Introduction to Parallel Programming for Multicore/Manycore Clusters

Introduction

Kengo Nakajima
Information Technology Center
The University of Tokyo

Descriptions of Class

- Technical & Scientific Computing I (4820-1027)
  - 科学技術計算Ｉ
  - Department of Mathematical Informatics
- Seminar on Computer Science I (4810-1204)
  - コンピュータ科学特別講義Ｉ
  - Department of Computer Science

Changes in 2015

- 2009-2014
  - Introduction to FEM Programming
    - FEM: Finite-Element Method：有限要素法
  - Summer (I) : FEM Programming for Solid Mechanics
  - Winter (II): Parallel FEM using MPI
    - The 1st part (summer) is essential for the 2nd part (winter)
- Problems
  - Many new (international) students in Winter, who did not take the 1st part in Summer
  - **They are generally more diligent than Japanese students**
- 2015 (Information in the printed handbook is wrong)
  - Summer (I) : Multicore programming using OpenMP
  - Winter (II): FEM + Parallel FEM using MPI for Heat Conduction
  - Part I & II are independent (maybe...)

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
    - Solving N^x scale problem using N^x computational resources during same computation time (weak scaling)

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): Computer Science, Computational Science, or Numerical Algorithms ?

This Class ...

- **Science**
- **Modeling**
- **Algorithm**
- **Software**
- **Hardware**

- **Target: Parallel FVM (Finite-Volume Method) using OpenMP**
- Science: 3D Poisson Equations
- Modeling: FVM
- Algorithm: Iterative Solvers etc.

- You have to know many components to learn FVM, 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)

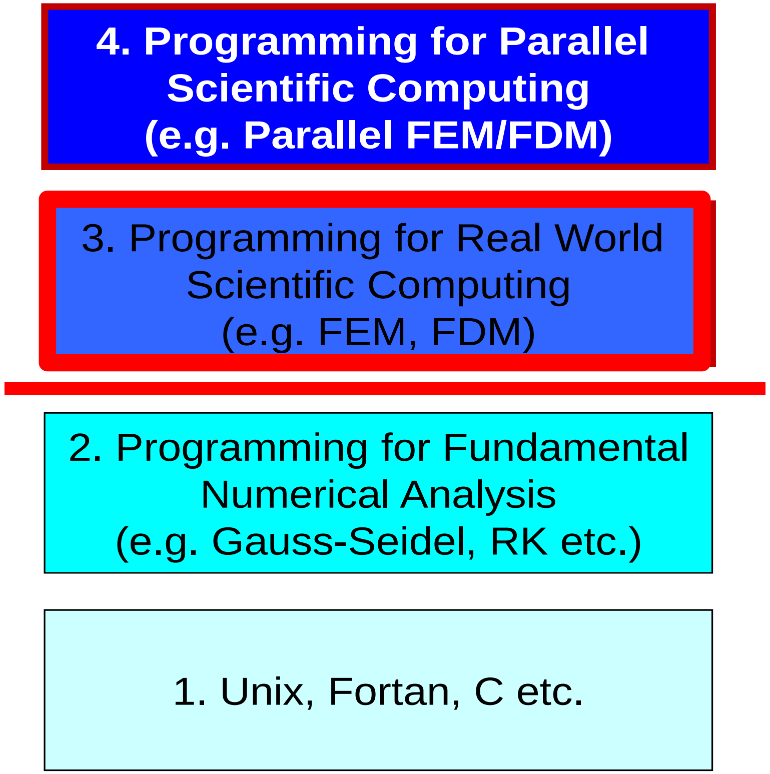
<u>Big gap here !!</u>

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

Unix, Fortan, C etc.

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.
  - 3: FVM, 4: Parallel FVM

Kengo Nakajima 中島研吾 (1/2)

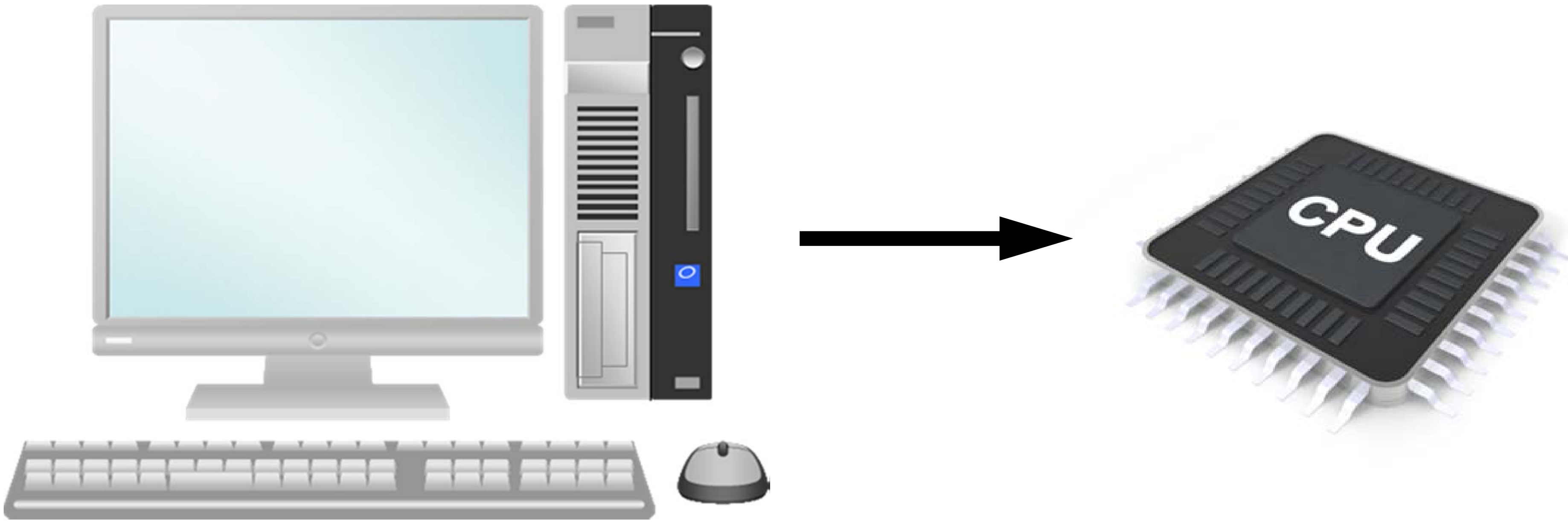
- Current Position
  - Professor, Supercomputing Research Division, Information Technology Center, The University of Tokyo（情報基盤センター）
    - Department of Mathematical Informatics, Graduate School of Information Science & Engineering, The University of Tokyo（情報理工・数理情報学）
    - Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo（工・電気系工学）
  - Visiting Senior Researcher, Advanced Institute for Computational Science (AICS), RIKEN
- Research Interest
  - High-Performance Computing
  - Parallel Numerical Linear Algebra (Preconditioning)
  - Parallel Programming Model
  - Computational Mechanics, Computational Fluid Dynamics
  - Adaptive Mesh Refinement, Parallel Visualization

Kengo Nakajima (2/2)

- Education
  - B.Eng (Aeronautics, The University of Tokyo, 1985)
  - M.S. (Aerospace Engineering, University of Texas, 1993)
  - Ph.D. (Quantum Engineering & System Sciences, The University of Tokyo, 2003)
- Professional
  - Mitsubishi Research Institute, Inc. (1985-1999)
  - Research Organization for Information Science & Technology (1999-2004)
  - The University of Tokyo
    - Department Earth & Planetary Science (2004-2008)
    - Information Technology Center (2008-)
  - JAMSTEC (2008-2011), part-time
  - RIKEN (2009-), part-time

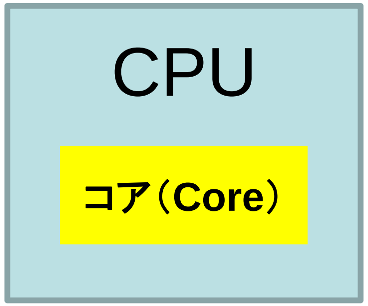
- **Supercomputers and Computational Science**
- Overview of the Class
- Future Issues

Computer & CPU

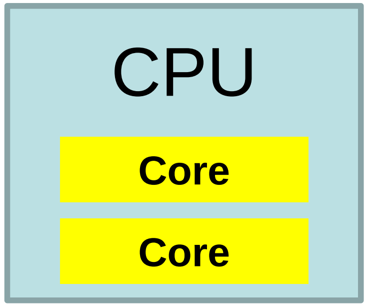


- Central Processing Unit（中央処理装置）:CPU
- CPU’s used in PC and Supercomputers are based on same architecture
- GHz: Clock Rate
  - Frequency: Number of operations by CPU per second
    - GHz -> 10^9 operations/sec
  - Simultaneous 4-8 instructions per clock

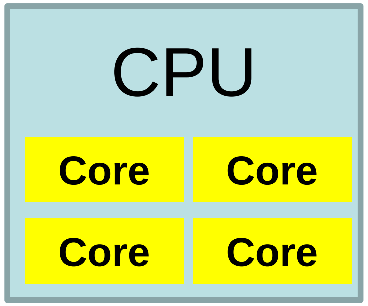
Multicore CPU



Single Core
1 cores/CPU

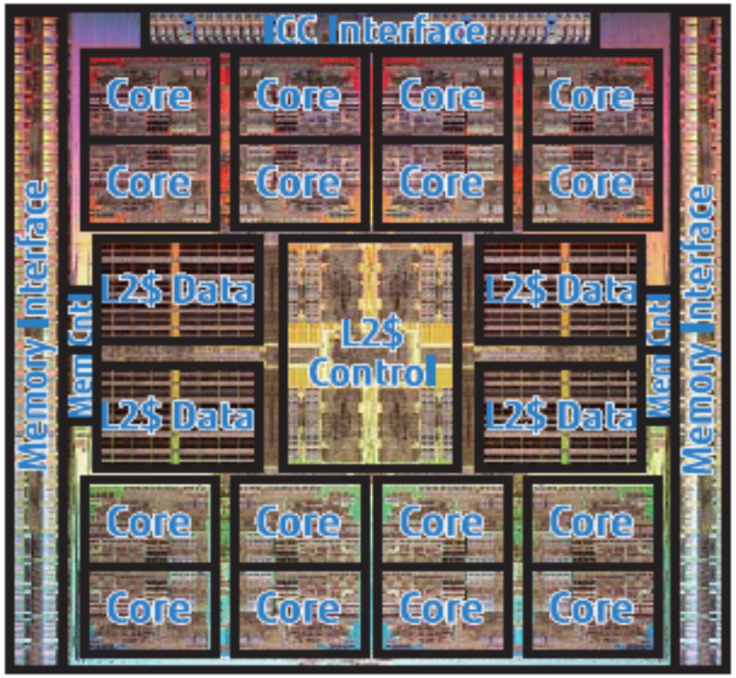


Dual Core
2 cores/CPU



Quad Core
4 cores/CPU

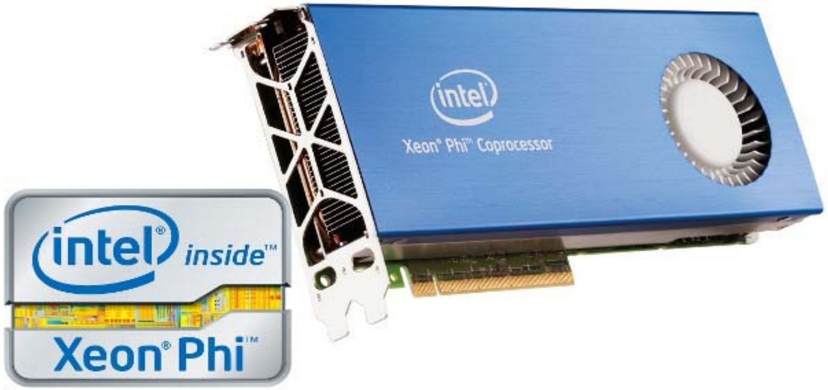
- Core= Central part of CPU
- Multicore CPU's with 4-8 cores are popular
  - Low Power



