東京大学
THE UNIVERSITY OF TOKYO

全国共同利用施設
東京大学情報基盤センター
Information Technology Center, The University of Tokyo

XcalableMP

Parallel Programming Support for Applications with Unstructured Meshes

Expectations for "Local View" of XcalableMP

Kengo NAKAJIMA
Information Technology Center
The University of Tokyo

International Workshop on Peta-Scale Computing Programming Environment, Languages and Tools (WPSE 2009), March 25-26 2009, Tsukuba, Japan

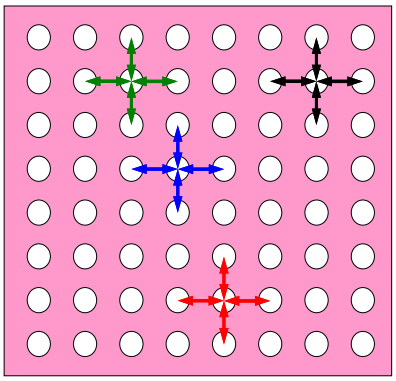
First of All

- MY background
  - Computational Mechanics, Computational Fluid Dynamics
    - FEM, FDM, FVM, BEM …
  - Iterative Linear Solvers, Parallel Preconditioning Methods
  - Parallel Programming Models
- I am a member of “XcalableMP” team, but
  - pro-MPI
  - or pro-Hybrid (MPI+OpenMP)

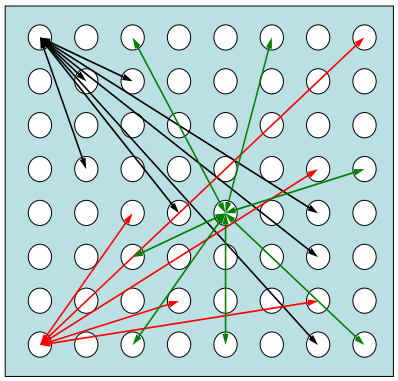
- Introduction
  - 3-min Introduction to Finite Element Method
- Parallel Finite-Element Methods
  - GeoFEM
  - Various Capabilities
- Critical Problems in Parallel FEM's
  - Expectations to ***XcalableMP***

Two Types of Applications

- Local
  - FEM, FDM, FVM
  - Local Operation
  - Comm. with Neighbors
  - Sparse Matrix
  - Memory/Latency-Bound

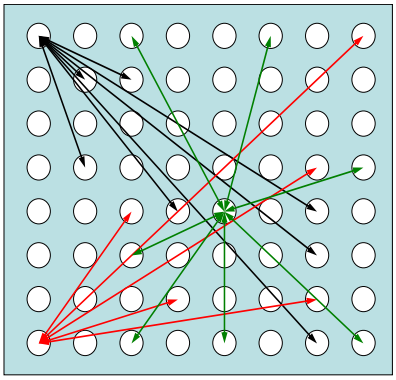


- Global
  - BEM, BIM, MD, Spectral
  - Global Operation
  - All-to-All Communications
  - Dense Matrix
  - Communication (BW)-Bound

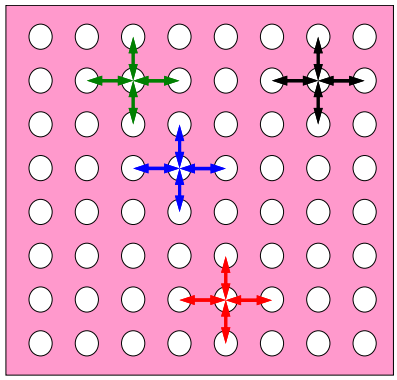


XcalableMP

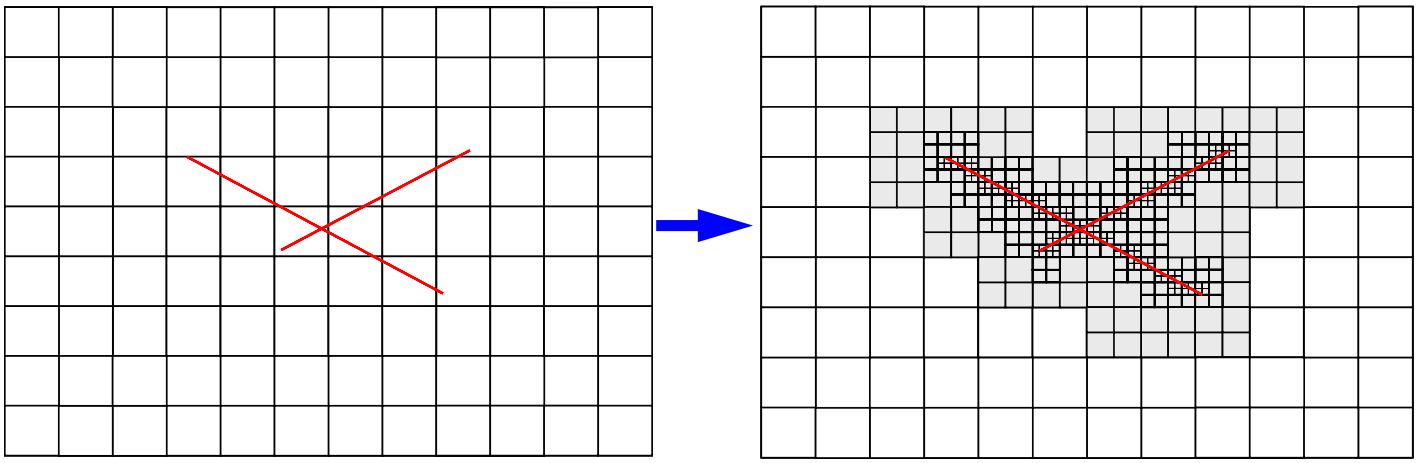
- Global View
  - “Global” applications
    - Generally, structured data
  - “Local” applications with Structured Grids (e.g. FDM)



