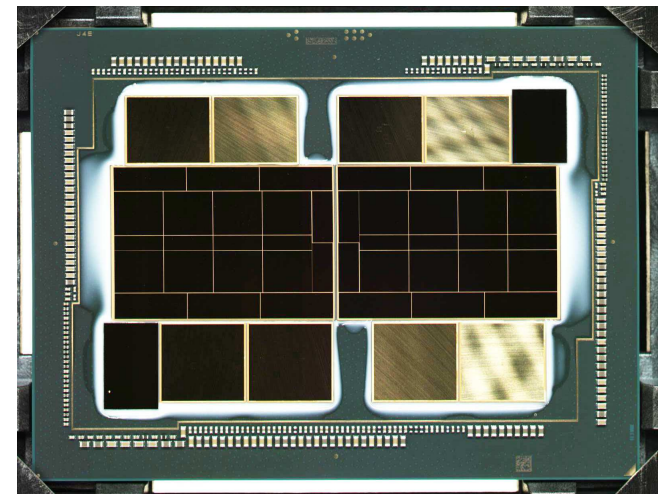
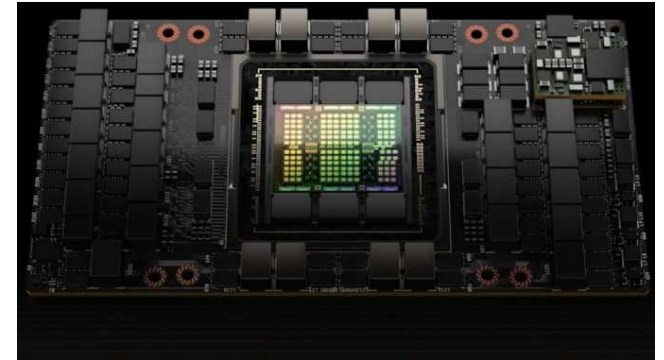
- GPU: Manycore
 - $O(10^1)$ - $O(10^2)$ cores
- More and more cores
 - Parallel computing

- **Fugaku: 48-cores/node**
 - **Fujitsu/ARM A64FX**



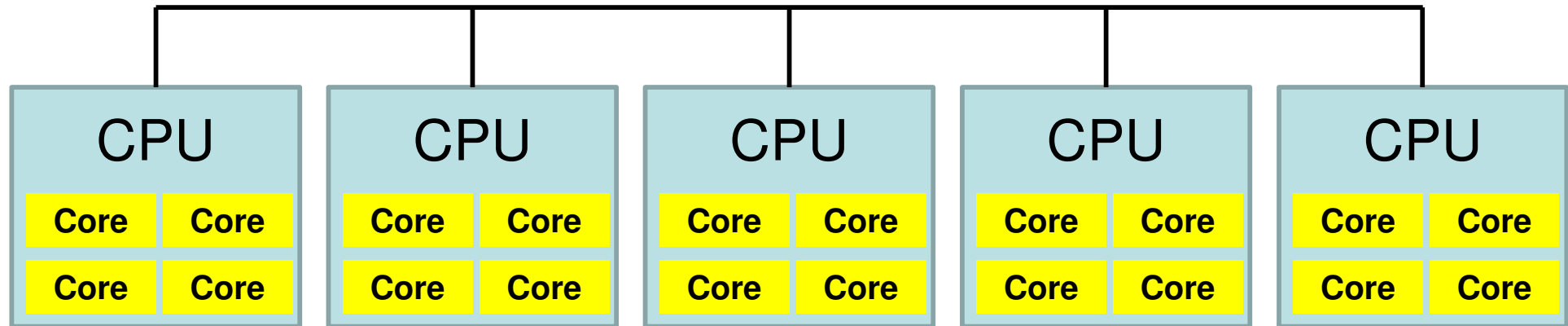
GPU/Accelerators

- GPU: Graphic Processing Unit
 - GPGPU: General Purpose GPU
 - $O(10^2)$ cores
 - High Memory Bandwidth
 - (was) cheap
 - NO stand-alone operations
 - Host CPU needed
 - Programming: CUDA, OpenACC etc.
- NVIDIA, AMD, Intel ...

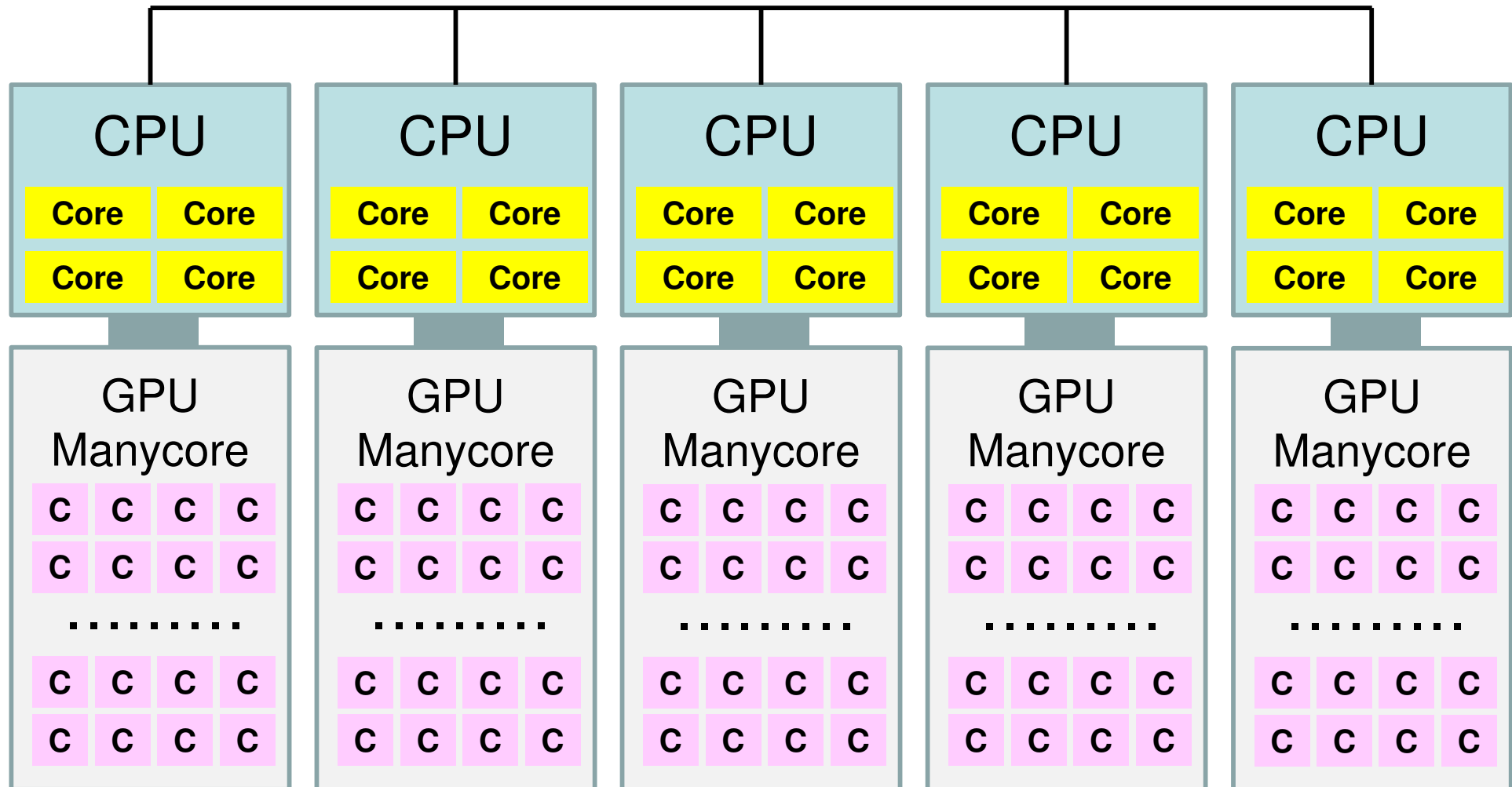


Parallel Supercomputers

Multicore CPU's are connected through network



Supercomputers with Heterogeneous/Hybrid Nodes

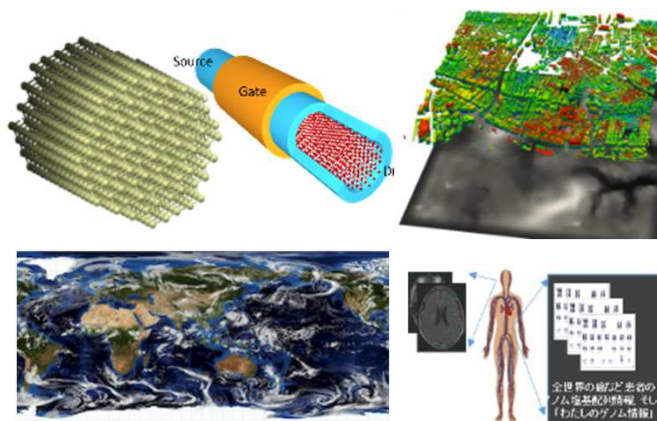
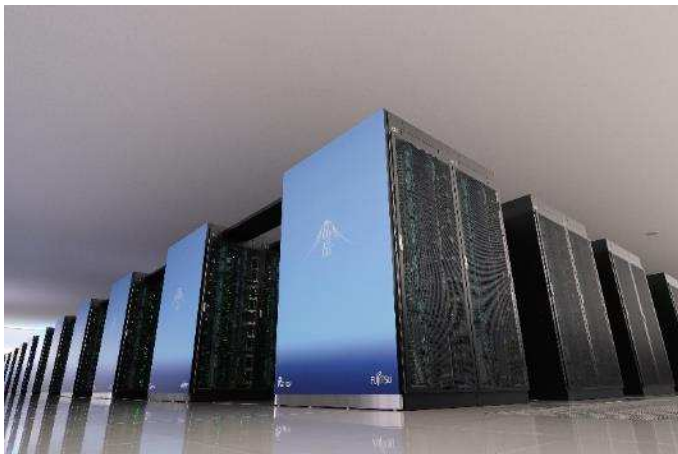