- GPU: Manycore
  - $O(10^1)$-$O(10^2)$ cores
- More and more cores
  - Parallel computing
- Oakleaf-FX at University of Tokyo: 16 cores
  - SPARC64™ IXfx

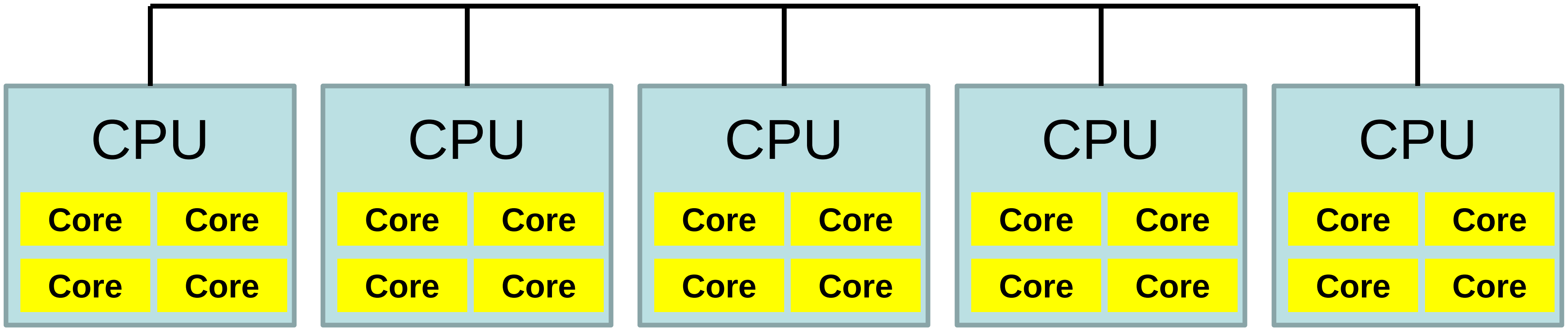
GPU/Manycores

- GPU : Graphic Processing Unit
  - GPGPU: General Purpose GPU
  - $O(10^2)$ cores
  - High Memory Bandwidth
  - Cheap
  - NO stand-alone operations
    - Host CPU needed
  - Programming: CUDA, OpenACC
- Intel Xeon/Phi: Manycore CPU
  - 60 cores
  - High Memory Bandwidth
  - Unix, Fortran, C compiler
  - Currently, host CPU needed
    - Stand-alone will be possible soon

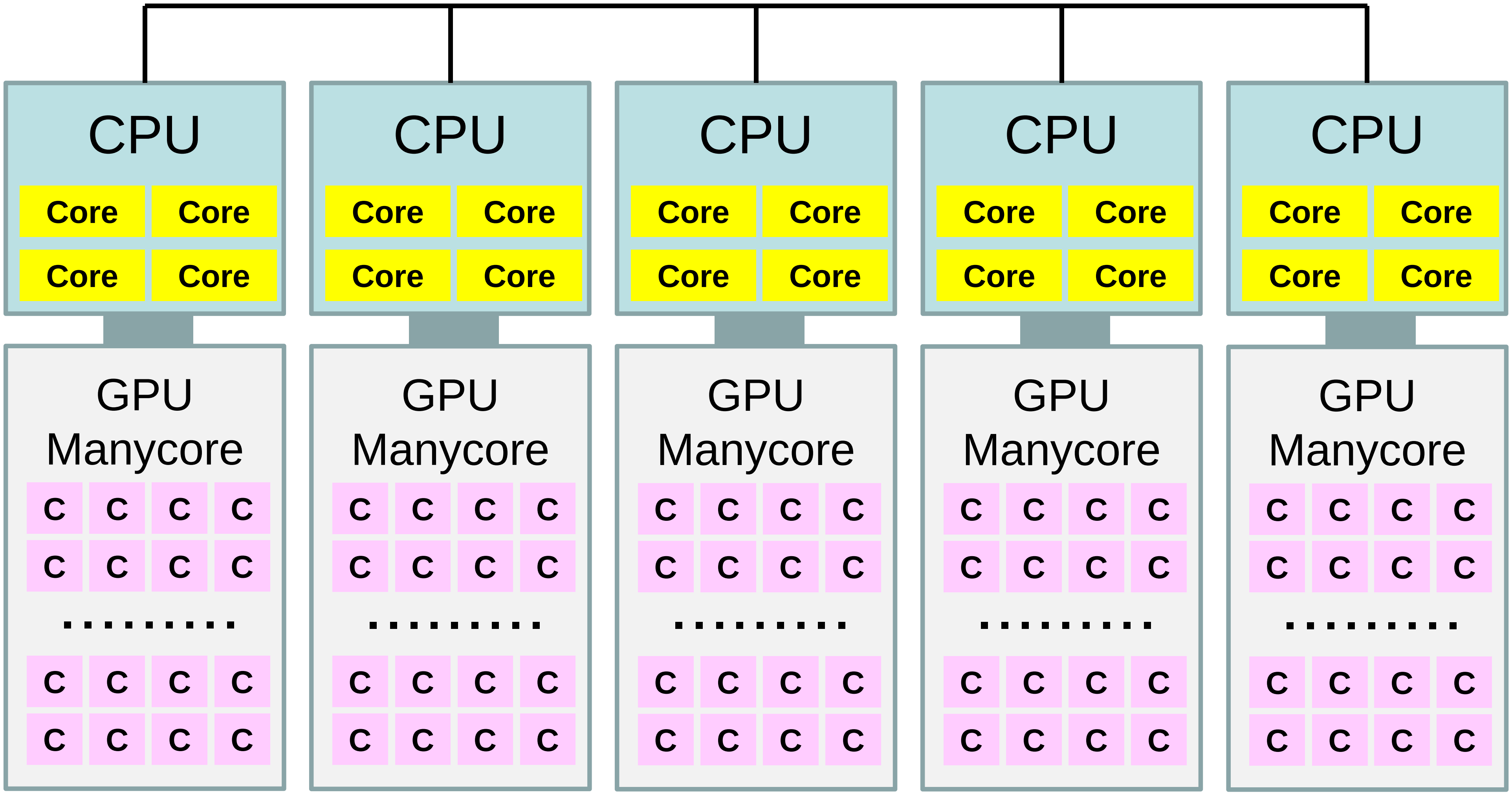


Parallel Supercomputers

Multicore CPU’s are connected through network



Supercomputers with Heterogeneous/Hybrid Nodes



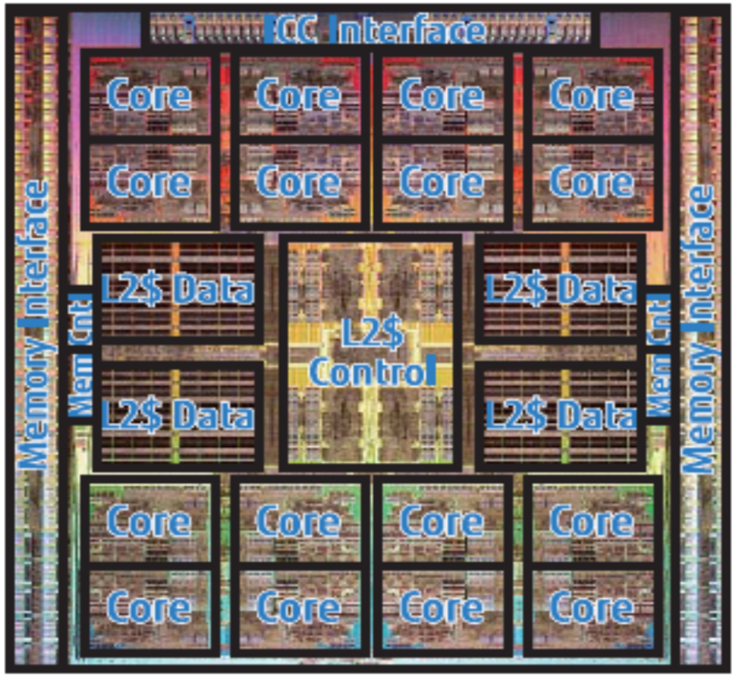
Performance of Supercomputers

- 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
    - $3 \times 10^9 \times 4$(or 8)=12(or 24) $\times 10^9$ FLOPS=12(or 24)GFLOPS

    - 10^6 FLOPS= 1 Mega FLOPS = 1 MFLOPS
    - 10^9 FLOPS= 1 Giga FLOPS = 1 GFLOPS
    - 10^{12} FLOPS= 1 Tera FLOPS = 1 TFLOPS
    - 10^{15} FLOPS= 1 Peta FLOPS = 1 PFLOPS
    - 10^{18} FLOPS= 1 Exa FLOPS = 1 EFLOPS