- Local View
  - “Local” applications with Unstructured Meshes (e.g. FEM)
    - Suitable for Complicated Geometries, Flexible
  - Pure SPMD style



FDM with AMR (Adaptive Mesh Refinement): Unstructured



XcalableMP

- Global View
  - Regular Grid

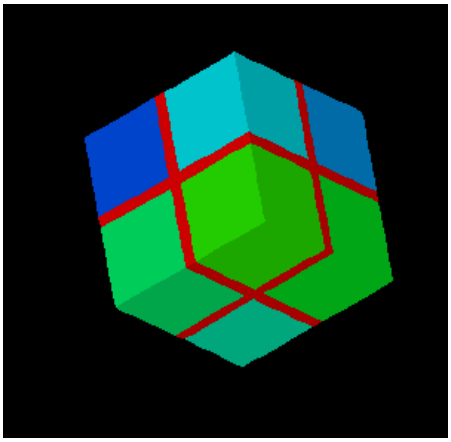
```
do k= 1, KMAX
  do j= 1, JMAX
    do i= 1, IMAX
      Y(i,j,k)=…
    enddo
  enddo
enddo
```

- Local View
  - Irregular Mesh

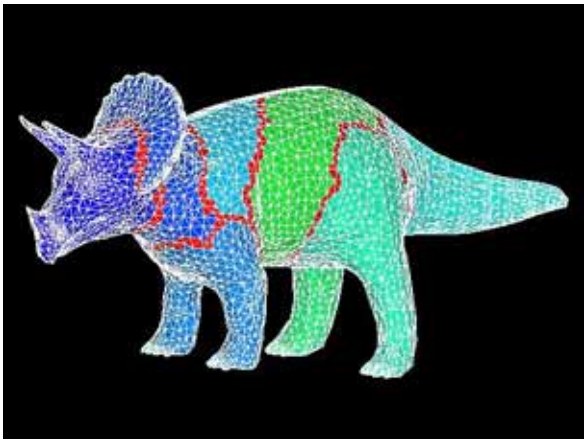
```
do i= 1, N
  Y(i)=…
enddo
```


XcalableMP

- Global View
  - Regular Grid



- Local View
  - Irregular Mesh

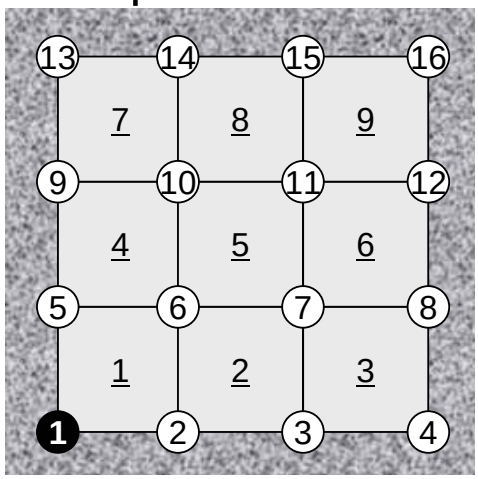


Finite-Element Method (FEM)

- 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 (λ=1)
  - uniform volume heat flux (Q=1)
  - T=0 at node 1
  - Insulated boundaries



Galerkin FEM procedures

- Apply Galerkin procedures to each element:

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

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

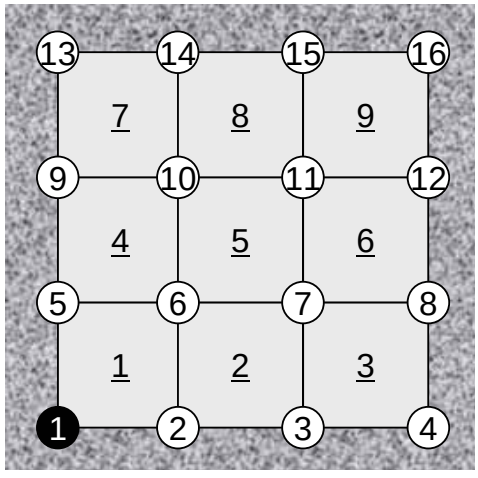
$\{\phi\}$: T at each vertex

$[N]$: Shape function (Interpolation function)

- Introduce the following "weak form" of original PDE using Green's theorem:

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

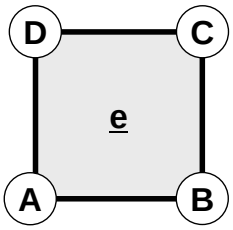
$$+\int_V Q[N]^T dV=0$$



Element Matrix

- Apply the integration to each element and form "element" matrix.