Performance of Supercomputers

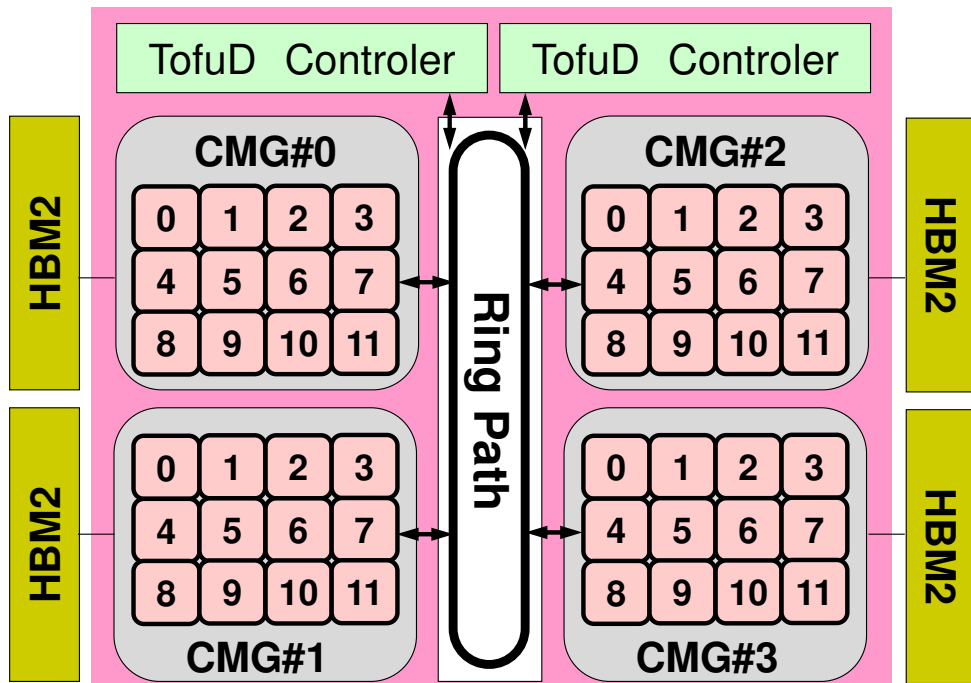
- Performance of CPU: Clock Rate
- FLOPS (Floating Point Operations per Second)
 - Real Number
- Recent Multicore CPU
 - 4-8 FLOPS per Clock
 - (e.g.) Peak performance of a core with 3GHz
 - $3 \times 10^9 \times 4(\text{or } 8) = 12(\text{or } 24) \times 10^9 \text{ FLOPS} = 12(\text{or } 24) \text{ GFLOPS}$
 - $10^6 \text{ FLOPS} = 1 \text{ Mega FLOPS} = 1 \text{ MFLOPS}$
 - $10^9 \text{ FLOPS} = 1 \text{ Giga FLOPS} = 1 \text{ GFLOPS}$
 - $10^{12} \text{ FLOPS} = 1 \text{ Tera FLOPS} = 1 \text{ TFLOPS}$
 - $10^{15} \text{ FLOPS} = 1 \text{ Peta FLOPS} = 1 \text{ PFLOPS}$
 - $10^{18} \text{ FLOPS} = 1 \text{ Exa FLOPS} = 1 \text{ EFLOPS}$

Supercomputer “Fugaku”

- **Ultra-scale “general-purpose” manycore system: 158,976 nodes (1 processor/node, total 7.6 M cores, theoretical peak 537PFLOPS (DP))**
- **Arm-based manycore processor: Fujitsu A64FX (Armv8.2-A SVE 512bit SIMD, #core 48 + 2/4, 3TF@2.0GHz, boost to 2.2GHz)**
 - 12 cores in a cluster of cores called CMG, connected to L2 and HBM memory chips
- **Advanced Memory technology: HBM2 32 GiB, 1024 GB/s bandwidth, packaged in CPU chip**
- **Scalable Interconnect: ToFu-D interconnect**



A64FX Processor on Fugaku



Name	A64FX
Processor # (Core #)	1 (48+ 2or4 Assistant Cores)
Frequency	2.2 GHz
Peak Performance	3.3792 TFLOPS
Memory Size	32 GiB
Memory Bandwidth	1,024 GB/s
L1 Cache	64 KiB/core (Inst/Data)
L2 Cache	8 MiB/CMG

- 4 CMG's (Core Memory Group), 12 cores/CMG
 - 48 Cores/Node (Processor)
 - $2.2\text{GHz} \times 32\text{DP} \times 48 = 3379.2 \text{ GFLOPS} = 3.3792 \text{ TFLOPS}$
- NUMA Architecture (Non-Uniform Memory Access)
 - Each core of a CMG can access to the memory on other CMG's
 - Utilization of the local memory is more efficient

TOP 500 List

<http://www.top500.org/>

- Ranking list of supercomputers in the world
- Performance (FLOPS rate) is measured by “Linpack” which solves large-scale linear equations.
 - Since 1993
 - Updated twice a year (International Conferences in June and November)
- **“Fugaku” has been #1 from June 2020 to Nov.2021 (4 times)**
- **“Frotier” is the 1st Exaflop System in June 2022**
- Linpack
 - iPhone version is available

10^{19} = 10 ExaFlops

10^{18} = 1 ExaFlops

10^{17} = 100 PetaFlops

10^{16} = 10 PetaFlops

10^{15} = 1 PetaFlops

10^{14} = 100 TeraFlops

10^{13} = 10 TeraFlops

10^{12} = 1 TeraFlops

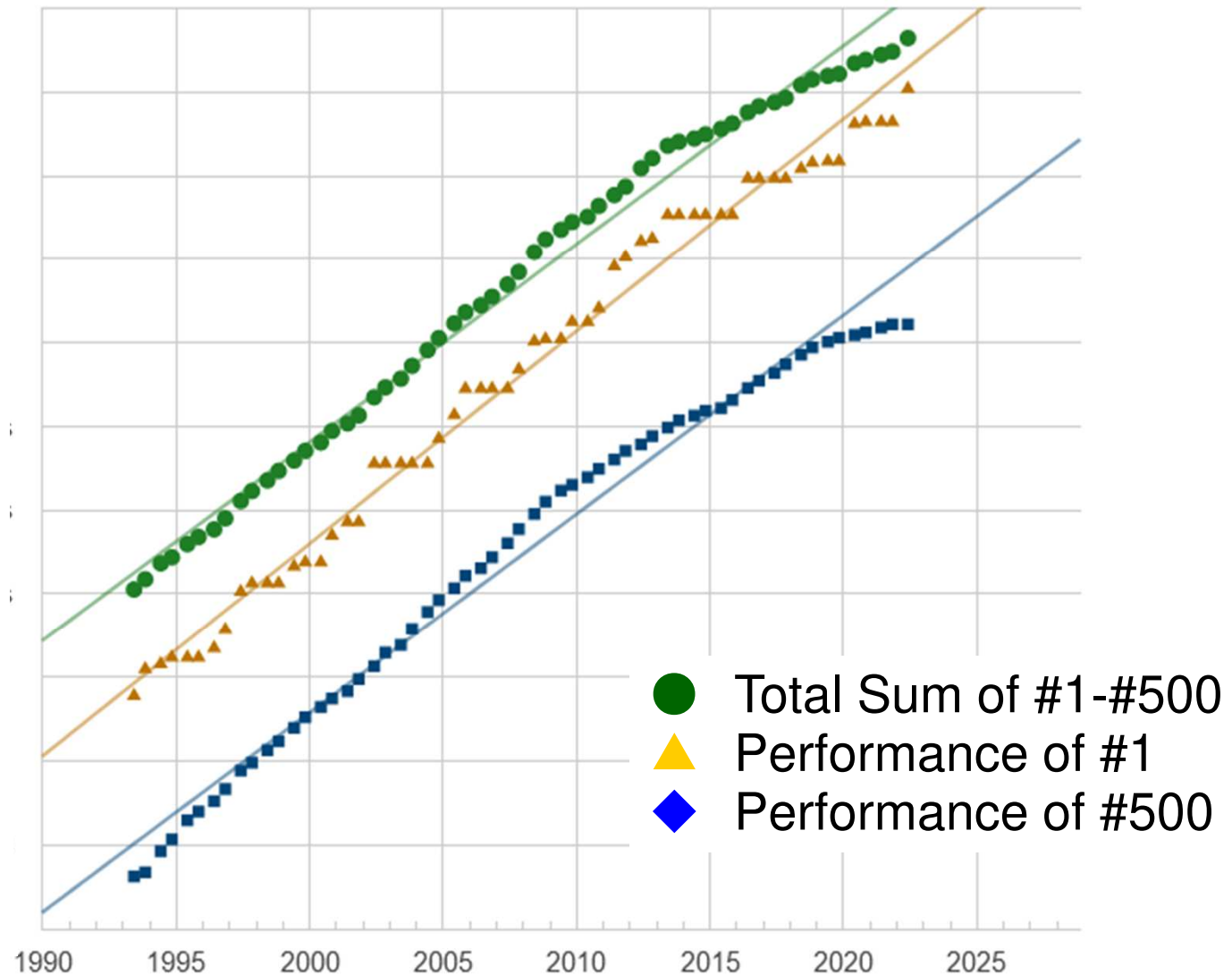
10^{11} = 100 GigaFlops

10^{10} = 10 GigaFlops

10^9 = 1 GigaFlops

10^8 = 100 MegaFlops

Projected Performance Development



59th TOP500 List (June, 2022)

<http://www.top500.org/>

R_{max} : Performance of Linpack (TFLOPS)
 R_{peak} : Peak Performance (TFLOPS),
 Power: kW

	Site	Computer/Year Vendor	Cores	R_{max} (PFLOPS)	R_{peak} (PFLOPS)	Power (kW)
1	<u>Frontier, 2022, USA</u> DOE/SC/Oak Ridge National Laboratory	HPE Cray EX235a, AMD Optimized 3 rd Gen. EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	8,730,112	1,102.00 (=1.102 EF)	1,685.65	21,100
2	<u>Fugaku, 2020, Japan</u> R-CCS, RIKEN	Fujitsu PRIMEHPC FX1000, Fujitsu A64FX 48C 2.2GHz, Tofu-D	7,630,848	442,010 (= 442.0 PF)	537,212.0	29,899
3	<u>LUMI, 2022, Finland</u> EuroHPC/CSC	HPE Cray EX235a, AMD Optimized 3 rd Gen. EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	1,110,144	151.90	214.35	2,942
4	<u>Summit, 2018, USA</u> DOE/SC/Oak Ridge National Laboratory	IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR InfiniBand	2,414,592	148.60	200.79	10,096
5	<u>Sierra, 2018, USA</u> DOE/NNSA/LLNL	IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR InfiniBand	1,572,480	94.64	125.71	7,438
6	<u>Sunway TaihuLight, 2016, China</u> National Supercomputing Center in Wuxi	Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway	10,649,600	93.01	125.44	15,371
7	<u>Perlmutter, 2021, USA</u> DOE/NERSC/LBNL	HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10	761,856	70.87	93.75	2,528
8	<u>Selene, 2020, USA</u> NVIDIA	NVIDIA DGX A100 SuperPOD, AMD EPYC 7742 64C 2.25GHz, NVIDIA GA100, Mellanox Infiniband HDR	555,520	63.46	79.22	2,646
9	<u>Tianhe-2A, 2018, China</u> National Super Computer Center in Guangzhou	TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000	4,981,760	61.44	100.68	18,482
10	<u>Adastra, 2022, France</u> GENCI-CINES	HPE Cray EX235a, AMD Optimized 3 rd Gen. EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11	319,072	46.10	61.61	921