Peak Performance of Oakleaf-FX

Fujitsu PRIMEHPC FX10 at U.Tokyo

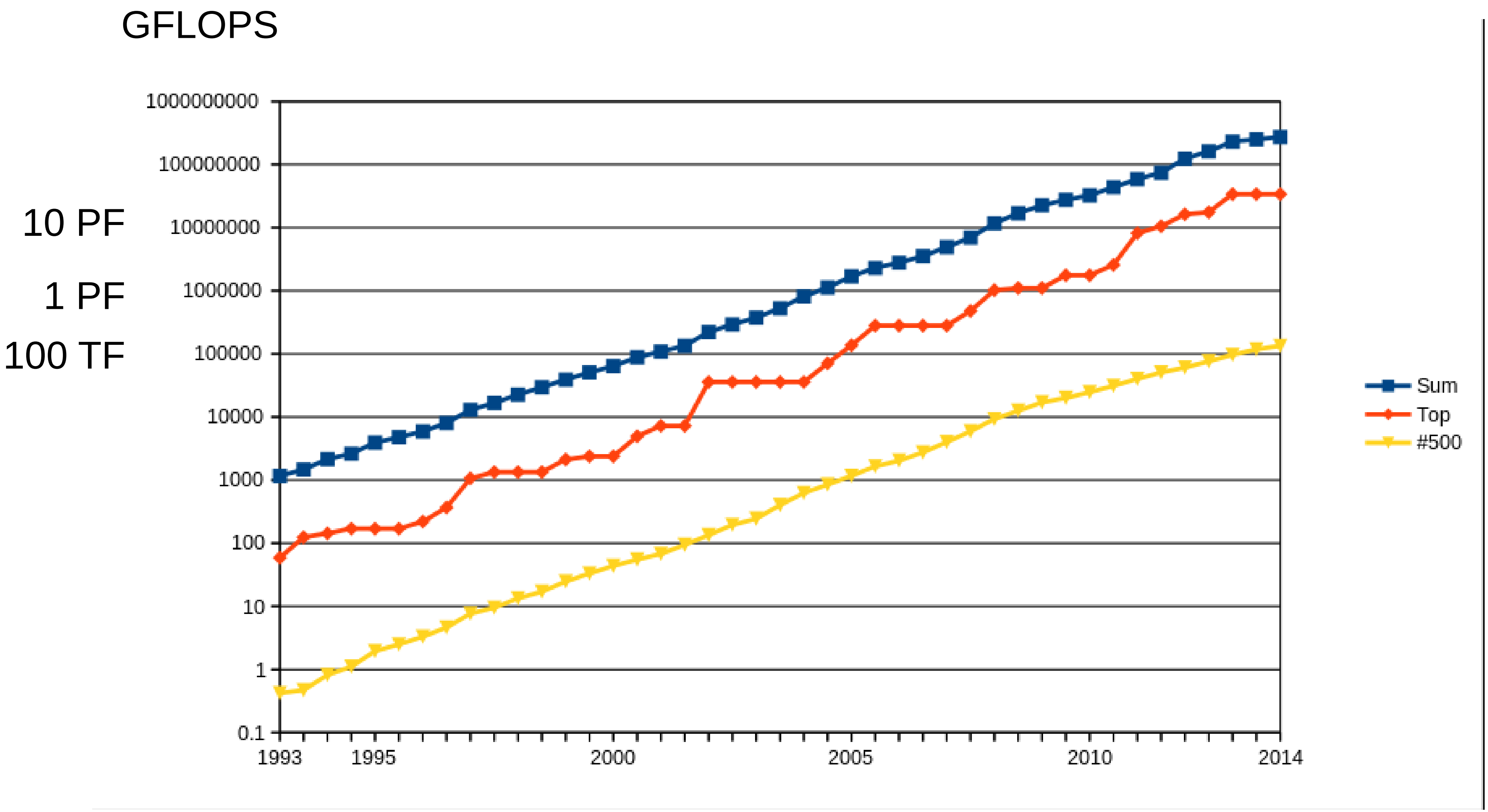


- 1.848 GHz
- 8 FLOP operations per Clock
- Peak Performance (1 core)
  - 1.848 × 8= 14.78 GFLOPS
- Peak Performance (1 node/16 cores)
  - 236.5 GFLOPS
- Peak Performance of Entire Performance
  - 4,800 nodes, 76,800 cores
  - 1.13 PFLOPS

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)
- Linpack
  - iPhone version is available



- PFLOPS: Peta (=10^{15}) Floating OPerations per Sec.
- Exa-FLOPS (=10^{18}) will be attained in 2020

http://www.top500.org/

44^{th} TOP500 List (November, 2014)

	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Tianjin, China	**Tianhe-2** Intel Xeon E5-2692, TH Express-2, IXeon Phi2013 NUDT	3120000	33863 (= 33.9 PF)	54902	17808
2	Oak Ridge National Laboratory, USA	**Titan** Cray XK7/NVIDIA K20x, 2012 Cray	560640	17590	27113	8209
3	Lawrence Livermore National Laboratory, USA	**Sequoia** BlueGene/Q, 2011 IBM	1572864	17173	20133	7890
4	**RIKEN AICS, Japan**	**K computer, SPARC64 VIIIfx , 2011 Fujitsu**	**705024**	**10510**	**11280**	**12660**
5	Argonne National Laboratory, USA	**Mira** BlueGene/Q, 2012 IBM	786432	8587	10066	3945
6	Swiss Natl. Supercomputer Center, Switzerland	**Piz Daint** Cray XC30/NVIDIA K20x, 2013, Cray	115984	6271	7789	2325
7	TACC, USA	**Stampede** Xeon E5-2680/Xeon Phi, 2012 Dell	462462	5168	8520	4510
8	Forschungszentrum Juelich (FZJ), Germany	**JuQUEEN** BlueGene/Q, 2012 IBM	458752	5009	5872	2301
9	DOE/NNSA/LLNL, USA	**Vulcan** BlueGene/Q, 2012 IBM	393216	4293	5033	1972
10	Government, USA	Cray CS-Storm/Xeon E5-2670/2680/NVIDIA K40, 2014 Cray	72800	3577	6132	1499

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

http://www.top500.org/

44^{th} TOP500 List (November, 2014)

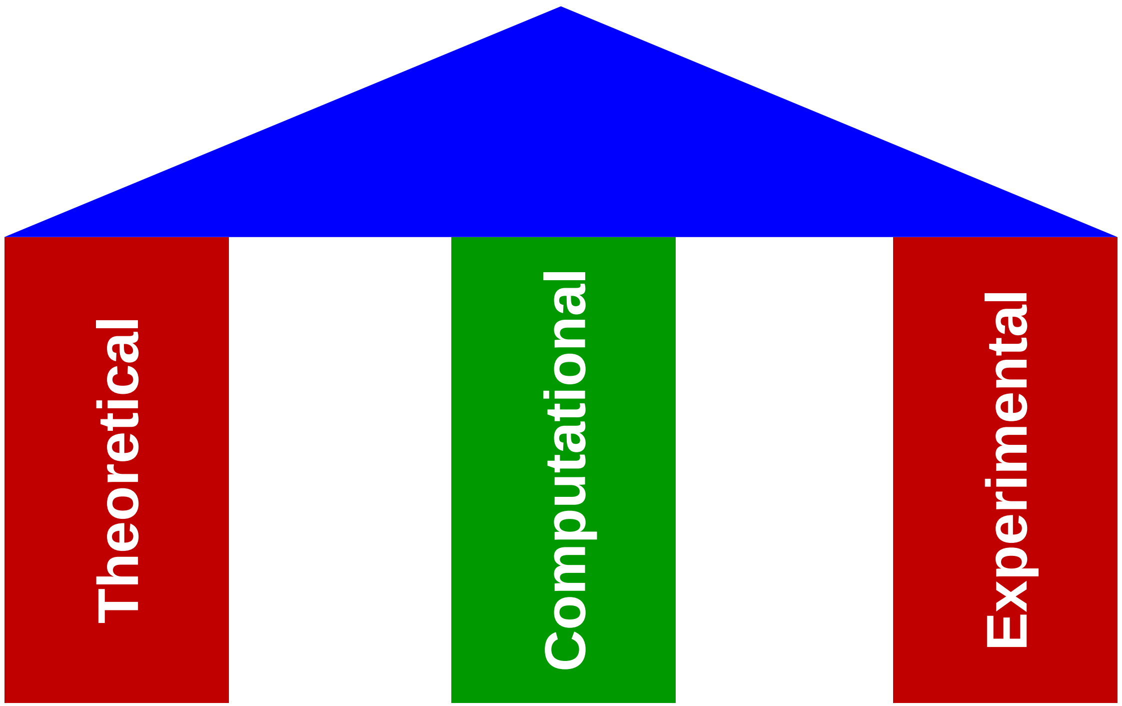
	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	National Supercomputing Center in Tianjin, China	**Tianhe-2** Intel Xeon E5-2692, TH Express-2, IXeon Phi2013 NUDT	3120000	33863 (= 33.9 PF)	54902	17808
2	Oak Ridge National Laboratory, USA	**Titan** Cray XK7/NVIDIA K20x, 2012 Cray	560640	17590	27113	8209
3	Lawrence Livermore National Laboratory, USA	**Sequoia** BlueGene/Q, 2011 IBM	1572864	17173	20133	7890
4	**RIKEN AICS, Japan**	**K computer, SPARC64 VIIIfx , 2011 Fujitsu**	**705024**	**10510**	**11280**	**12660**
5	Argonne National Laboratory, USA	**Mira** BlueGene/Q, 2012 IBM	786432	8587	10066	3945
6	Swiss Natl. Supercomputer Center, Switzerland	**Piz Daint** Cray XC30/NVIDIA K20x, 2013, Cray	115984	6271	7789	2325
7	TACC, USA	**Stampede** Xeon E5-2680/Xeon Phi, 2012 Dell	462462	5168	8520	4510
8	Forschungszentrum Juelich (FZJ), Germany	**JuQUEEN** BlueGene/Q, 2012 IBM	458752	5009	5872	2301
9	DOE/NNSA/LLNL, USA	**Vulcan** BlueGene/Q, 2012 IBM	393216	4293	5033	1972
10	Government, USA	Cray CS-Storm/Xeon E5-2670/2680/NVIDIA K40, 2014 Cray	72800	3577	6132	1499
48	**ITC/U. Tokyo Japan**	**Oakleaf-FX SPARC64 IXfx, 2012 Fujitsu**	**76800**	**1043**	**1135**	**1177**

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

http://www.top500.org/

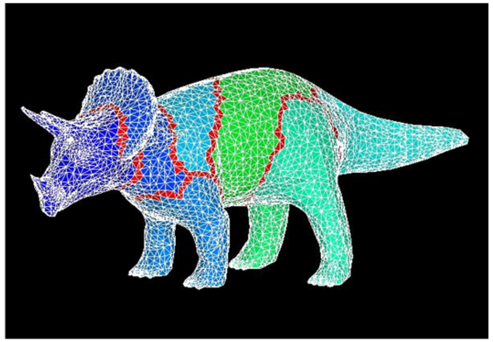
Computational Science
The 3rd Pillar of Science

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

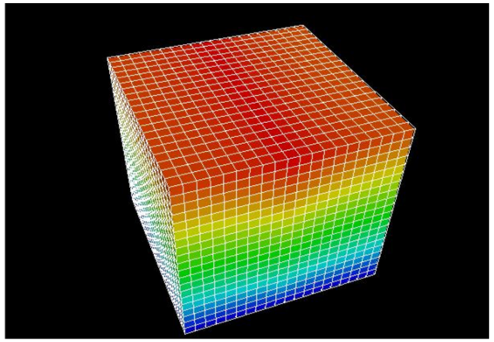


Methods for Scientific Computing

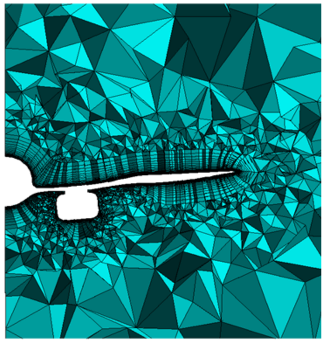
- Numerical solutions of PDE (Partial Diff. Equations)
- Grids, Meshes, Particles
  - Large-Scale Linear Equations
  - Finer meshes provide more accurate solutions