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

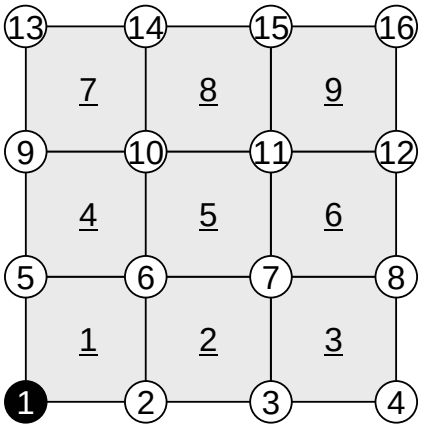


$$[k^{(e)}]\{\phi^{(e)}\} = \{f^{(e)}\}$$

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

Global (Overall) Matrix

Accumulate each element matrix to "global" matrix.

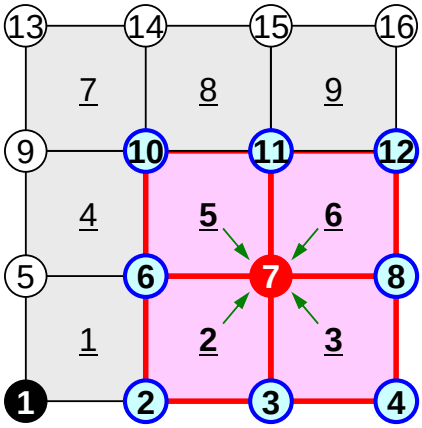


$$[K]\{\Phi\} = \{F\}$$

$$\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
 & X & D & X & & X & X & X & & & & & & & & \\
 & & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
 & X & X & X & & X & D & X & & X & X & X & & & & \\
 & & X & X & & & X & D & & & X & X & & & & \\
 & & & & X & X & & & D & X & & & X & X & & \\
 & & & & X & X & X & & X & D & X & & X & X & X & \\
 & & & & & X & X & X & & X & D & X & & X & X & X \\
 & & & & & & X & X & & & X & D & & & X & X \\
 & & & & & & & & X & X & & & D & X & & \\
 & & & & & & & & X & X & X & & X & D & X & \\
 & & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_6 \\ \Phi_7 \\ \Phi_8 \\ \Phi_9 \\ \Phi_{10} \\ \Phi_{11} \\ \Phi_{12} \\ \Phi_{13} \\ \Phi_{14} \\ \Phi_{15} \\ \Phi_{16} \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16} \end{Bmatrix}$$

To each node …

Effect of surrounding elem's/nodes are accumulated.



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

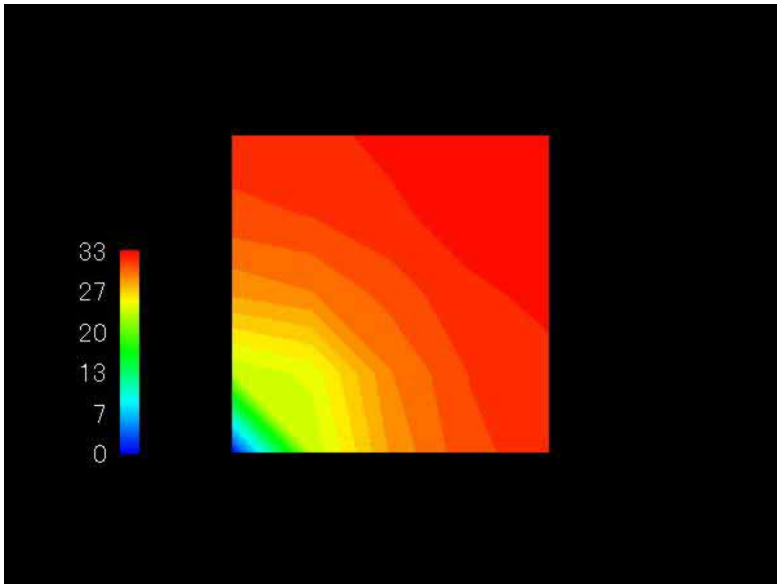
$$\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
 & X & D & X & & X & X & X & & & & & & & & \\
 & & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
 & X & X & X & & X & D & X & & X & X & X & & & & \\
 & & X & X & & & X & D & & & X & X & & & & \\
 & & & & X & X & & & D & X & & & X & X & & \\
 & & & & X & X & X & & X & D & X & & X & X & X & \\
 & & & & & X & X & X & & X & D & X & & X & X & X \\
 & & & & & & X & X & & & X & D & & & X & X \\
 & & & & & & & & X & X & & & D & X & & \\
 & & & & & & & & X & X & X & & X & D & X & \\
 & & & & & & & & & X & X & X & & X & D & X \\
 & & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix}\Phi_1\\ \Phi_2\\ \Phi_3\\ \Phi_4\\ \Phi_5\\ \Phi_6\\ \Phi_7\\ \Phi_8\\ \Phi_9\\ \Phi_{10}\\ \Phi_{11}\\ \Phi_{12}\\ \Phi_{13}\\ \Phi_{14}\\ \Phi_{15}\\ \Phi_{16}\end{Bmatrix}
=
\begin{Bmatrix}F_1\\ F_2\\ F_3\\ F_4\\ F_5\\ F_6\\ F_7\\ F_8\\ F_9\\ F_{10}\\ F_{11}\\ F_{12}\\ F_{13}\\ F_{14}\\ F_{15}\\ F_{16}\end{Bmatrix}$$