Linpack on My iPhone XS



Cray-1S

Normal Mode

Low-Power Mode

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10

Mflop/s: 18800.05

Time: 0.0364

Norm Res: 5.1700

Precision: 2.22044605e-16

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10

Mflop/s: 8359.34

Time: 0.0726

Norm Res: 5.1700

Precision: 2.22044605e-16

Multithread (parallel)

Max Mflop/s: 20627.07
Avg Mflop/s: 17294.78

Max Mflop/s: 11868.06
Avg Mflop/s: 9329.87

20:29

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10

Mflop/s: 5511.57

Time: 0.0152

Norm Res: 5.1700

Precision: 2.22044605e-16

20:28

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10

Mflop/s: 3737.27

Time: 0.0224

Norm Res: 5.1700

Precision: 2.22044605e-16

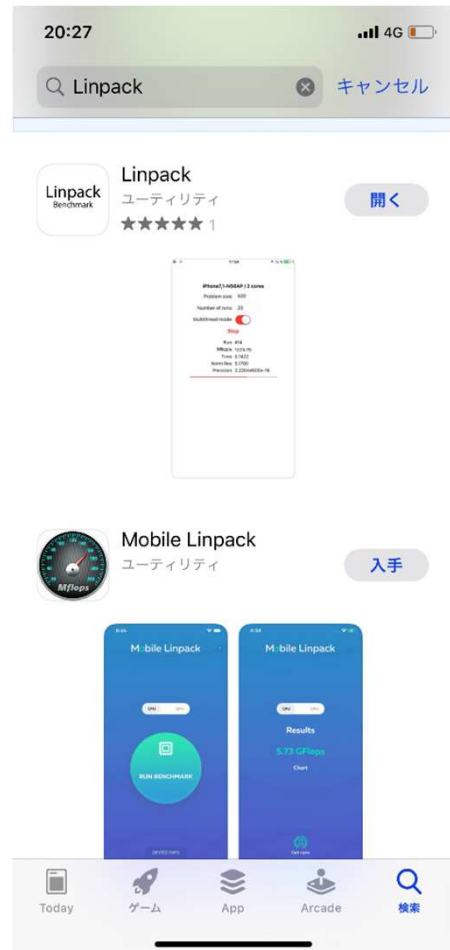
Single-thread (serial)

Max Mflop/s: 5521.89
Avg Mflop/s: 4921.39

Max Mflop/s: 3769.22
Avg Mflop/s: 3586.08

- Performance of my iPhone XS is about 20,000 Mflops
 - Fugaku: 3.83 Tflops
- Cray-1S
 - Supercomputer of my company in 1985 with 80 Mflops
 - I do not know the price, but we had to pay 10 USD for 1 sec. computing !

Linpack on My iPhone XS



Normal Mode

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10
Mflop/s: 18800.05
Time: 0.0364
Norm Res: 5.1700
Precision: 2.22044605e-16

Max Mflop/s: 20627.07
Avg Mflop/s: 17294.78

Low-Power Mode

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10
Mflop/s: 8359.34
Time: 0.0726
Norm Res: 5.1700
Precision: 2.22044605e-16

Max Mflop/s: 11868.06
Avg Mflop/s: 9329.87

20:29

ライブドアニュース **インストール**

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10
Mflop/s: 5511.57
Time: 0.0152
Norm Res: 5.1700
Precision: 2.22044605e-16

Max Mflop/s: 5521.89
Avg Mflop/s: 4921.39

20:28

App Store

ライブドアニュース **インストール**

iPhone11,2-D321AP / 6 cores

Problem size: 500

Number of runs: 10

Multithread mode:

Run benchmark

Run: #10
Mflop/s: 3737.27
Time: 0.0224
Norm Res: 5.1700
Precision: 2.22044605e-16

Max Mflop/s: 3769.22
Avg Mflop/s: 3586.08

- You can change Problem size, and # of runs.
 - “Size=500” means linear equations $Ax=b$ with 500 unknowns are solved
- Actually, problem size affects performance of computing so much !!

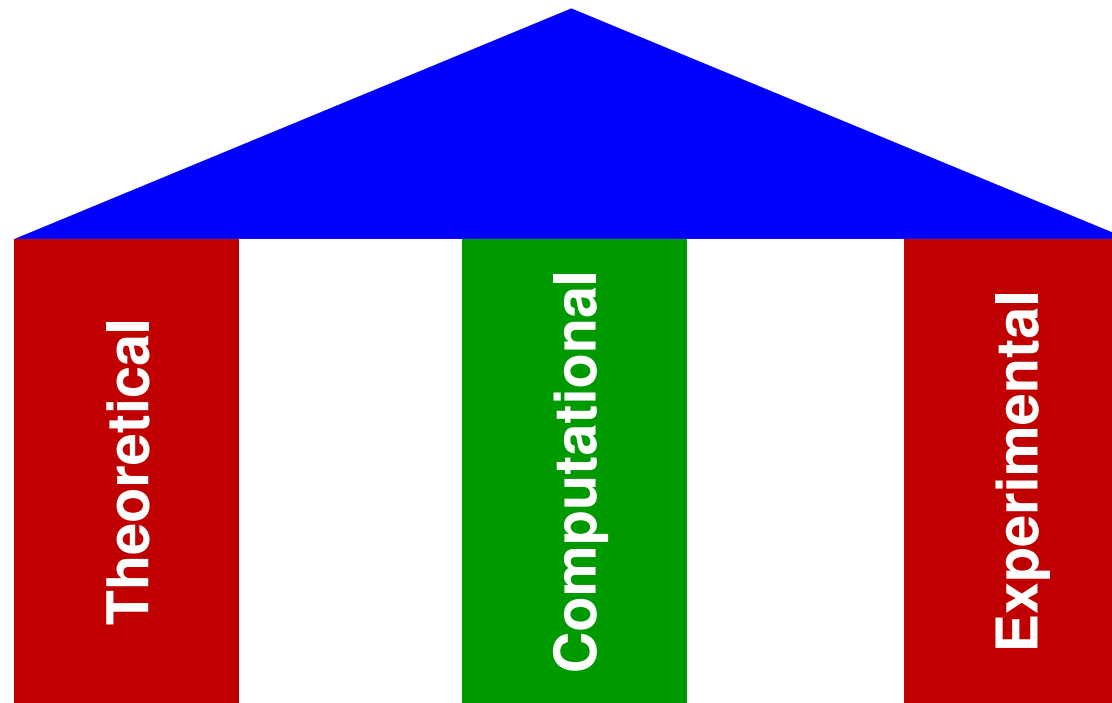
Benchmarks

- TOP 500 (Linpack, HPL(High Performance Linpack))
 - Direct Linear Solvers, FLOPS rate
 - Regular Dense Matrices, Continuous Memory Access
 - Computing Performance
- HPCG
 - Preconditioned Iterative Solvers, FLOPS rate
 - Irregular Sparse Matrices derived from FEM Applications with Many “0” Components
 - Irregular/Random Memory Access,
 - Closer to “Real” Applications than HPL
 - Performance of Memory, Communications
- Green 500
 - FLOPS/W rate for HPL (TOP500)

Computational Science

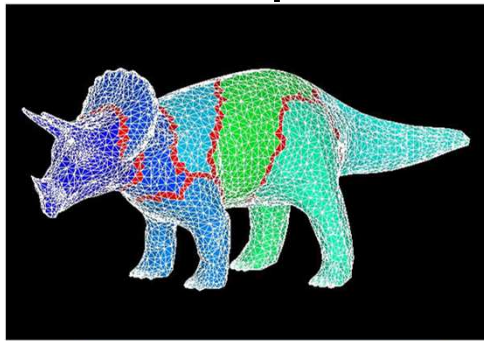
The 3rd Pillar of Science

- Theoretical & Experimental Science
- Computational Science
 - The 3rd Pillar of Science
 - Simulations using Supercomputers

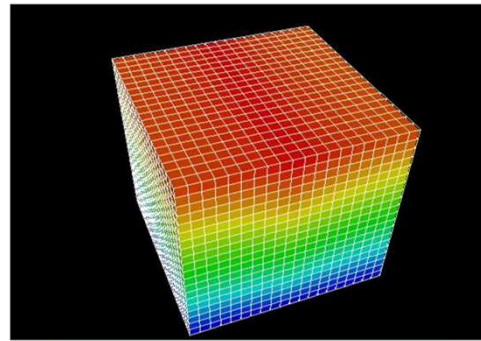


Methods for Scientific Computing

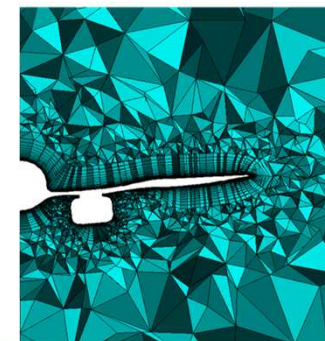
- Numerical solutions of PDE (Partial Diff. Equations)
- Grids, Meshes, Particles
 - Large-Scale Linear Equations
 - Finer meshes provide more accurate solutions