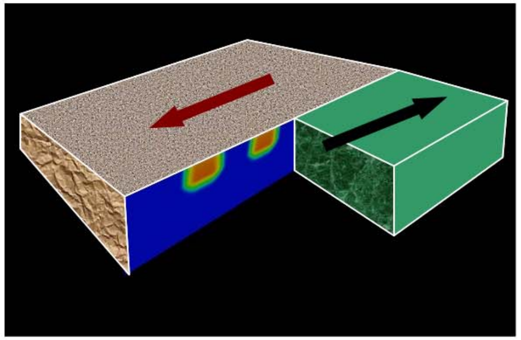
有限要素法
Finite Element Method
FEM



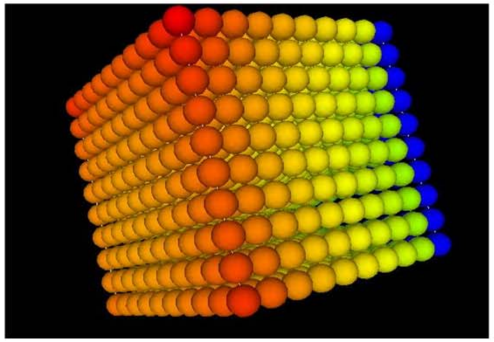
差分法
Finite Difference Method
FDM



有限体積法
Finite Volume Method
FVM



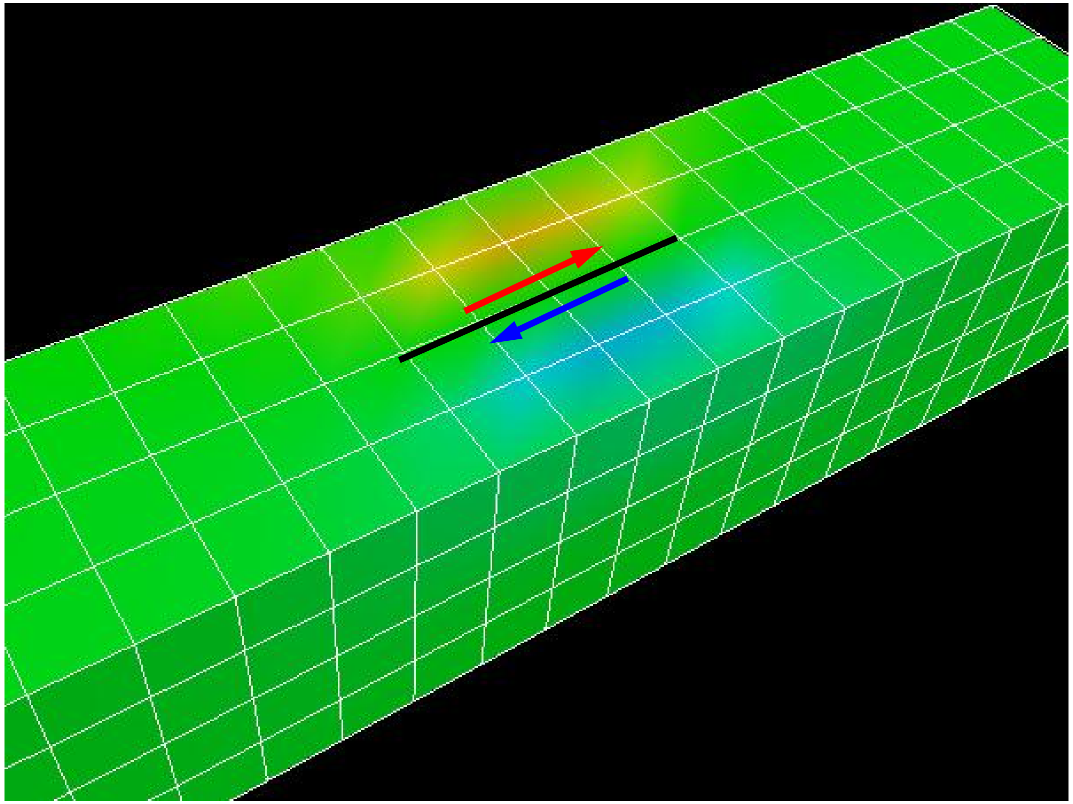
境界要素法
Boundary Element Method
BEM

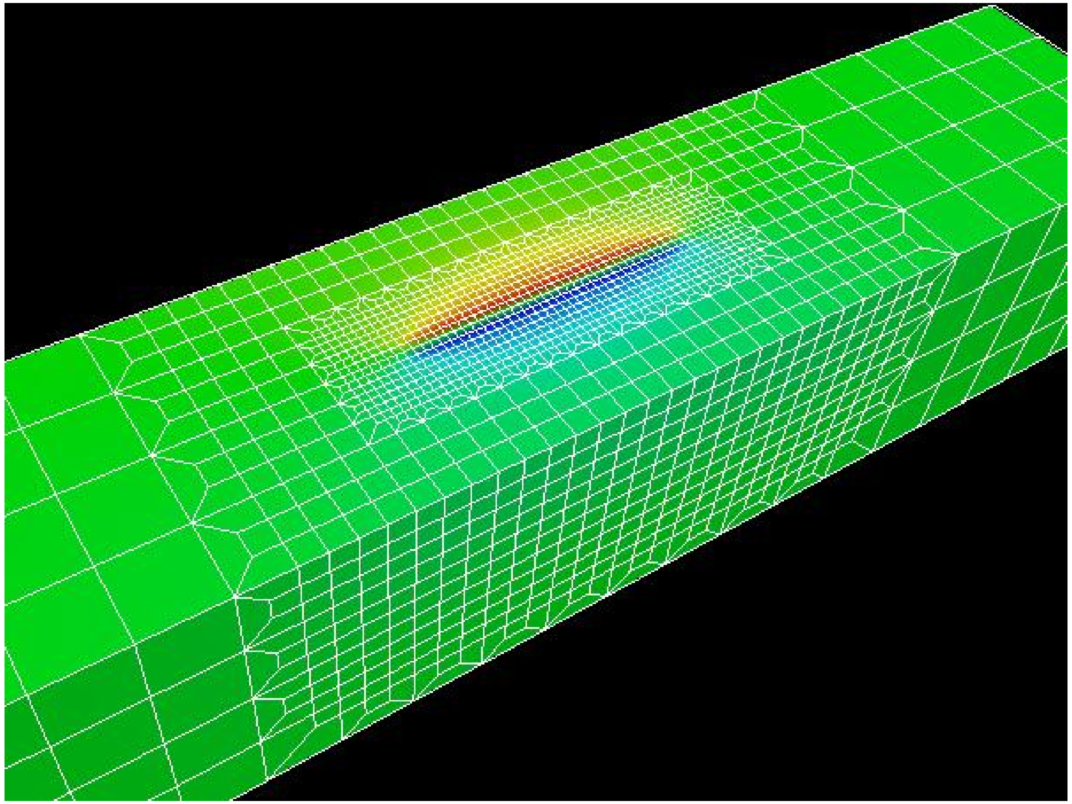


個別要素法
Discrete Element Method
DEM

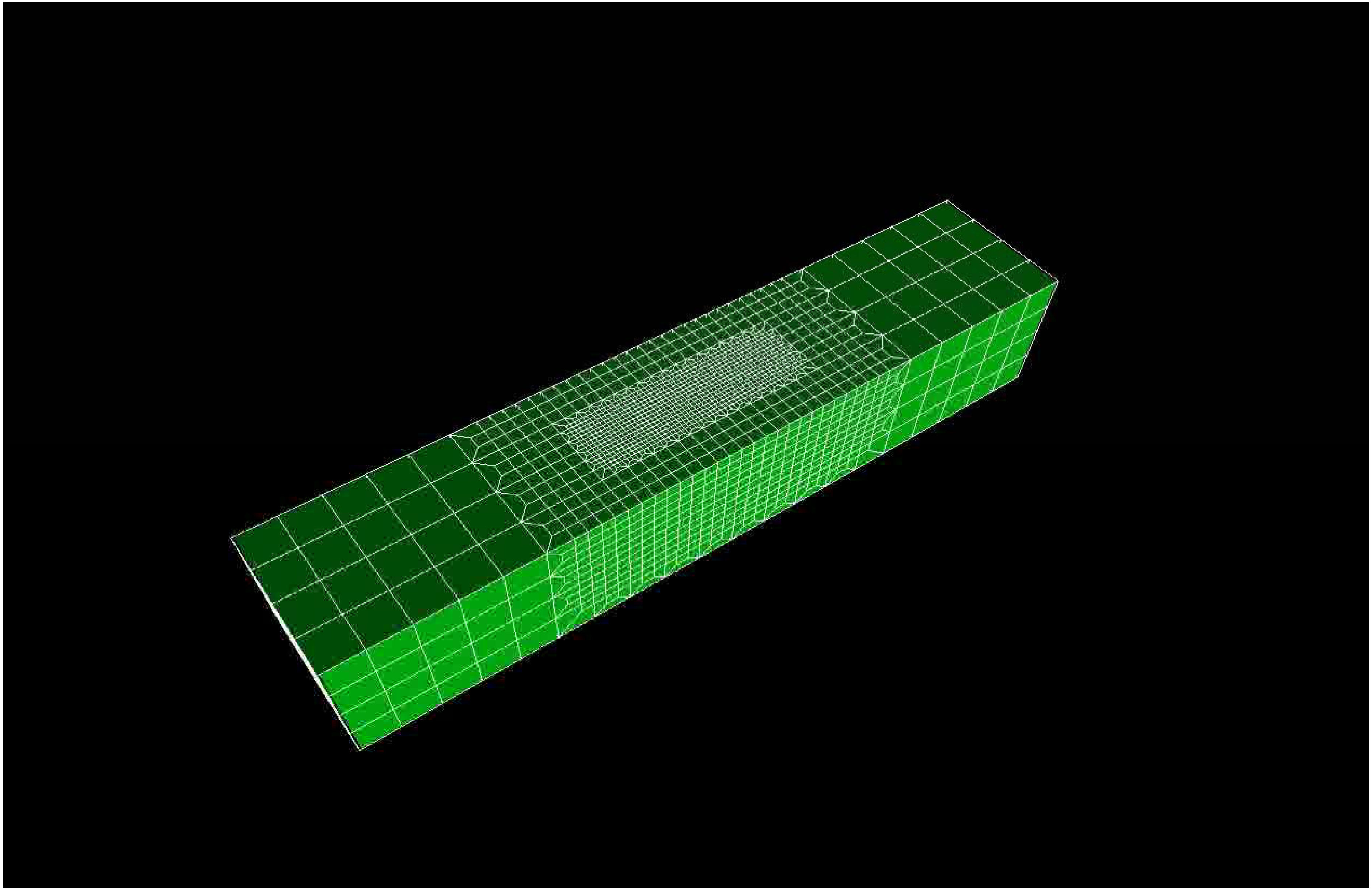
3D Simulations for Earthquake Generation Cycle San Andreas Faults, CA, USA