Solve the obtained global/overall equations

under certain boundary conditions (Φ_1=0 in this case)

$$\begin{bmatrix}
D & X & & & X & X & & & & & & & & & & \\
X & D & X & & X & X & X & & & & & & & & & \\
 & X & D & X & & X & X & X & & & & & & & & \\
 & & X & D & & & X & X & & & & & & & & \\
X & X & & & D & X & & & X & X & & & & & & \\
X & X & X & & X & D & X & & X & X & X & & & & & \\
 & X & X & X & & X & D & X & & X & X & X & & & & \\
 & & X & X & & & X & D & & & X & X & & & & \\
 & & & & X & X & & & D & X & & & X & X & & \\
 & & & & X & X & X & & X & D & X & & X & X & X & \\
 & & & & & X & X & X & & X & D & X & & X & X & X \\
 & & & & & & X & X & & & X & D & & & X & X \\
 & & & & & & & & X & X & & & D & X & & \\
 & & & & & & & & X & X & X & & X & D & X & \\
 & & & & & & & & & X & X & X & & X & D & X \\
 & & & & & & & & & & X & X & & & X & D
\end{bmatrix}
\begin{Bmatrix}\Phi_1\\ \Phi_2\\ \Phi_3\\ \Phi_4\\ \Phi_5\\ \Phi_6\\ \Phi_7\\ \Phi_8\\ \Phi_9\\ \Phi_{10}\\ \Phi_{11}\\ \Phi_{12}\\ \Phi_{13}\\ \Phi_{14}\\ \Phi_{15}\\ \Phi_{16}\end{Bmatrix}
=
\begin{Bmatrix}F_1\\ F_2\\ F_3\\ F_4\\ F_5\\ F_6\\ F_7\\ F_8\\ F_9\\ F_{10}\\ F_{11}\\ F_{12}\\ F_{13}\\ F_{14}\\ F_{15}\\ F_{16}\end{Bmatrix}$$

Result …



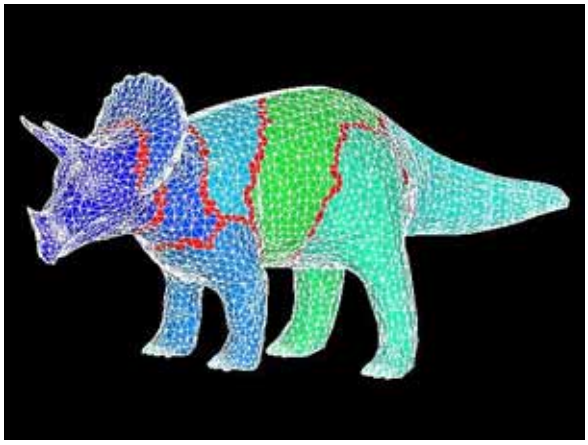
Features of FEM Applications (1/2)

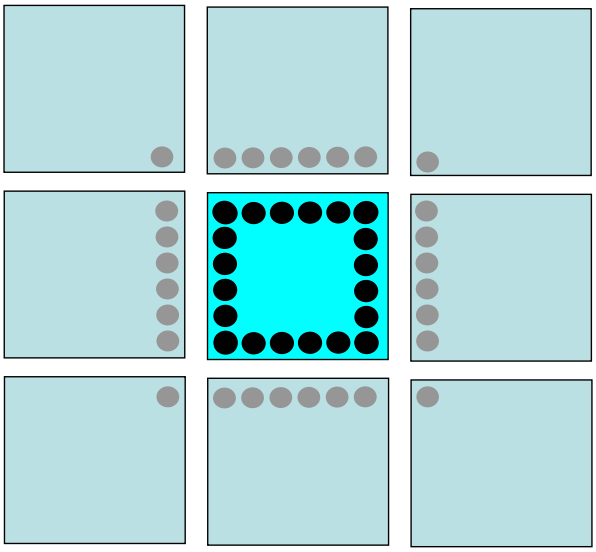
- Local Operations, Element-by-Element
  - Coefficient Matrix: Sparse
  - Good Feature for Parallel Operations
- Irregular Sparse Matrices
  - Indirect Access
  - Memory-Bound

```
do i= 1, N
  jS= index(i-1)+1
  jE= index(i)
  do j= jS, jE
    in  = item(j)
    Y(i)= Y(i) + AMAT(j)*X(in)
  enddo
enddo
```

Features of FEM Applications (1/2)

- Parallel FEM with Domain Decomposition
  - Communications with ONLY Neighbors
  - Small Amount of Communications
  - Latency-Bound