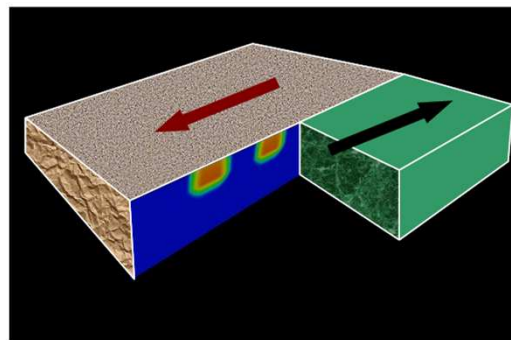
有限要素法
Finite Element Method
FEM



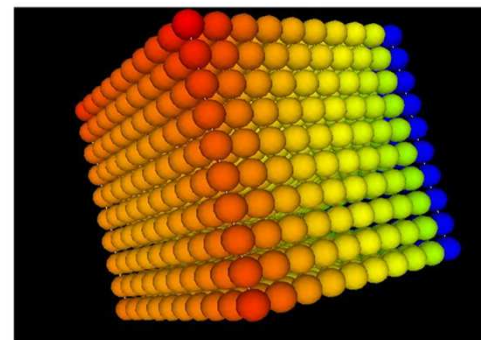
差分法
Finite Difference Method
FDM



有限体積法
Finite Volume Method
FVM



境界要素法
Boundary Element Method
BEM

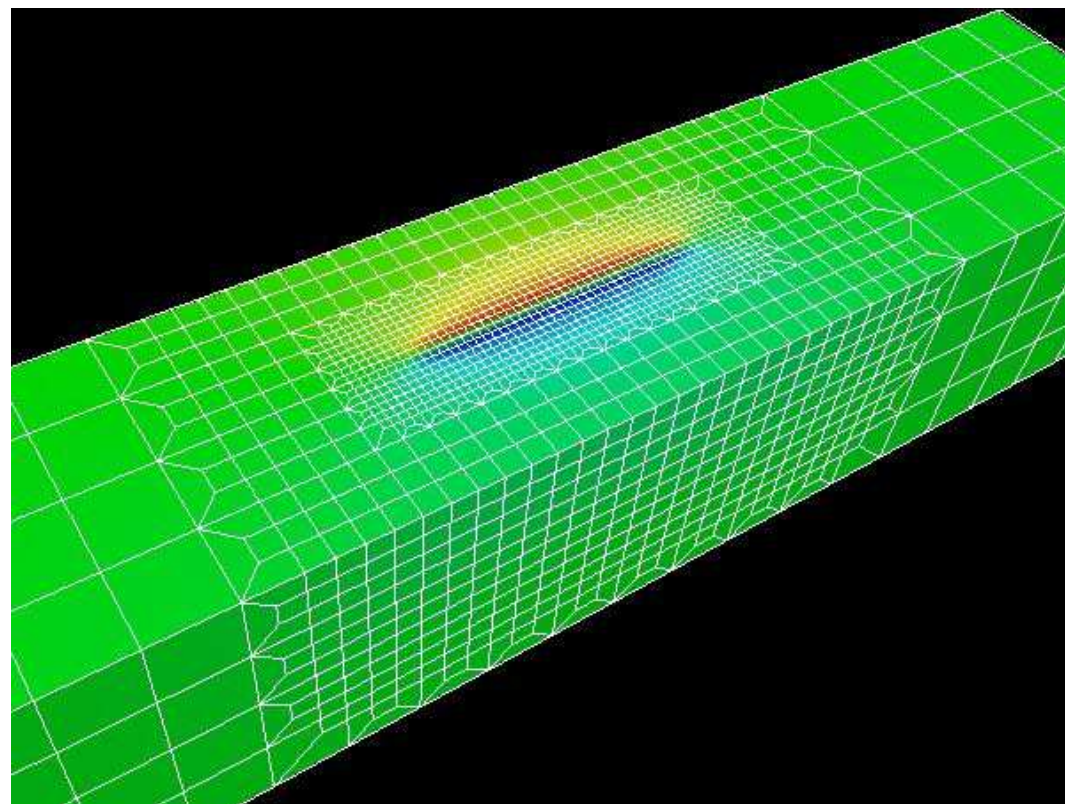
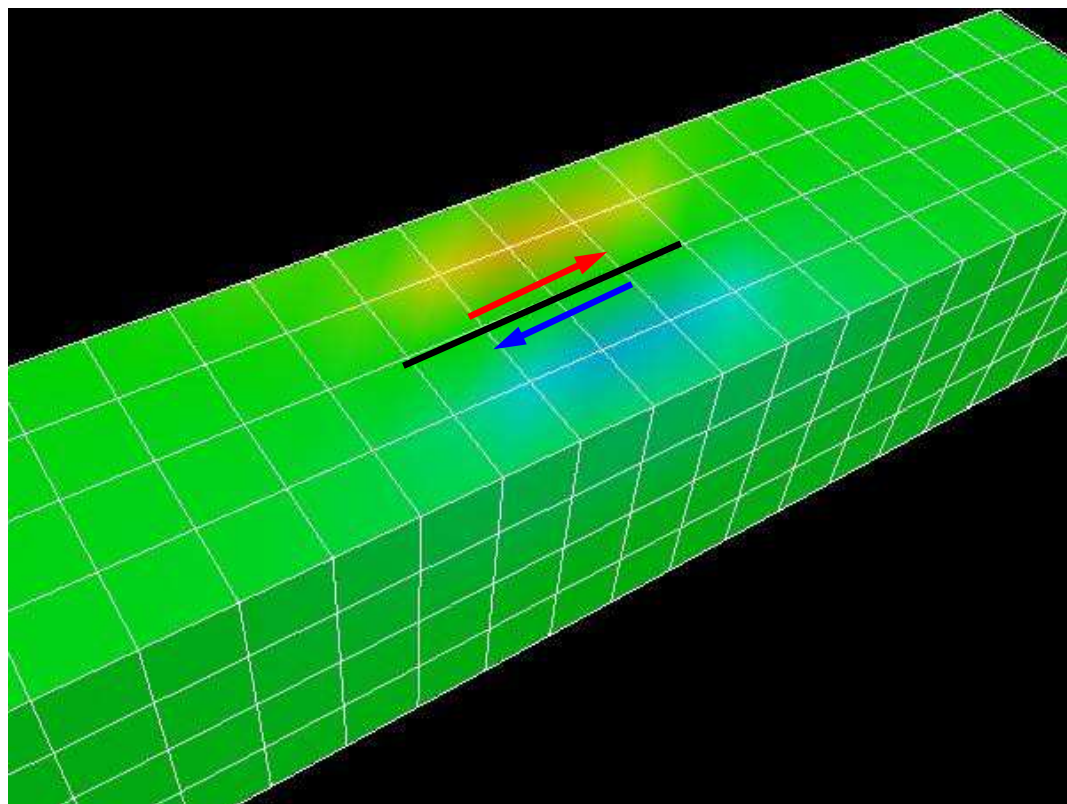


個別要素法
Discrete Element Method
DEM

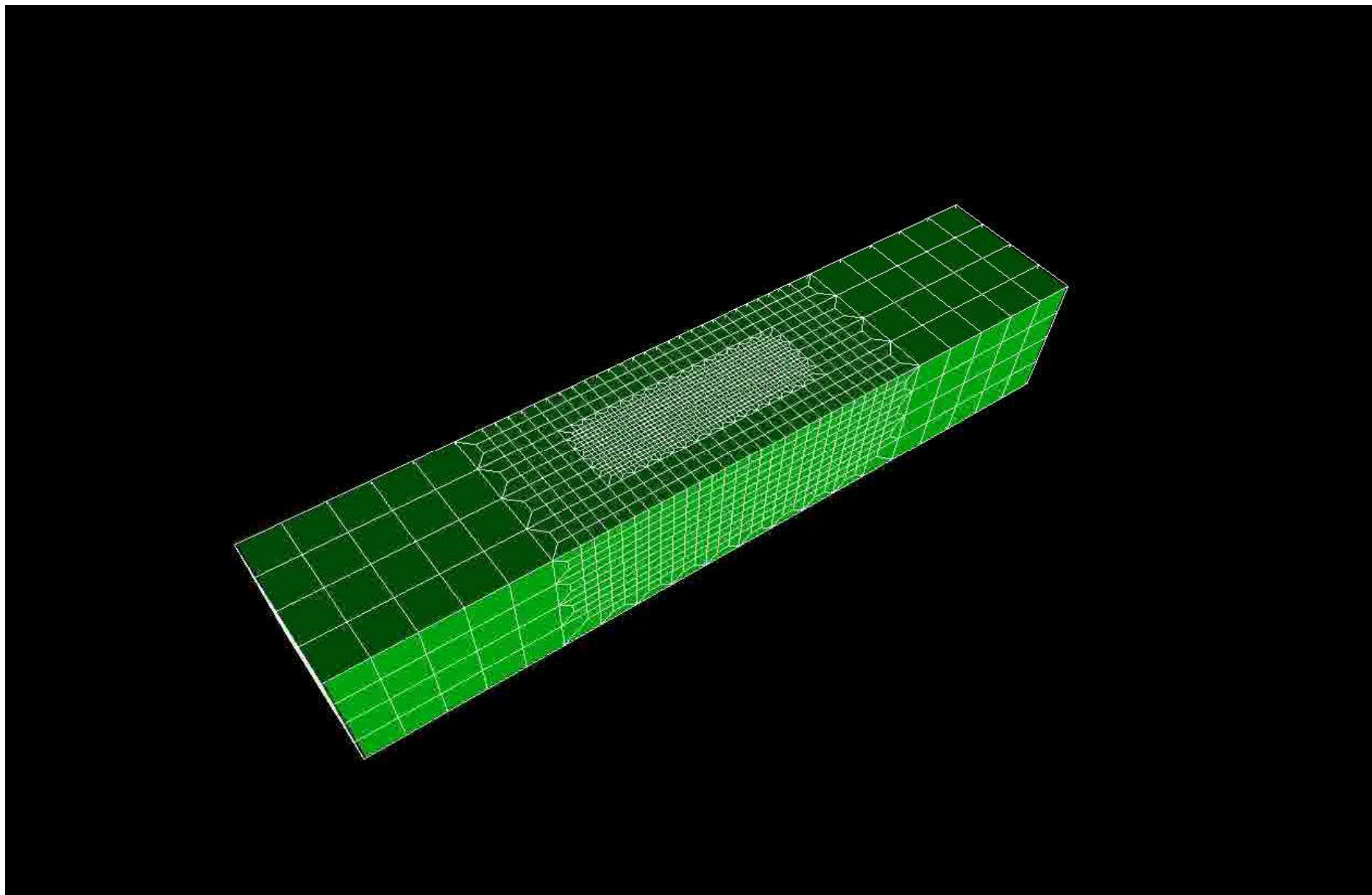
3D Simulations for Earthquake Generation Cycle

San Andreas Faults, CA, USA

Stress Accumulation at Transcurrent Plate Boundaries

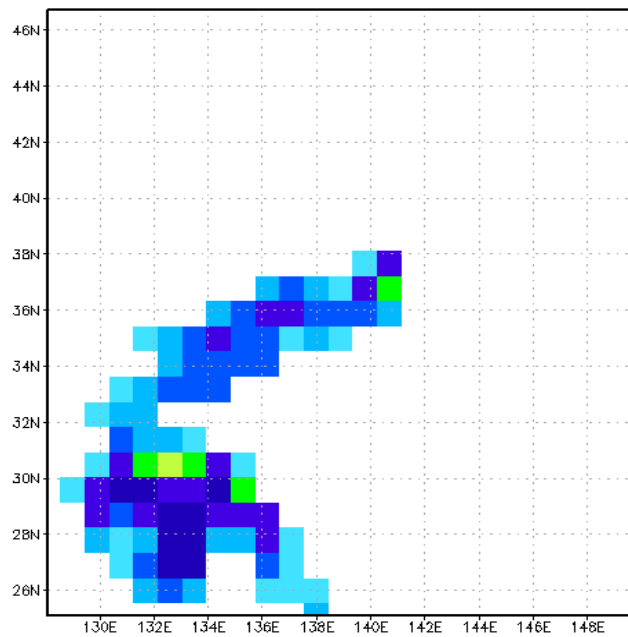


Adaptive FEM: High-resolution needed at meshes with large deformation (large accumulation)