Stress Accumulation at Transcurrent Plate Boundaries

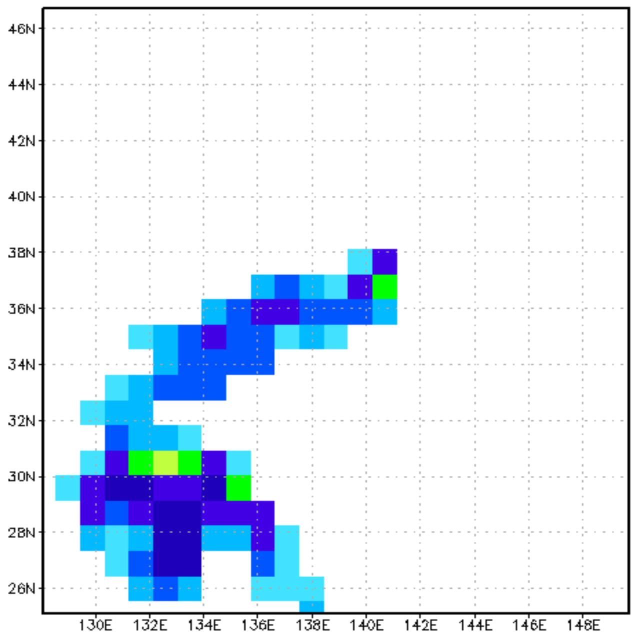




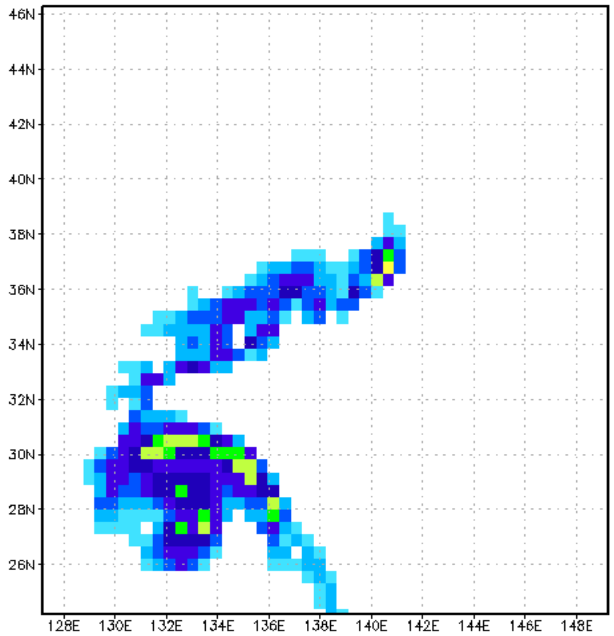
Adaptive FEM: High-resolution needed at meshes with large deformation (large accumulation)



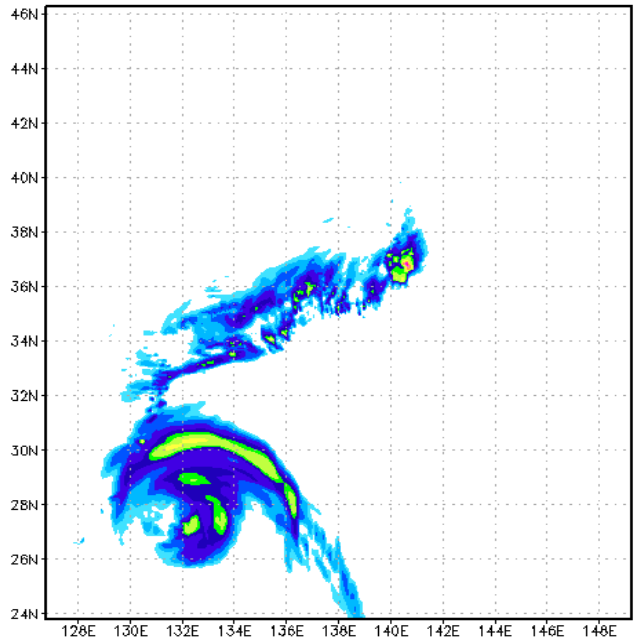
Typhoon Simulations by FDM
Effect of Resolution



Δh=100km



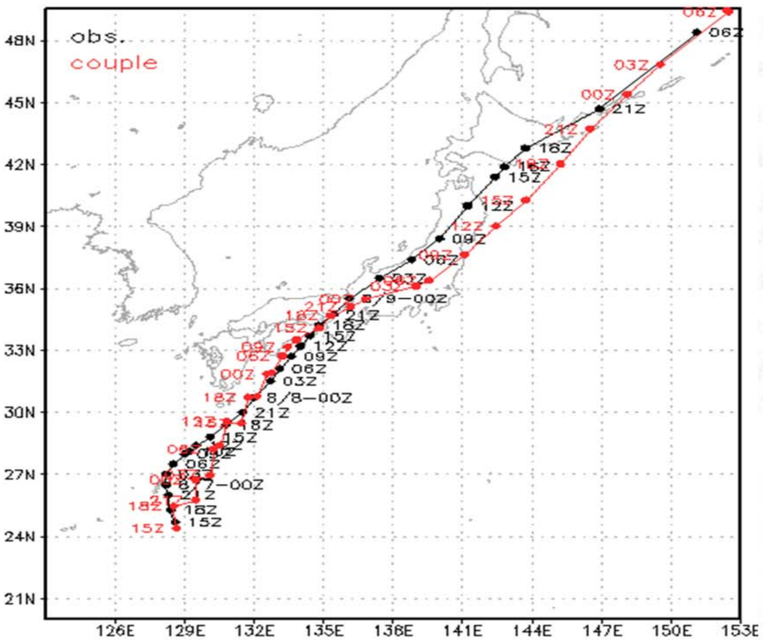
Δh=50km

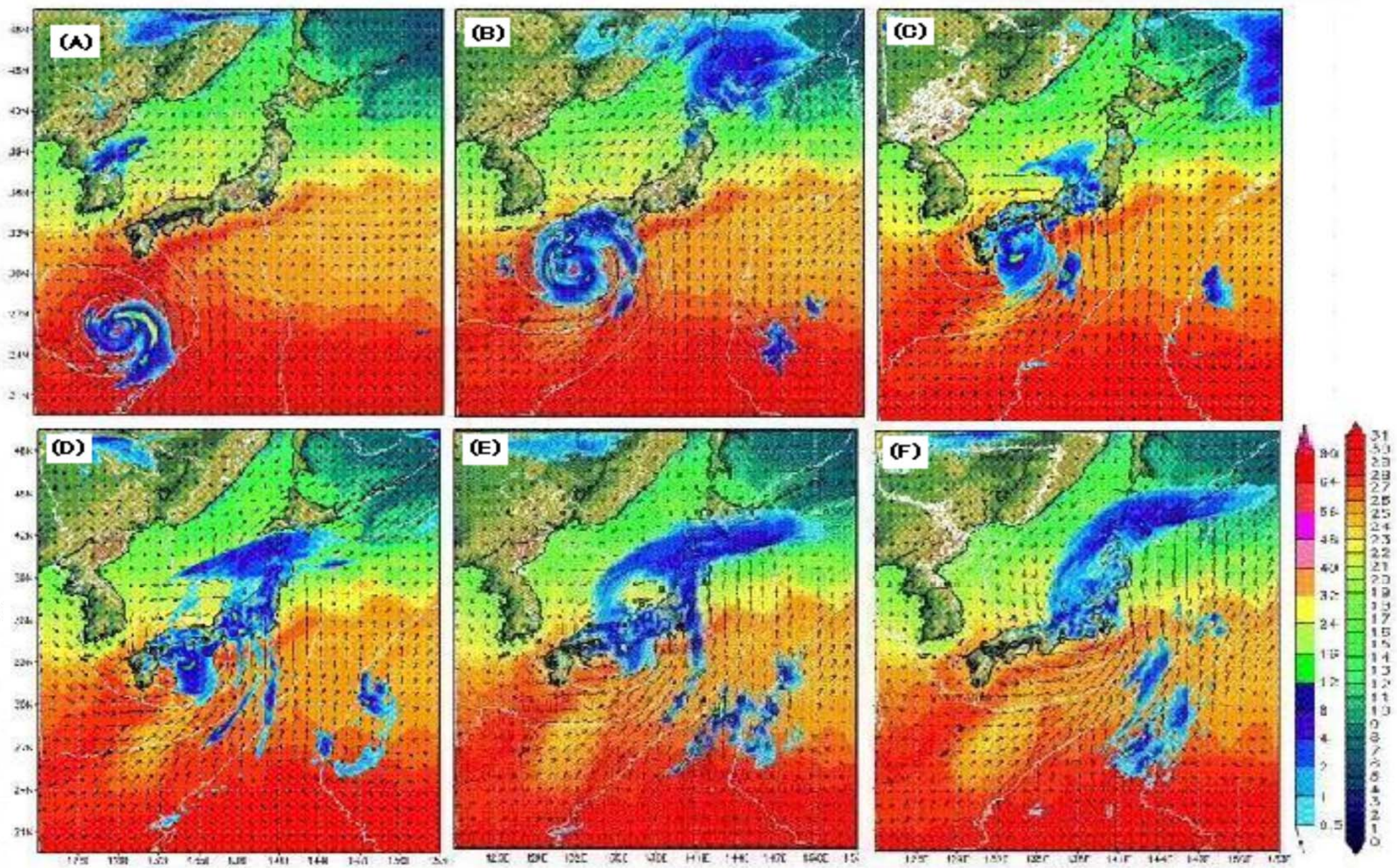


Δh=5km

[JAMSTEC]

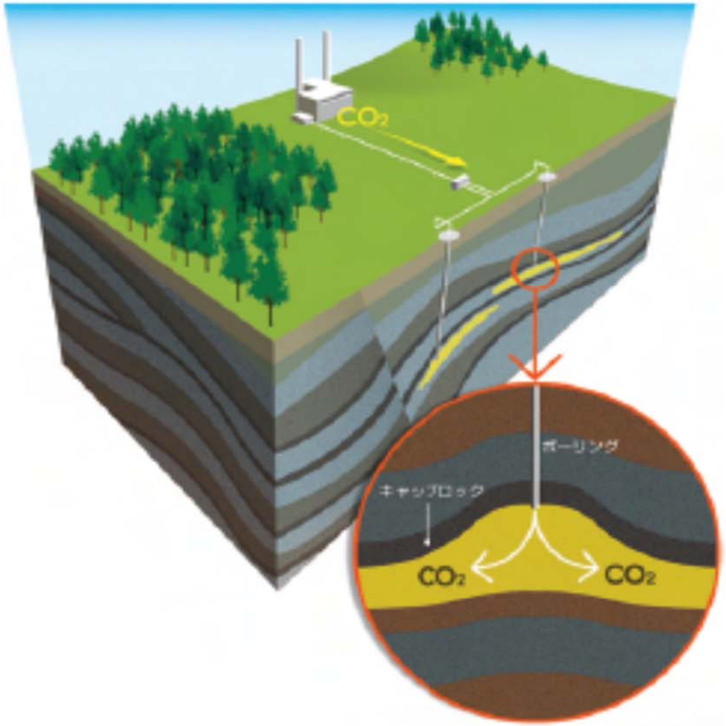
Simulation of Typhoon MANGKHUT in 2003 using the Earth Simulator