- Introduction
  - 3-min Introduction to Finite Element Method
- Parallel Finite-Element Methods
  - GeoFEM
  - Various Capabilities
- Critical Problems in Parallel FEM's

GeoFEM: FY.1998-2002

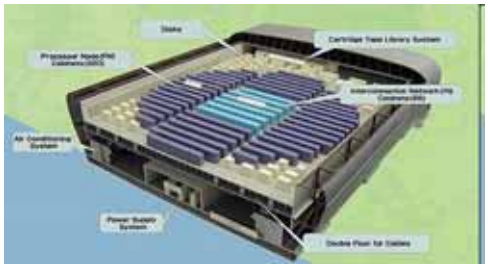
http://geofem.tokyo.rist.or.jp/

- Parallel FEM platform for solid earth simulation.
  - parallel I/O, **parallel linear solvers**, parallel visualization
  - solid earth: earthquake, plate deformation, mantle/core convection, etc.
- Part of national project by STA/MEXT for large-scale earth science simulations using the Earth Simulator.
- Strong collaborations between computer/computational science and natural science (solid earth) communities.
- Started from development of a parallel FEM application for solid mechanics
  - Common capabilities have been extracted for “platform”

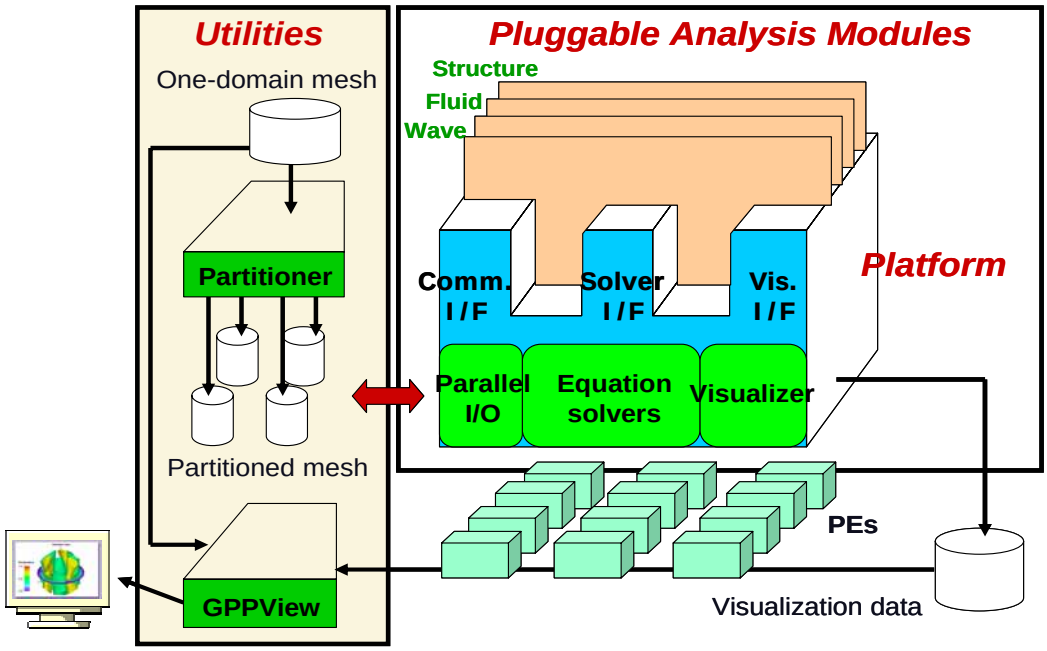
Earth Simulator (ES)

http://www.es.jamstec.go.jp/

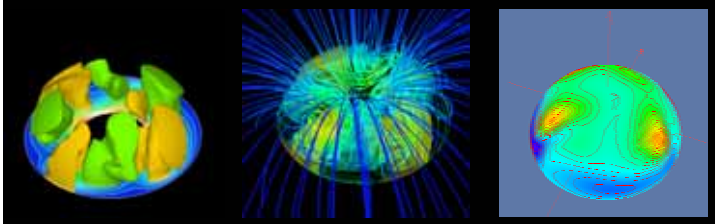
- 640×8= 5,120 Vector Processors
  - **SMP Cluster-Type Architecture**
  - 8 GFLOPS/PE
  - 64 GFLOPS/Node
  - 40 TFLOPS/ES
- 16 GB Memory/Node, 10 TB/ES
- 640×640 Crossbar Network
  - 12.3 GB/sec×2
- Memory BWTH with 32 GB/sec.
- **35.6 TFLOPS for LINPACK (2002-March)**
  - **14th in Nov.06 list (Jun.07 list this week)**
- **26 TFLOPS for AFES (Climate Simulation)**



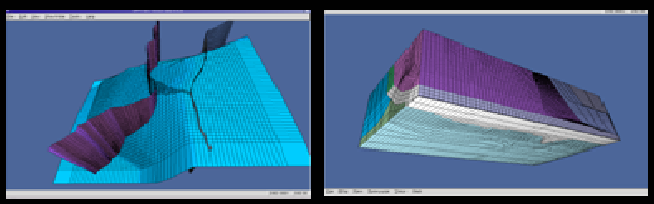
System Configuration of GeoFEM, FEM: Modularity