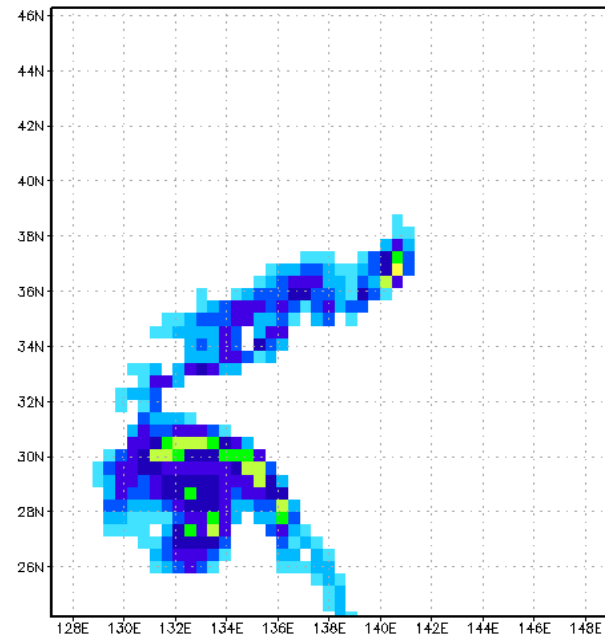


Typhoon Simulations by FDM

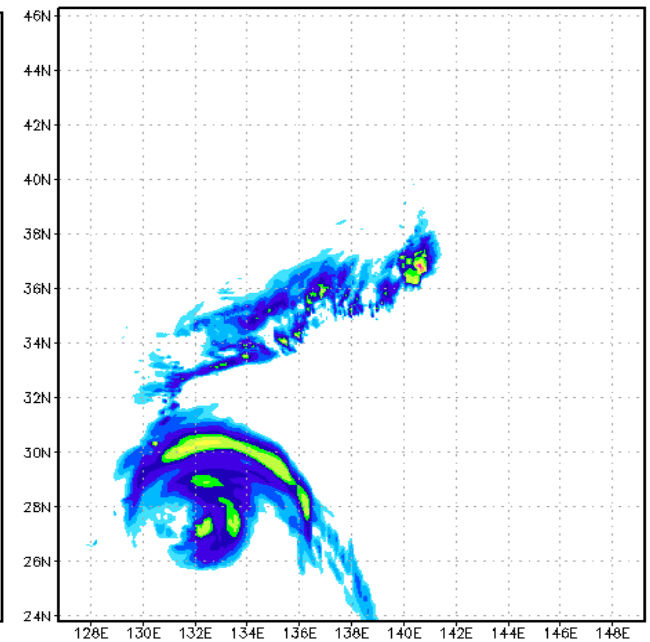
Effect of Resolution



$\Delta h = 100\text{km}$



$\Delta h = 50\text{km}$



$\Delta h = 5\text{km}$

Simulation of Geologic CO₂ Storage

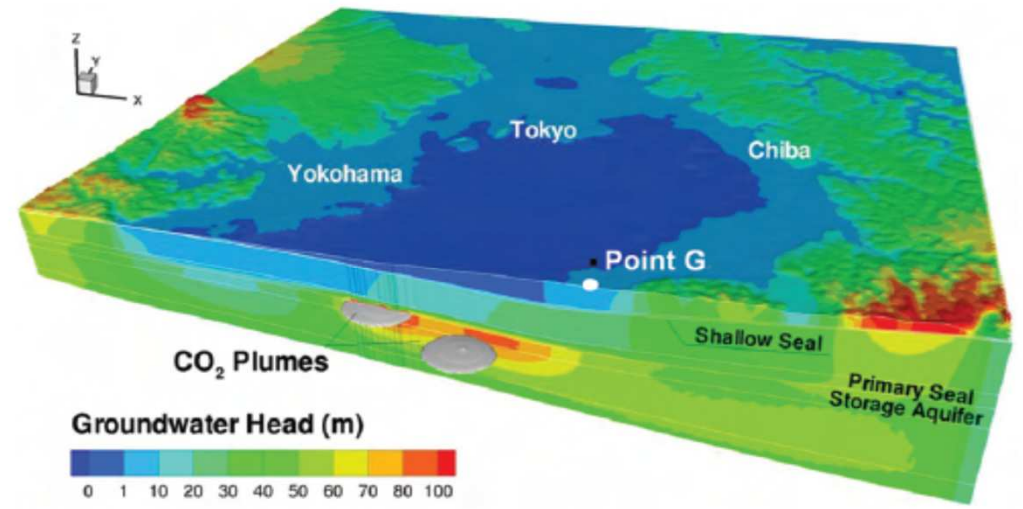
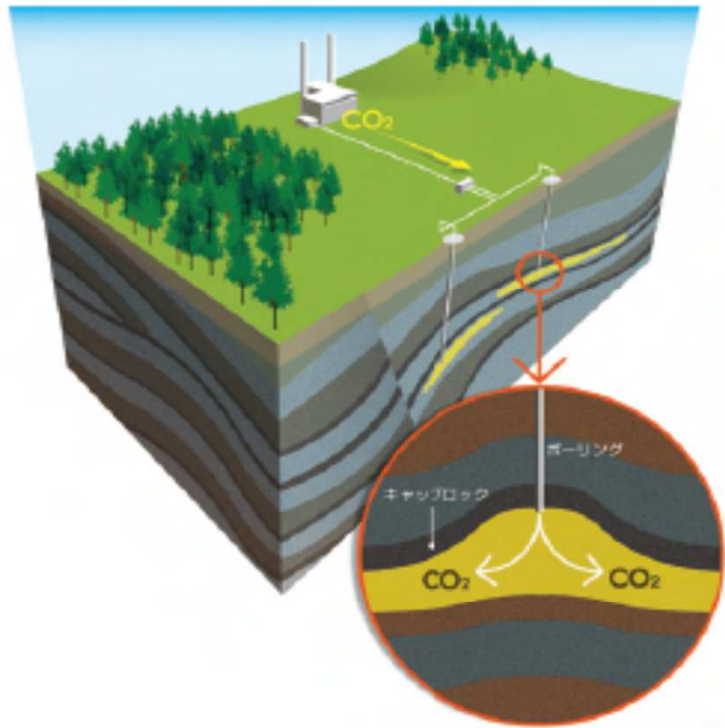
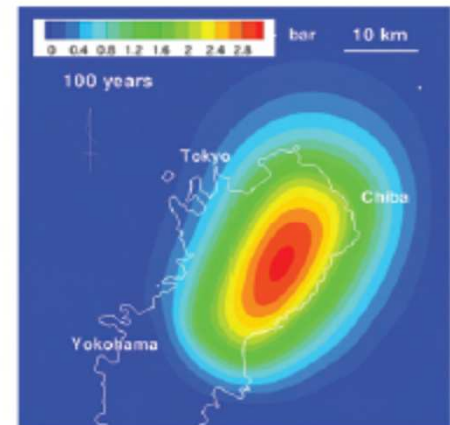
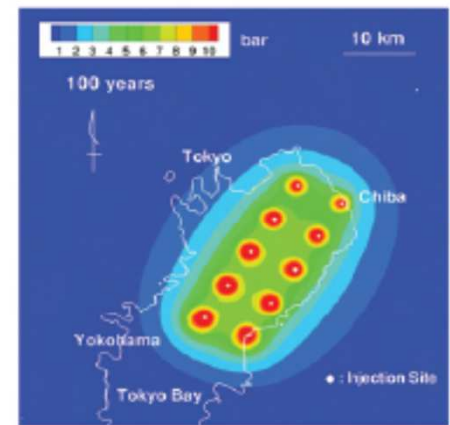
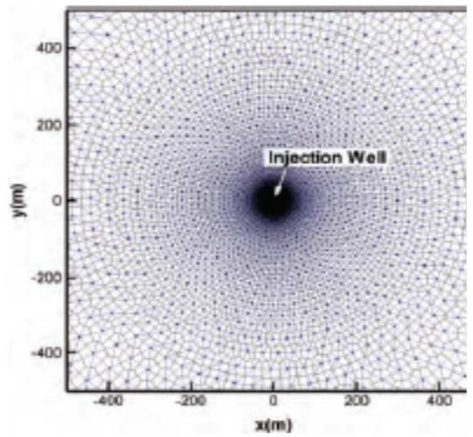
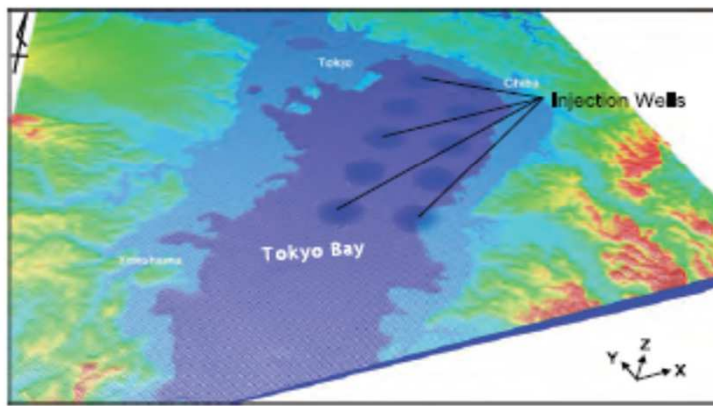


図-4 CO₂ 圧入後の地下水圧 (全水頭換算) の分布 (100 年後)



(a) 深部遮蔽層下面

(b) 浅部遮蔽層下面

図-5 圧力上昇量の平面分布 (初期状態からの増分、圧入開始から 100 年後)

[Dr. Hajime Yamamoto, Taisei]

Simulation of Geologic CO₂ Storage

- International/Interdisciplinary Collaborations
 - Taisei (Science, Modeling)
 - Lawrence Berkeley National Laboratory, USA (Modeling)
 - Information Technology Center, the University of Tokyo (Algorithm, Software)
 - JAMSTEC (Earth Simulator Center) (Software, Hardware)
 - NEC (Software, Hardware)
- 2010 Japan Geotechnical Society (JGS) Award