[JAMSTEC]

Simulation of Geologic CO_2 Storage



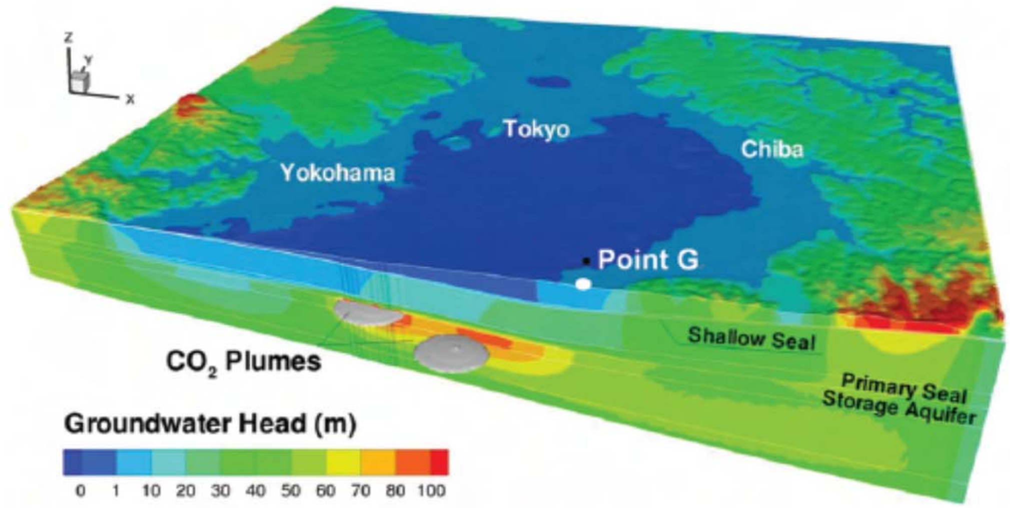
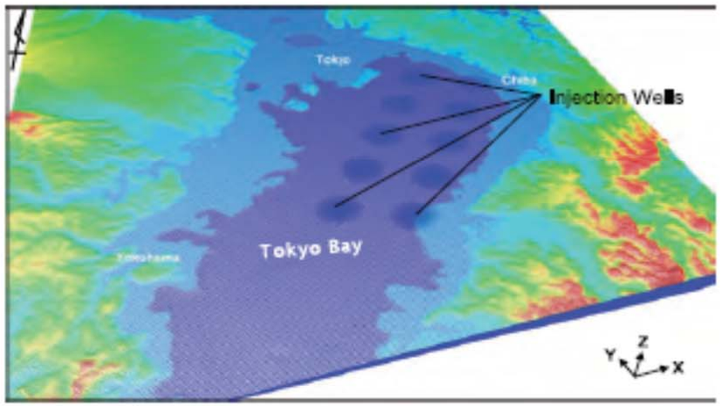
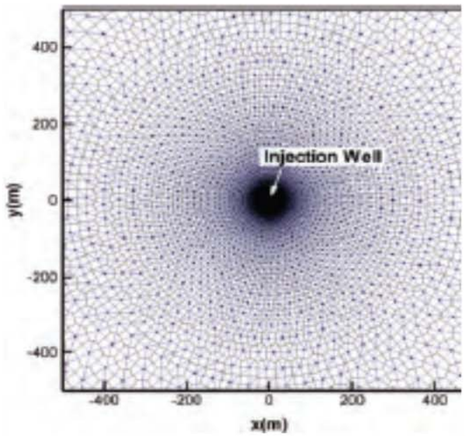
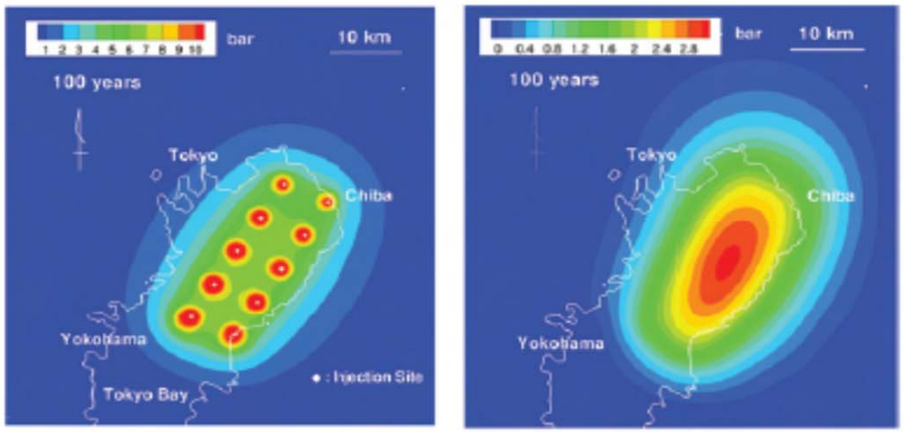


図 -4　CO_2 圧入後の地下水圧（全水頭換算）の分布（100 年後）







(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 CO_2 Storage

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

Science

Modeling

Algorithm

Software

Hardware

Simulation of Geologic CO_2 Storage

- Science
  - Behavior of CO_2 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 10^7 unknowns
- Numerical Algorithm
  - Fast algorithm for large-scale linear equations developed by Information Technology Center, the University of Tokyo
- Supercomputer
  - Earth Simulator (Peak Performance: 130 TFLOPS)
    - NEC, JAMSEC

Concentration of CO_2 in Groundwater

Meshes with higher resolution provide more accurate prediction ⇒ Larger Model/Linear Equations

[Dr. Hajime Yamamoto, Taisei]

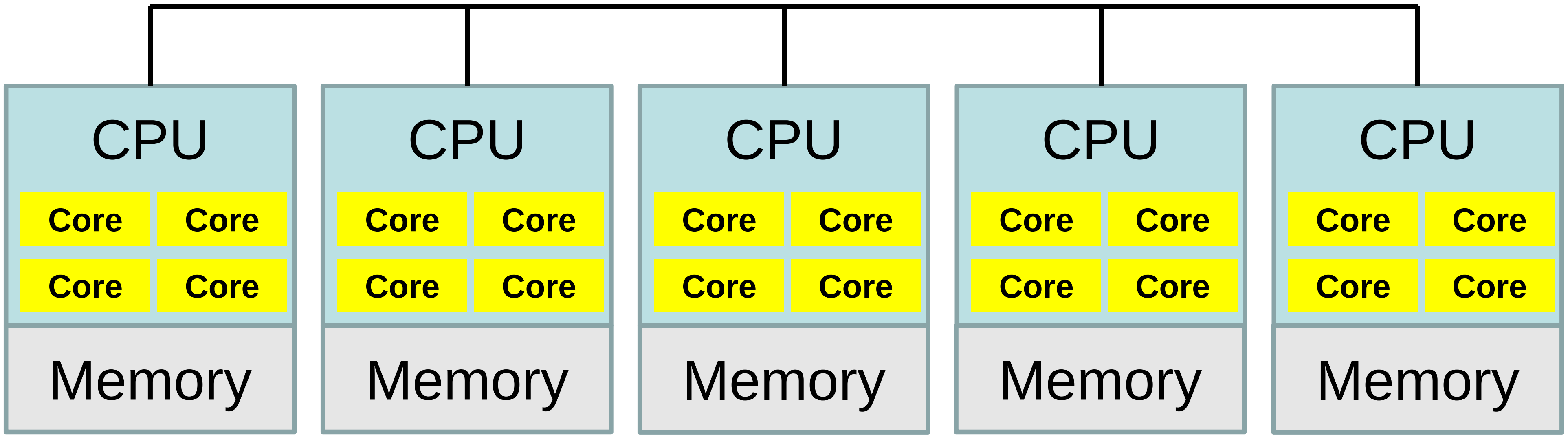
Motivation for Parallel Computing, again

- 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
    - Solving N^x scale problem using N^x computational resources during same computation time.

- Supercomputers and Computational Science
- **Overview of the Class**
- Future Issues

Our Current Target: Multicore Cluster

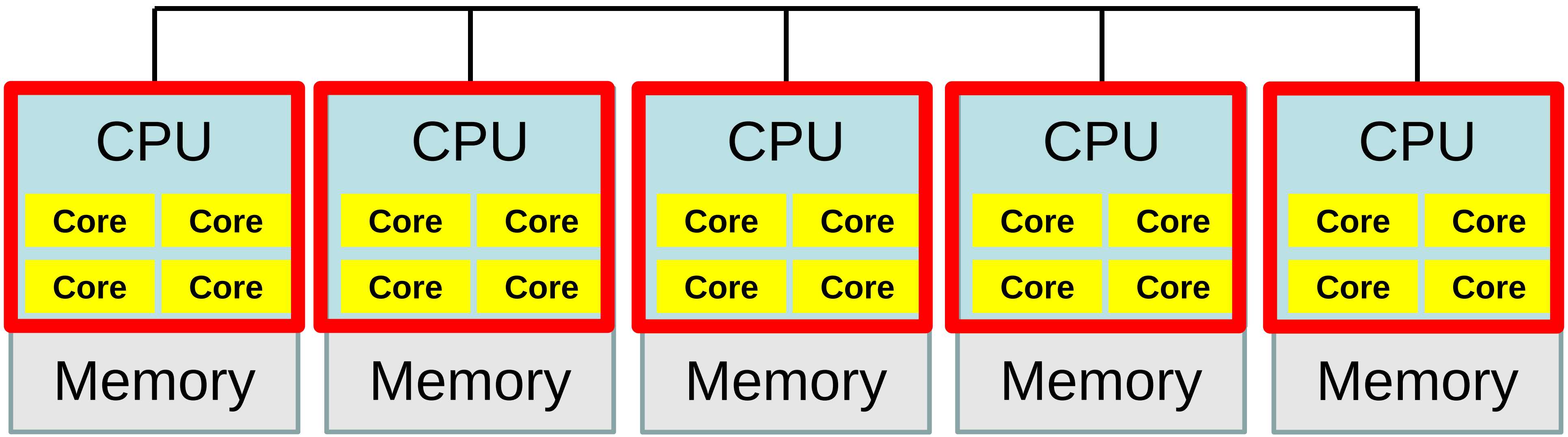
Multicore CPU's are connected through network