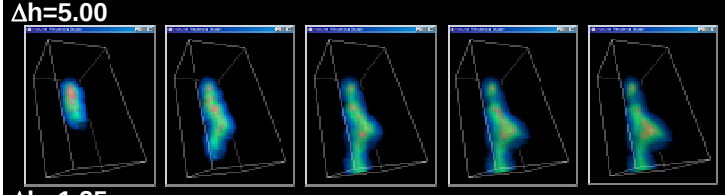
Results on Solid Earth Simulation



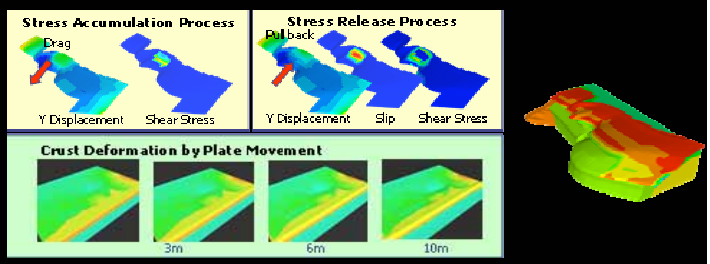
Magnetic Field of the Earth : MHD code



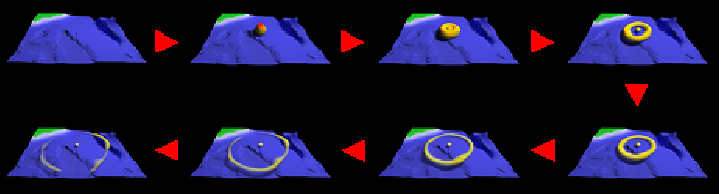
Complicated Plate Model around Japan Islands



Transportation by Groundwater Flow through Heterogeneous Porous Media



Simulation of Earthquake Generation Cycle in Southwestern Japan



TSUNAMI !!

Goal of GeoFEM (Fall 1997)

as Environment for Development of Parallel FEM Applications

- NO MPI call's in user's code !!!!!
- As serial as possible !!!!!
  - Original FEM code developed for single CPU machine can work on parallel computers with smallest modification.
- Careful design of the local data structure for distributed parallel computing is (was) very important
  - This was the most critical part for the "success" of GeoFEM

Six reasons why HPF failed

Brad Chamberlain (Cray/Chapel)

- Lack of good performance soon enough, lack of patience from the use community
- Lack of portable performance model, execution model
- Inability to drop the lower levels of parallel computation
- Lack of rich abstraction
  - reasonable support for dense multidimensional arrays
  - but not for sparse data structures
- Lack of general parallel programming models
- Lack of an open source implementation

H.Murai (NEC/XcalableMP)

Parallel Computing in GeoFEM

Algorithms: Parallel Iterative Solvers & Local Data Structure

- Parallel Iterative Solvers by (Fortran90+MPI)
  - Iterative method is the only choice for large-scale problems with parallel processing.
  - Portability is important -> from PC clusters to Earth Simulator
- Appropriate Local Data Structure for (FEM+Parallel Iterative Method)
  - FEM is based on local operations.

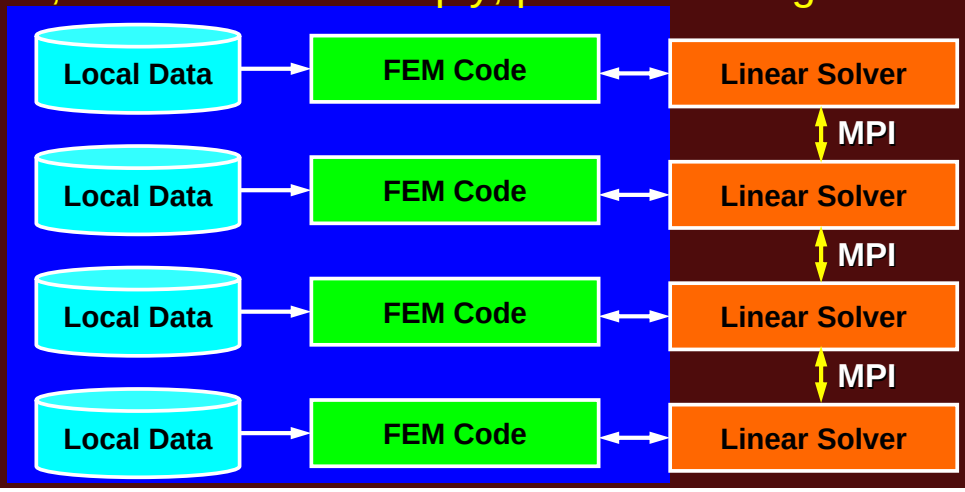
Parallel Computing in FEM

SPMD: Single-Program Multiple-Data

Large Scale Data -> partitioned into Distributed Local Data Sets.