Science

Modeling

Algorithm

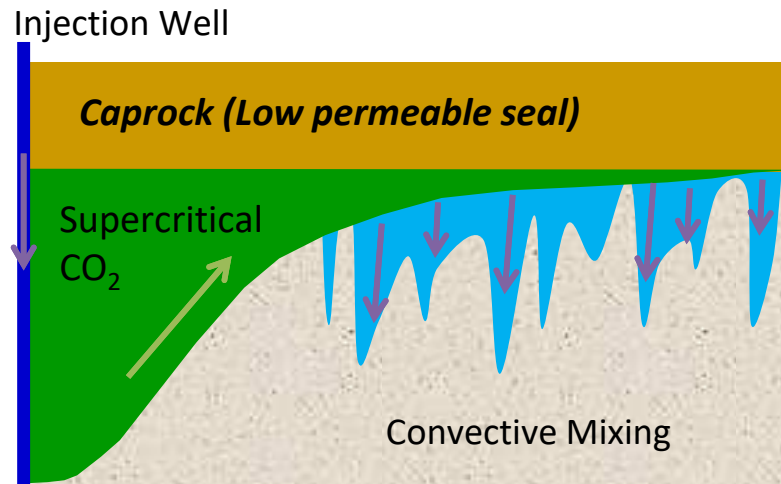
Software

Hardware

Simulation of Geologic CO₂ Storage

- Science
 - Behavior of CO₂ in supercritical state at deep reservoir
- PDE's
 - 3D Multiphase Flow (Liquid/Gas) + 3D Mass Transfer
- Method for Computation
 - TOUGH2 code based on FVM, and developed by Lawrence Berkeley National Laboratory, USA
 - More than 90% of computation time is spent for solving large-scale linear equations with more than 10⁷ unknowns
- Numerical Algorithm
 - Fast algorithm for large-scale linear equations developed by Information Technology Center, the University of Tokyo
- Supercomputer
 - Earth Simulator II (NEX SX9, JAMSTEC, 130 TFLOPS)
 - Oakleaf-FX (Fujitsu PRIMEHP FX10, U.Tokyo, 1.13 PFLOPS)

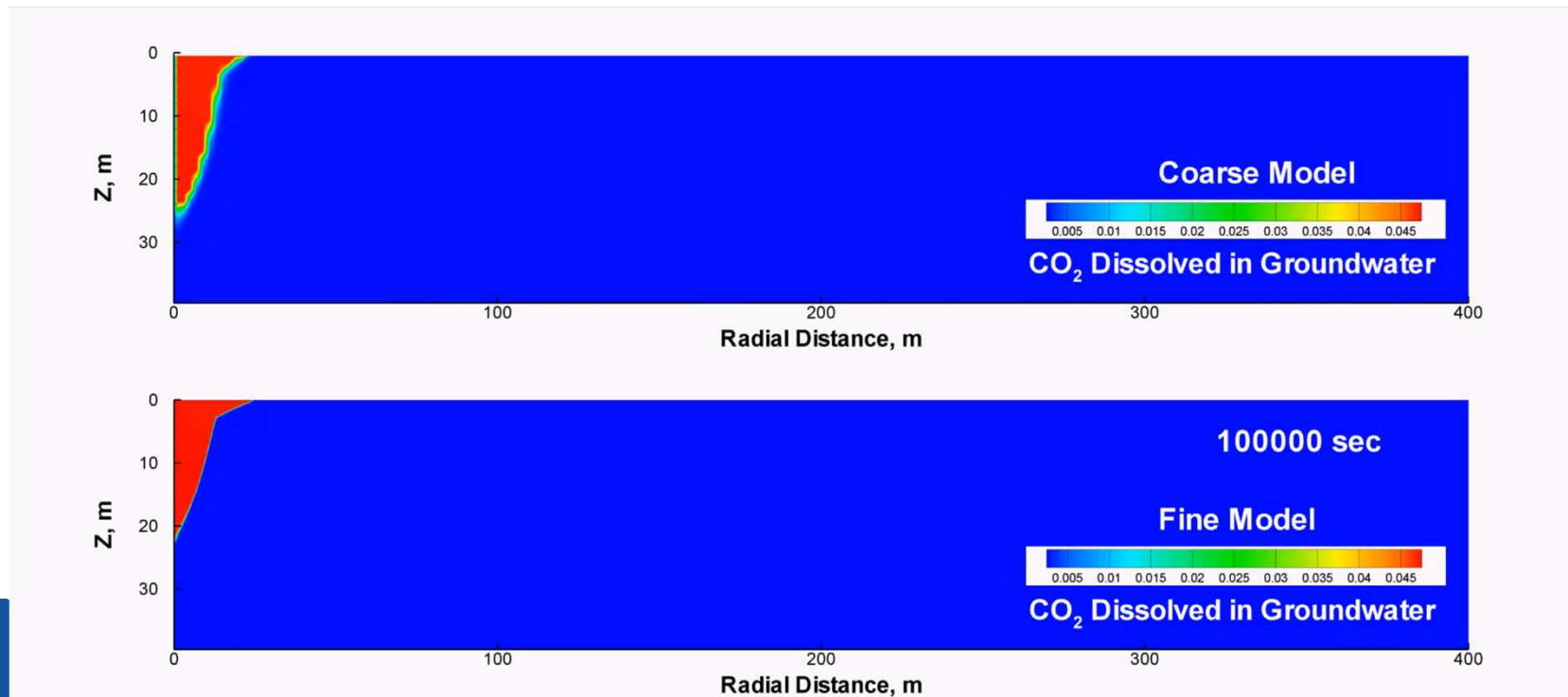
Diffusion-Dissolution-Convection Process

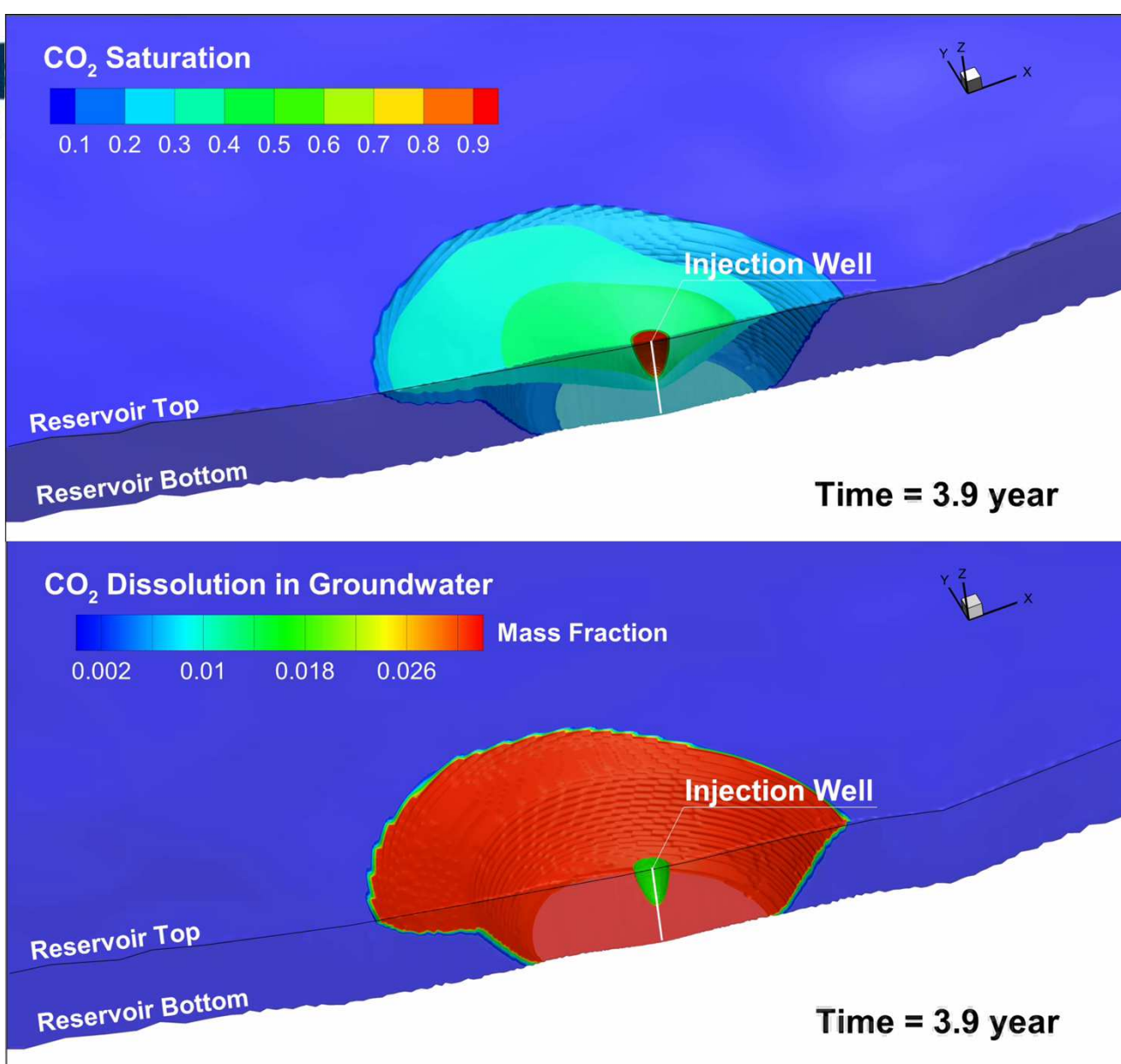


- Buoyant scCO₂ overrides onto groundwater
- Dissolution of CO₂ increases water density
- Denser fluid laid on lighter fluid
- Rayleigh-Taylor instability invokes convective mixing of groundwater

The mixing significantly enhances the CO₂ dissolution into groundwater, resulting in more stable storage

Preliminary 2D simulation (Yamamoto et al., GHGT11) [Dr. Hajime Yamamoto, Taisei]





Density convections for 1,000 years:

Flow Model

Only the far side of the vertical cross section passing through the injection well is depicted.