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

Our Current Target: Multicore Cluster

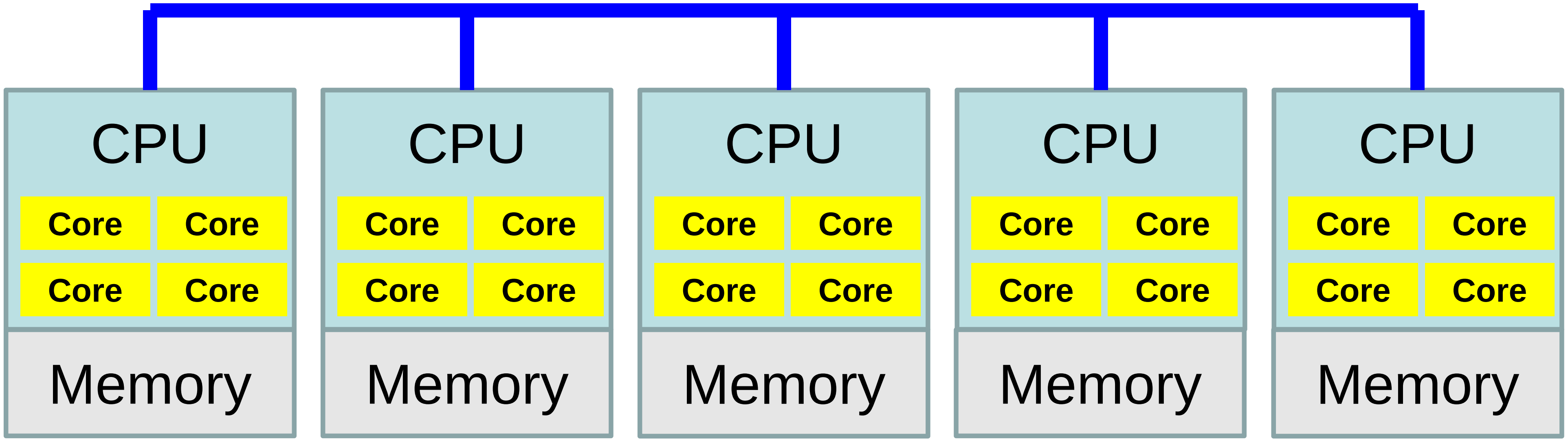
Multicore CPU's are connected through network



- 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

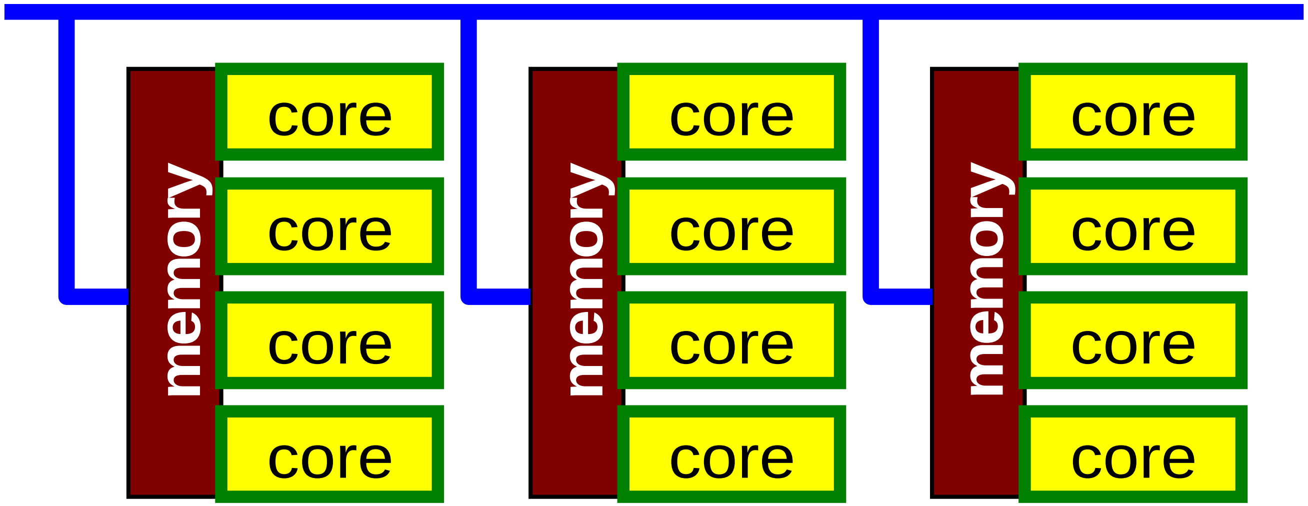


- OpenMP
  - ✓ Multithreading
  - ✓ Intra Node (Intra CPU)
  - ✓ Shared Memory
- MPI
  - ✓ Message Passing
  - ✓ Inter Node (Inter CPU)
  - ✓ Distributed Memory

Flat MPI vs. Hybrid

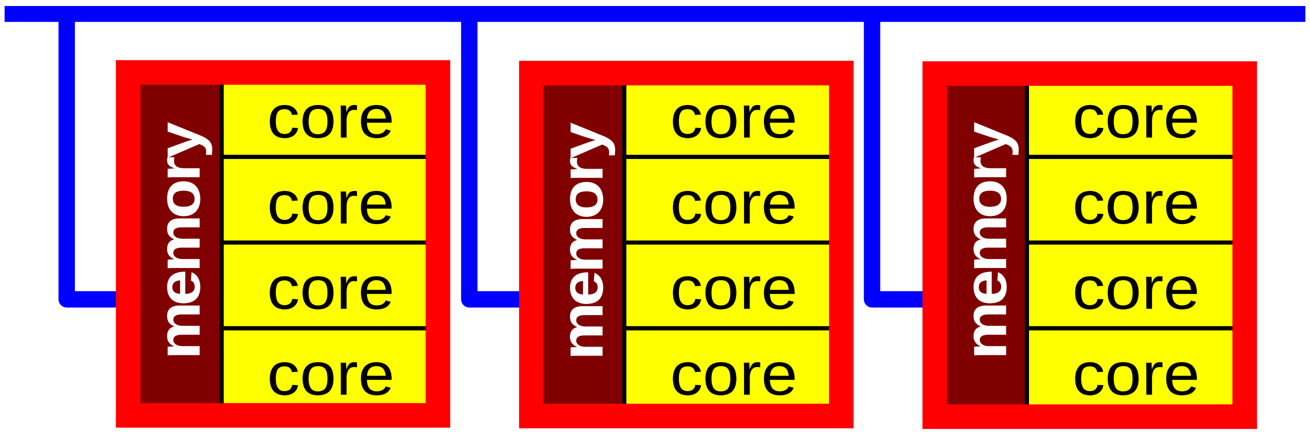
Flat-MPI：Each Core -> Independent

- MPI only
- Intra/Inter Node



Hybrid：Hierarchal Structure

- OpenMP
- MPI



Example of OpnMP/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-NEO)= X(NOD_EXPORT(k))
        enddo

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

- In order to make full use of modern supercomputer systems with multicore/manycore architectures, **hybrid parallel programming with message-passing and multithreading is essential**.
- MPI for message–passing and OpenMP for multithreading are the most popular ways for parallel programming on multicore/manycore clusters.
- In this class, we “parallelize” a finite-volume method code with Krylov iterative solvers for Poisson’s equation on Fujitsu PRIMEHPC FX10 supercomputer at the University of Tokyo (Oakleaf-FX) .
- **Because of limitation of time, we are (mainly) focusing on multithreading by OpenMP.**
- ICCG solver (Conjugate Gradient iterative solvers with Incomplete Cholesky preconditioning) is a widely-used method for solving linear equations.
- Because it includes “data dependency” where writing/reading data to/from memory could occur simultaneously, parallelization using OpenMP is not straight forward.

- We need certain kind of reordering in order to extract parallelism.

- Lectures and exercise on the following issues **<u>related to OpenMP</u>** will be conducted:
    - ✓ Finite-Volume Method (FVM)
    - ✓ Kyrilov Iterative Method
    - ✓ Preconditioning
    - ✓ Implementation of the Program
    - ✓ Introduction to OpenMP
    - ✓ Reordering/Coloring Method
    - ✓ Parallel FVM Code using OpenMP

- (If we have time) Lectures and exercise on the following issues **<u>related to MPI (and OpenMP)</u>** will be conducted:
    - ✓ Parallel FVM on Distributed Memory Systems
    - ✓ **<u>Very Brief Introduction of MPI</u>**
    - ✓ Data Structure for Parallel FVM
    - ✓ **<u>Parallel FVM Code using OpenMP/MPI Hybrid Parallel Programming Model</u>**

Date	ID	Title
Apr-06 (M)	CS-01	Introduction
Apr-13 (M)	CS-02	FVM (1/2)
Apr-20 (M)	CS-03	FVM (2/2)
Apr-27 (M)	CS-04	Login to FX10, OpenMP (1/2)
May-11 (M)	CS-05	OpenMP (2/2)
May-18 (M)	CS-06	Reordering (1/2)
May-25 (M)	CS-07	Reordering (2/2)
May-28 (Th)	**CS-08**	**Parallel Code by OpenMP (1/2)**
Jun-01 (M)	**(canceled)**	
Jun-08 (M)	CS-09	Parallel Code by OpenMP (2/2)
Jun-15 (M)	CS-10	OpenMP/MPI Hybrid (1/3)
Jun-22 (M)	**(canceled)**	
Jun-26 (F)	**CS-11**	**OpenMP/MPI Hybrid (2/3)**
Jun-29 (M)	**(canceled)**	
Jul-06 (M)	CS-12	OpenMP/MPI Hybrid (3/3)

"Prerequisites"

- Fundamental physics and mathematics
  - Linear algebra, analytics
- Experiences in fundamental numerical algorithms
  - LU factorization/decomposition, Gauss-Seidel
- Experiences in programming by C or Fortran
- Experiences and knowledge in UNIX
- User account of ECCS2012 must be obtained:
  - http://www.ecc.u-tokyo.ac.jp/doc/announce/newuser.html

Strategy

- If you can develop programs by yourself, it is ideal... but difficult.
  - focused on “reading”, not developing by yourself
  - Programs are in C and Fortran
    - Lectures are done by ...
- Lecture Materials
  - available at **NOON Friday** through WEB.
    - http://nkl.cc.u-tokyo.ac.jp/15s/
  - NO hardcopy is provided (Today is exceptional)
- Starting at 08:30
  - You can enter the building after 08:00
- Taking seats from the front row.
- Terminals must be shut-down after class.

Grades

- 1 or 2 Reports on programming

If you have any questions, please feel free to contact me !