FEM code on each PE assembles coefficient matrix for each local data set : this part is completely local, same as serial operations

Global Operations & Communications happen only in Linear Solvers dot products, matrix-vector multiply, preconditioning



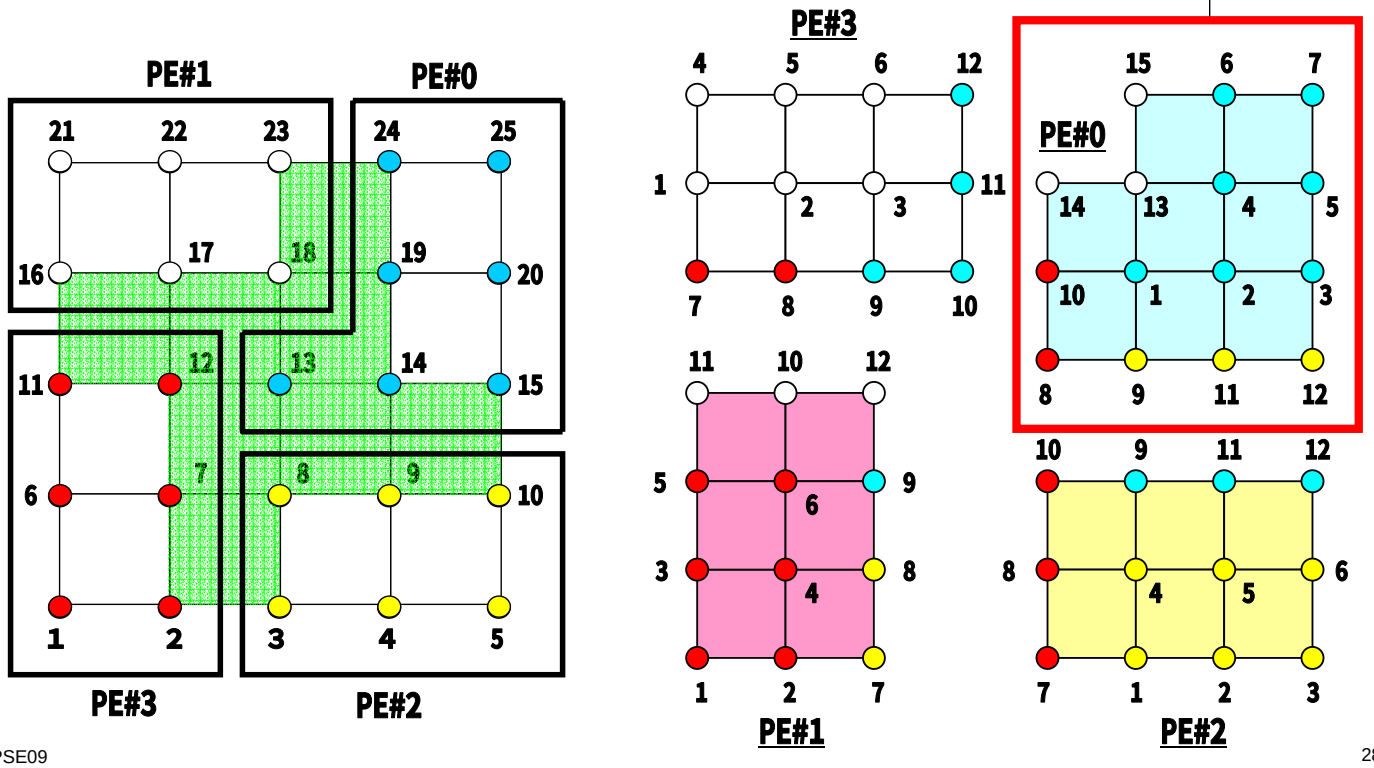
Parallel Computing in GeoFEM

- Finally, users can develop parallel FEM codes easily using GeoFEM without considering parallel operations.
  - Local data structure and linear solvers do it.
  - Basically, same procedures as those of serial operations.
  - This is possible because FEM is based on local operations. FEM is really suitable for parallel computing.

- ***NO MPI in user's code***
- ***Plug-in***

Node-based Partitioning

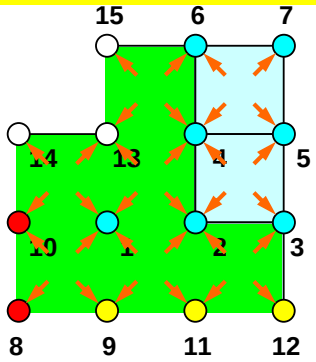
internal nodes - elements - external nodes

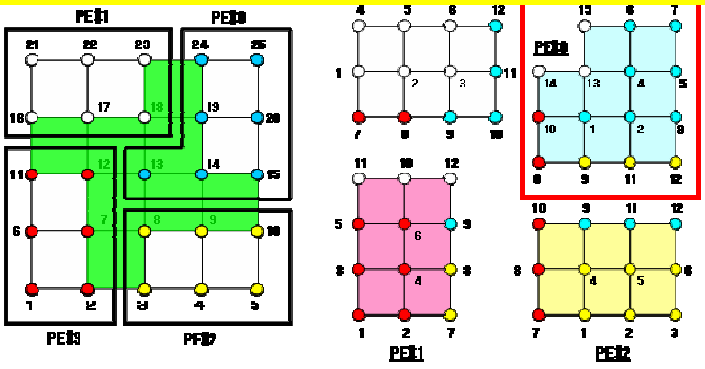


Node-based Partitioning

internal nodes - elements - external nodes

- Partitioned nodes themselves (Internal Nodes)
- Elements which include Internal Nodes
- External Nodes included in the Elements
  in overlapped region among partitions.
- Info of External Nodes are required for completely local element–based operations on each processor.