Reservoir Condition

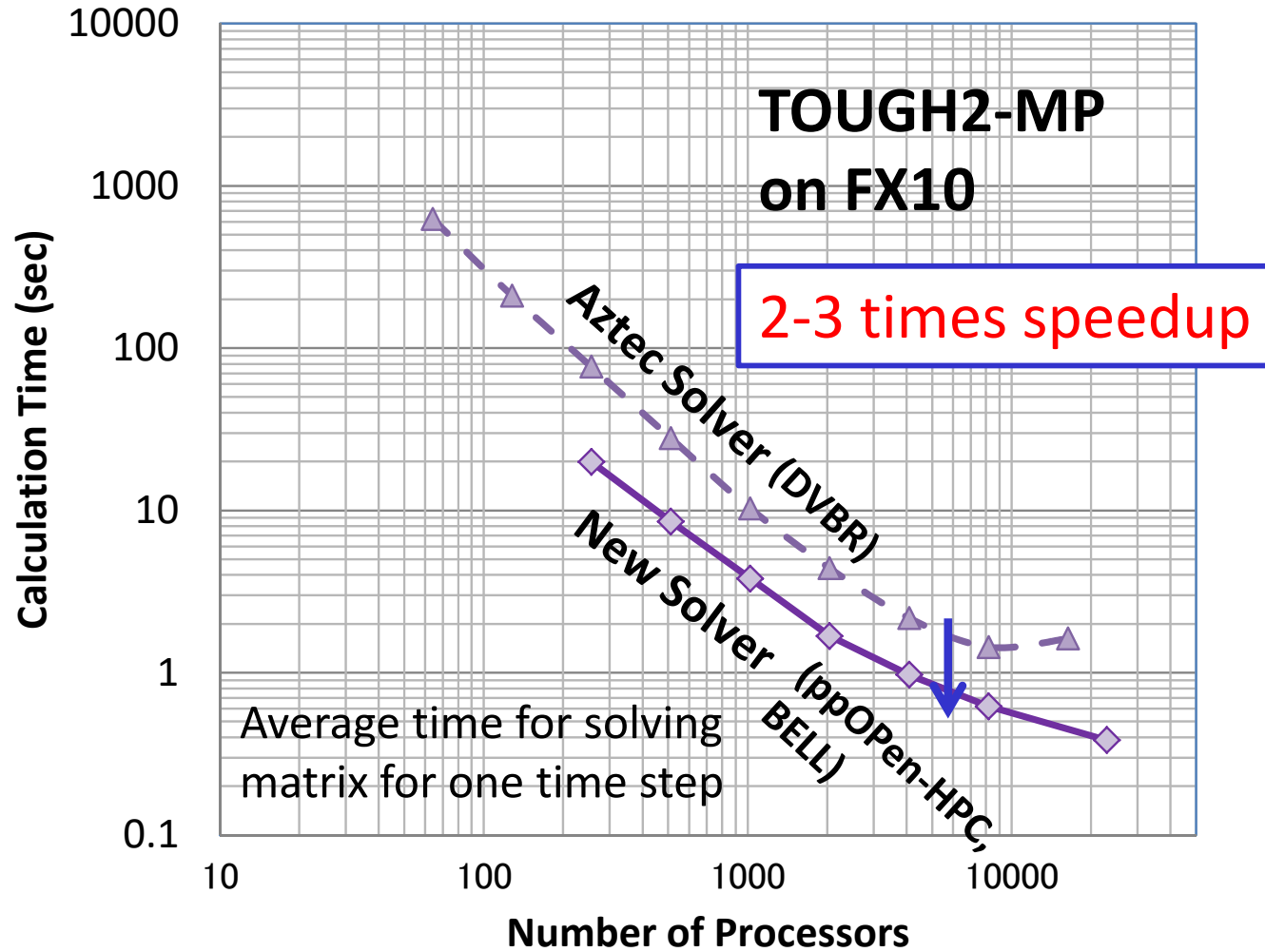
- Permeability: 100 md
- Porosity: 20%
- Pressure: 3MPa
- Temperature: 100°C
- Salinity: 15wt%

[Dr. Hajime Yamamoto, Taisei]

- The meter-scale fingers gradually developed to larger ones in the field-scale model
- Huge number of time steps ($> 10^5$) were required to complete the 1,000-yrs simulation
- Onset time (10-20 yrs) is comparable to theoretical (linear stability analysis, 15.5yrs)

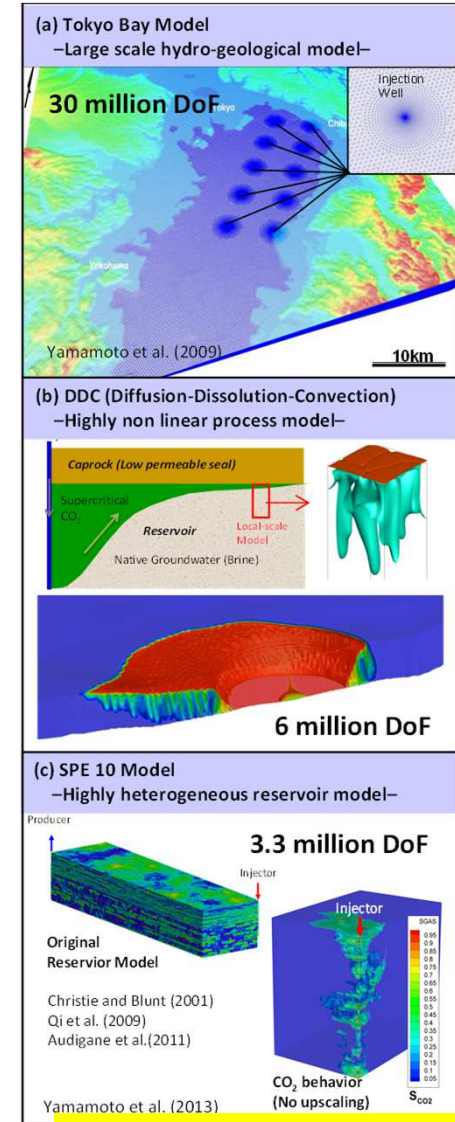
Simulation of Geologic CO₂ Storage

30 million DoF (10 million grids × 3 DoF/grid node)



[Dr. Hajime Yamamoto, Taisei]

Fujitsu FX10 (Oakleaf-FX), 30M DOF: 2x-3x improvement



※ 3D Multiphase Flow (Liquid/Gas) + 3D Mass Transfer

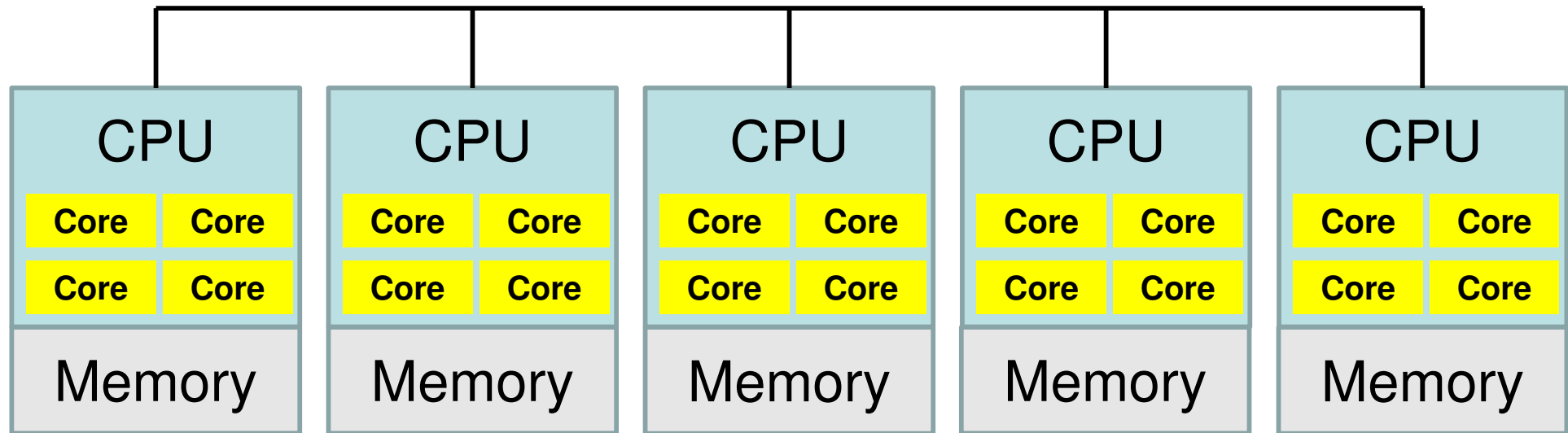
Motivation for Parallel Computing, again

- Large-scale parallel computer enables fast computing in large-scale scientific simulations with detailed models. Computational science develops new frontiers of science and engineering.
- Why parallel computing ?
 - faster
 - larger
 - “larger” is more important from the view point of “new frontiers of science & engineering”, but “faster” is also important.
 - + more complicated
 - Ideal: Scalable
 - Weak Scaling, Strong Scaling

- Target
 - Parallel FEM
 - Introduction to AI/DNN (Deep Neural Network)
- Supercomputers and Computational Science
- **Overview of the Class**
- Future Issues

Our Current Target: Multicore Cluster

Multicore CPU's are connected through network



- OpenMP

- ✓ Multithreading
- ✓ Intra Node (Intra CPU)
- ✓ Shared Memory

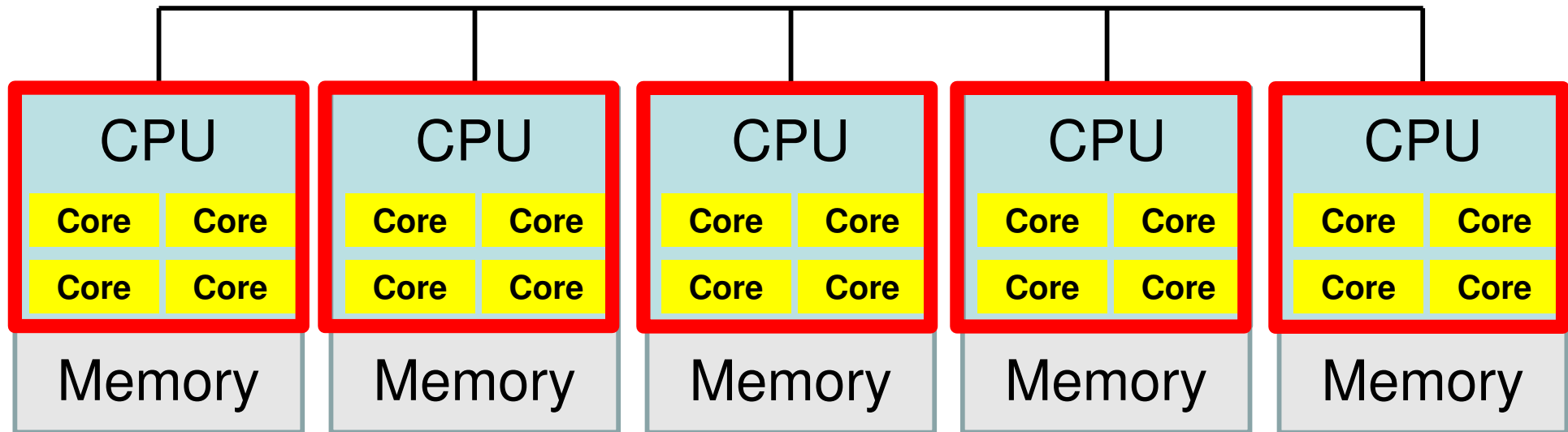
- MPI

- ✓ Message Passing
- ✓ Inter Node (Inter CPU)
- ✓ Distributed Memory



Our Current Target: Multicore Cluster

Multicore CPU's are connected through network



- OpenMP

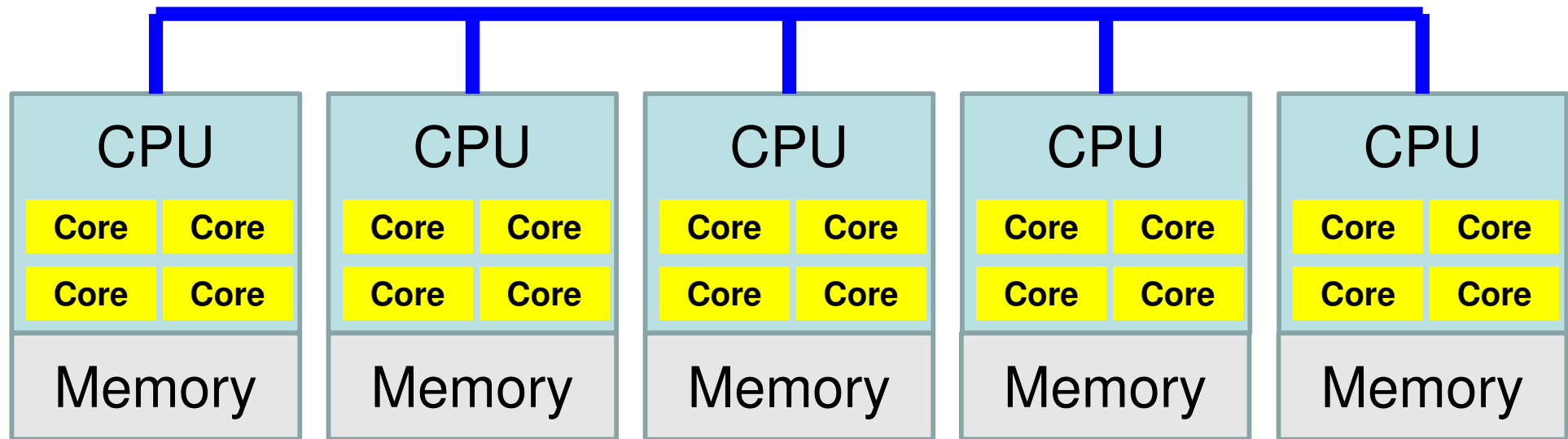
- ✓ Multithreading
- ✓ Intra Node (Intra CPU)
- ✓ Shared Memory