- Office: 3F Annex/Information Technology Center #36
  - 情報基盤センター別館3F 36号室
- ext.: 22719
- e-mail: nakajima(at)cc.u-tokyo.ac.jp
- **NO specific office hours, appointment by e-mail**

- http://nkl.cc.u-tokyo.ac.jp/15s/
- http://nkl.cc.u-tokyo.ac.jp/seminars/2015-Spring/ 日本語資料（一部）

Keywords for OpenMP

- OpenMP
  - Directive based, (seems to be) easy
  - Many books

- Data Dependency
  - Conflict of reading from/writing to memory
  - Appropriate reordering of data is needed for “consistent” parallel computing
  - NO detailed information in OpenMP books: very complicated

Some Technical Terms

- Processor, Core
  - Processing Unit (H/W), Processor=Core for single-core proc's
- Process
  - Unit for MPI computation, nearly equal to "core"
  - Each core (or processor) can host multiple processes (but not efficient)
- PE (Processing Element)
  - PE originally mean "processor", but it is sometimes used as "process" in this class. Moreover it means "domain" (next)
    - In multicore proc's: PE generally means "core"
- Domain
  - domain=process (=PE), each of "MD" in "SPMD", each data set
- Process ID of MPI (ID of PE, ID of domain) starts from "0"
  - if you have 8 processes (PE's, domains), ID is 0~7

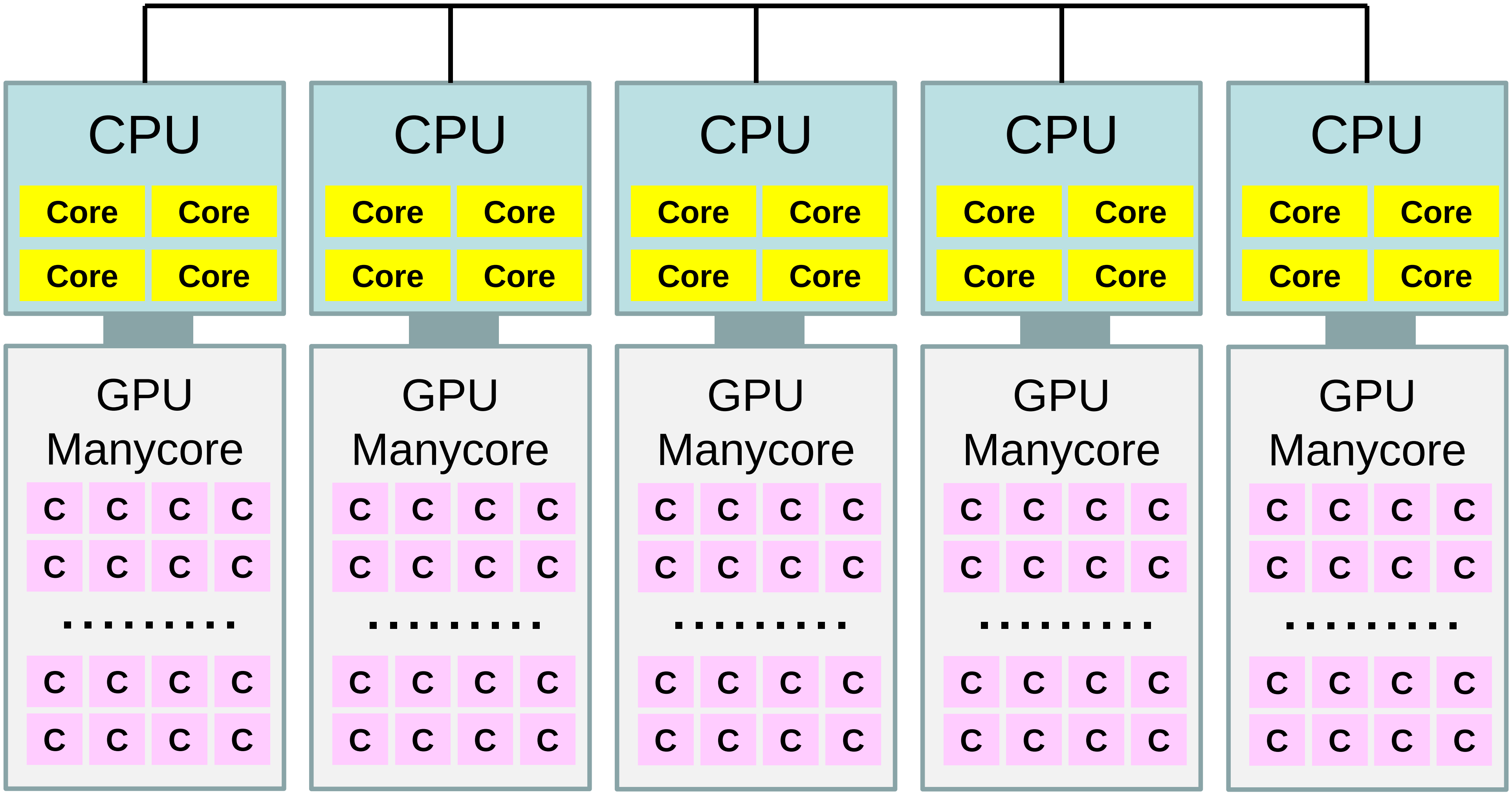
- Supercomputers and Computational Science
- Overview of the Class
- **Future Issues**

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

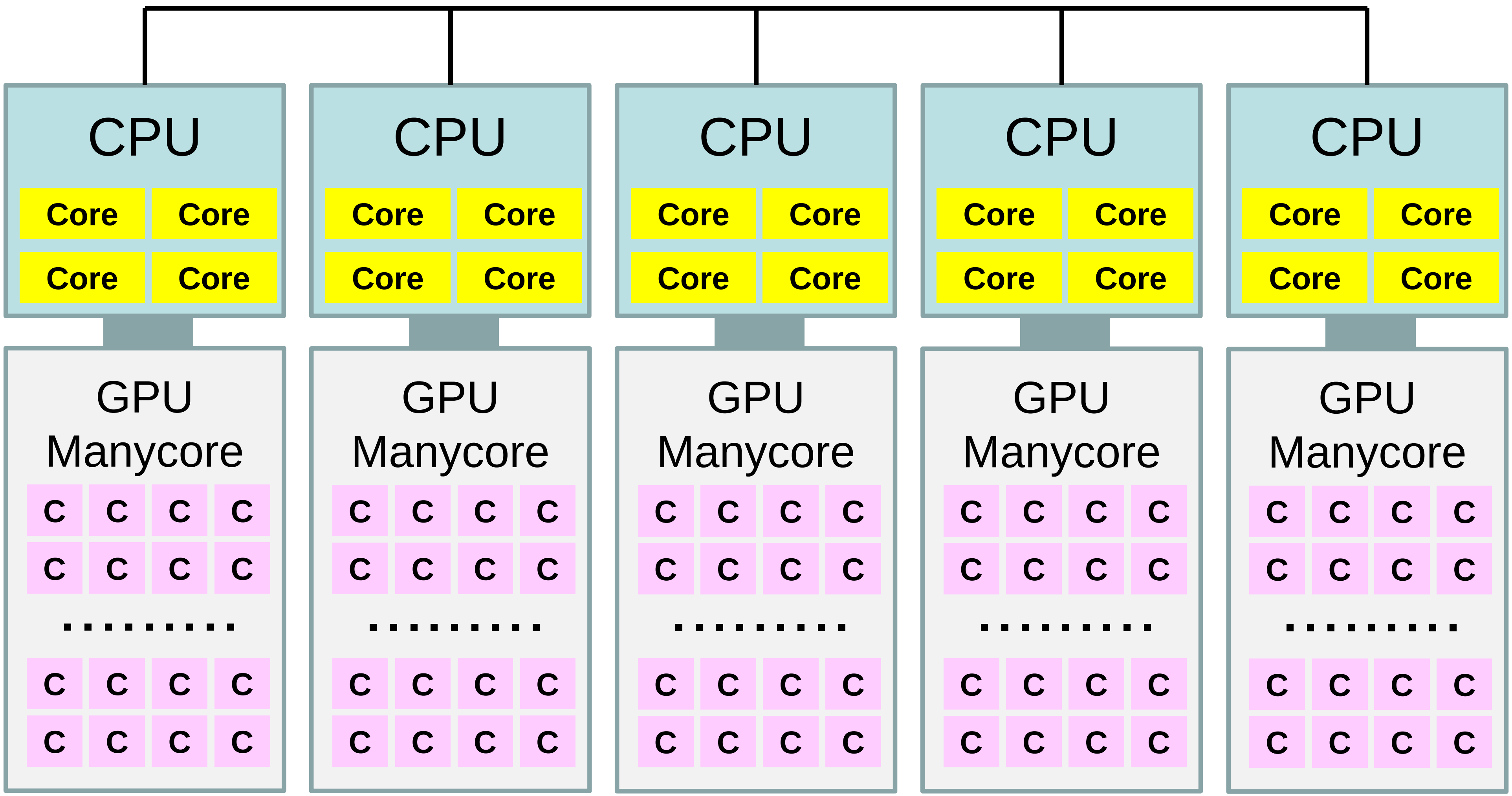
Supercomputers with Heterogeneous/Hybrid Nodes



Hybrid Parallel Programming Model is essential for Post-Peta/Exascale Computing

- Message Passing (e.g. MPI) + Multi Threading (e.g. OpenMP, CUDA, OpenCL, OpenACC etc.)
- In K computer and FX10, hybrid parallel programming is recommended
  - MPI + Automatic Parallelization by Fujitsu's Compiler
- Expectations for Hybrid
  - Number of MPI processes (and sub-domains) to be reduced
  - O(10^8-10^9)-way MPI might not scale in Exascale Systems
  - Easily extended to Heterogeneous Architectures
    - CPU+GPU, CPU+Manycores (e.g. Intel MIC/Xeon Phi)
    - MPI+X: OpenMP, OpenACC, CUDA, OpenCL

This class is also useful for this type of parallel system



Parallel Programming Models

- Multicore Clusters (e.g. K, FX10)
  - MPI + OpenMP and (Fortan/C/C++)
- Multicore + GPU (e.g. Tsubame)
  - GPU needs host CPU
  - MPI and [(Fortan/C/C++) + CUDA, OpenCL]
    - complicated,
  - MPI and [(Fortran/C/C++) with OpenACC]
    - close to MPI + OpenMP and (Fortran/C/C++)
- Multicore + Intel MIC/Xeon-Phi (e.g. Stampede)
  - Xeon-Phi needs host CPU (currently)
  - MPI + OpenMP and (Fortan/C/C++) is possible
    - + Vectorization

Future of Supercomputers (1/2)

- Technical Issues
  - Power Consumption
  - Reliability, Fault Tolerance, Fault Resilience
  - Scalability (Parallel Performancce)
- Petascale System
  - 2MW including A/C, 2M$/year, O($10^5$～$10^6$) cores
- Exascale System (10^3x Petascale)
  - 2020-2023 (?)
    - 2GW (2 B$/year !), O($10^8$～$10^9$) cores
  - Various types of innovations are on-going
    - to keep power consumption at 20MW (100x efficiency)
    - CPU, Memory, Network ...
  - Reliability

Future of Supercomputers (2/2)

- Not only hardware, but also numerical models and algorithms must be improved:
  - 省電力（Power-Aware/Reducing Algorithms）
  - 耐故障（Fault Resilient Algorithms）
  - 通信削減（Communication Avoiding/Reducing Algorithms）

- Co-Design by experts from various area (SMASH) is important
  - Exascale system will be a special-purpose system, not a general-purpose one.