- MPI

- ✓ Message Passing
- ✓ Inter Node (Inter CPU)
- ✓ Distributed Memory

Our Current Target: Multicore Cluster

Multicore CPU's are connected through network

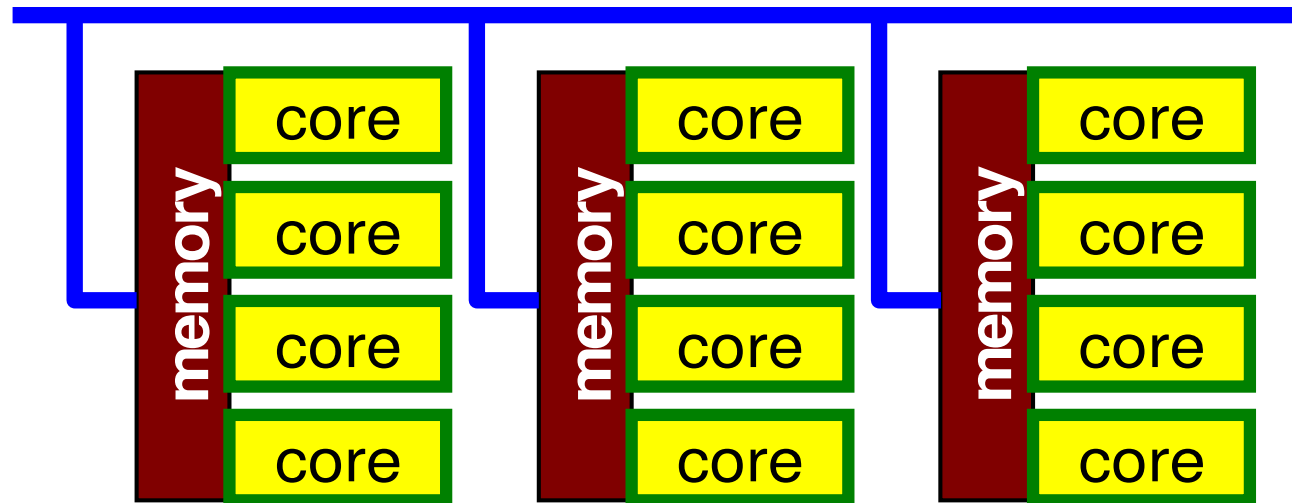


- OpenMP
 - ✓ Multithreading
 - ✓ Intra Node (Intra CPU)
 - ✓ Shared Memory
- MPI (after October)
 - ✓ Message Passing
 - ✓ Inter Node (Inter CPU)
 - ✓ Distributed Memory

Flat MPI vs. Hybrid

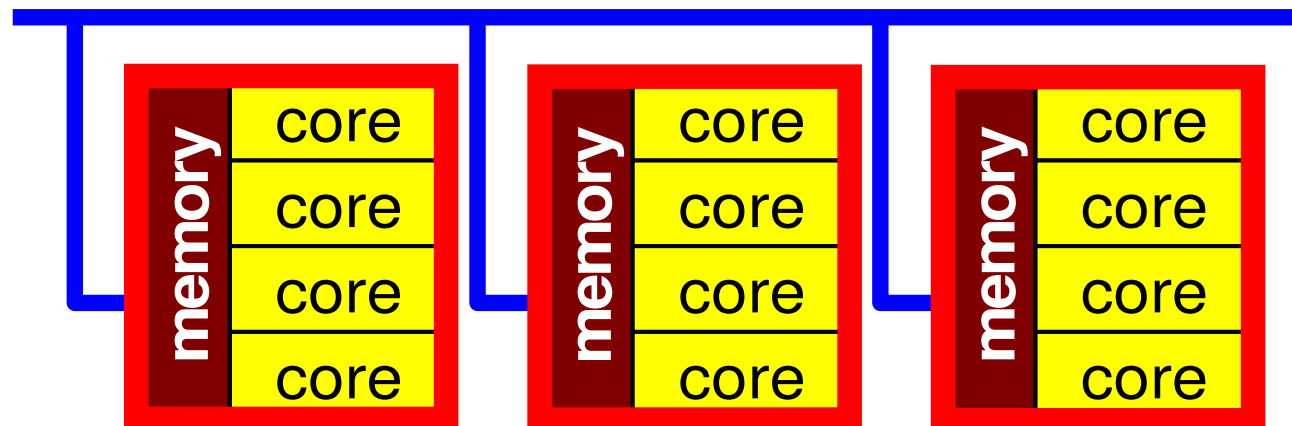
Flat-MPI: Each Core -> Independent

- MPI only
- Intra/Inter Node



Hybrid: Hierarchical Structure

- OpenMP
- MPI



Example of OpenMP/MPI Hybrid

Sending Messages to Neighboring Processes

MPI: Message Passing, OpenMP: Threading with Directives

```
!C
!C- SEND

do neib= 1, NEIBPETOT
  II= (LEVEL-1)*NEIBPETOT
  istart= STACK_EXPORT(II+neib-1)
  inum = STACK_EXPORT(II+neib ) - istart
!$omp parallel do
  do k= istart+1, istart+inum
    WS(k-NE0)= X(NOD_EXPORT(k))
  enddo

  call MPI_Isend (WS(istart+1-NE0), inum, MPI_DOUBLE_PRECISION, &
& NEIBPE(neib), 0, MPI_COMM_WORLD, &
& req1(neib), ierr)
enddo
```

Prerequisites

- Knowledge and experiences in fundamental methods for numerical analysis (e.g. Gaussian elimination, SOR)
- Knowledge and experiences in UNIX/Linux
 - emacs, vi, nano etc.
- Experiences in programming using FORTRAN or C
- Fundamental Issues of FEM
 - <http://nkl.cc.u-tokyo.ac.jp/2022-RIKEN-IHPCSS/02-FEM/FEMintro.pdf>

Information

- Slack Channel
 - <https://rikeninternat-zgz8012.slack.com>
- Class Materials
 - <http://nkl.cc.u-tokyo.ac.jp/2022-RIKEN-IHPCSS/>

September 12 (Mon)

09:00-12:00	Introduction, 1D FEM	Nakajima
13:30-17:00	3D FEM	Nakajima
17:00-18:00	Fugaku Virtual Tour, Overview of Fugaku	Imamura, Terao, Uno

September 13 (Tue)

09:00-12:00	Road to Parallel FEM, Introduction to MPI (1/3)	Nakajima, Terao
13:30-15:30	Exercise (Optional)	Terao, Nakajima
15:30-18:00	Introduction to MPI (2/3)	Nakajima, Terao

September 14 (Wed)

09:00-12:00	Exercise (Optional)	Terao, Nakajima
13:30-15:00	Introduction to MPI (3/3)	Nakajima, Terao
15:00-18:00	Parallel FEM (1/2)	Nakajima, Terao

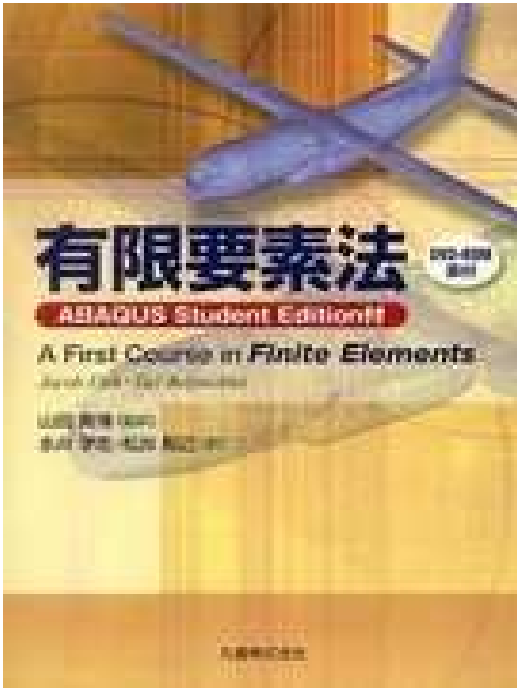
September 15 (Thu)

09:00-12:00	Parallel FEM (2/2)	Nakajima, Terao
13:30-15:00	Parallel Visualization	Nakajima, Terao
15:00-18:00	OpenMP/MPI Hybrid	Nakajima, Terao

September 16 (Fri)

09:00-12:00	PETSc	Terao, Nakajima
13:30-17:30	CNN	Imaura, Terao
17:30-	Closing	All

References



- Fish, Belytschko, A First Course in Finite Elements, Wiley, 2007
 - Japanese version is also available
 - “ABAQUS Student Edition” included
- Smith et al., Programming the Finite Element Method (4th edition), Wiley, 2004
 - Parallel FEM included
- Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover, 2000

- Target: Parallel FEM
- Supercomputers and Computational Science
- Overview of the Class
- **Future Issues**

Technical Issues: Future of Supercomputers

- Power Consumption
- Reliability, Fault Tolerance, Fault Resilience
- Scalability (Parallel Performance)

Key-Issues towards Appl./Algorithms on Exa-Scale Systems

Jack Dongarra (ORNL/U. Tennessee) at ISC 2013

- Hybrid/Heterogeneous Architecture
 - Multicore + GPU/Manycores (Intel MIC/Xeon Phi)
 - Data Movement, Hierarchy of Memory
- Communication/Synchronization Reducing Algorithms
- Mixed Precision Computation
- Auto-Tuning/Self-Adapting
- Fault Resilient Algorithms
- Reproducibility of Results

Supercomputers with Heterogeneous/Hybrid Nodes