What is Communication ?

- Getting Information for EXTERNAL NODES from EXTERNAL PARTITIONS.
- “Communication tables” in local data structure includes the procedures for communication.

How to “Parallelize” Iterative Solvers ?

e.g. CG method (with no preconditioning)

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

- Parallel procedures are required in:
  - Dot products
  - Matrix-vector multiplication

How to “Parallelize” Dot Products

- use MPI_ALLreduce after local operations

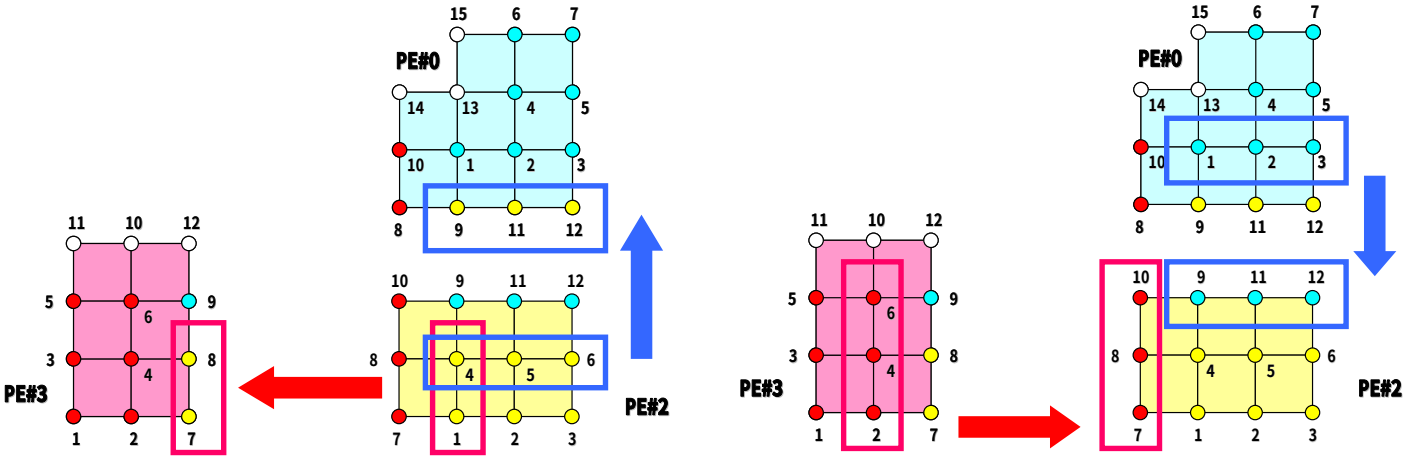
```
RHO= 0.d0

do i= 1, N
  RHO= RHO + W(i,R)*W(i,Z)
enddo

Allreduce RHO
```

How to “Parallelize” Matrix-Vec. Multilplication

- We need values of {p} vector at EXTERNAL nodes BEFORE computation !!

```
get {p} at EXTERNAL nodes
do i= 1, N
  q(i)= D(i)*p(i)
  do k= INDEX_L(i-1)+1, INDEX_U(i)
    q(i)= q(i) + AMAT_L(k)*p(ITEM_L(k))
  enddo

  do k= INDEX_U(i-1)+1, INDEX_U(i)
    q(i)= q(i) + AMAT_U(k)*p(ITEM_U(k))
  enddo
enddo
```


SEND, RECV via Comm. Table



Domain-to-Domain Communication

Exchange Boundary Information (SEND/RECV)

```
subroutine SOLVER_SEND_RECV                          &
           (N, NEIBPETOT, NEIBPE,                    &
            IMPORT_INDEX, IMPORT_NODE,               &
            EXPORT_INDEX, EXPORT_NODE,               &
            WS, WR, X, SOLVER_COMM, my_rank)

implicit REAL*8 (A-H,O-Z)
include 'mpif.h'
parameter (KREAL= 8)
integer IMPORT_INDEX(0:NEIBPETOT), IMPORT_NODE(N)
integer EXPORT_INDEX(0:NEIBPETOT), EXPORT_NODE(N)
integer SOLVER_COMM, my_rank
integer req1(NEIBPETOT), req2(NEIBPETOT)
integer sta1(MPI_STATUS_SIZE, NEIBPETOT)
integer sta2(MPI_STATUS_SIZE, NEIBPETOT)
real(kind=KREAL)  X(N), NEIBPE(NEIBPETOT), WS(N), WR(N)
```

SEND

```
do neib= 1, NEIBPETOT
  istart= EXPORT_INDEX(neib-1)
  inum  = EXPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    WS(k)= X(EXPORT_NODE(k))
  enddo
  call MPI_ISEND
      (WS(istart+1), inum, MPI_DOUBLE_PRECISION,   &
       NEIBPE(neib), 0, SOLVER_COMM,               &
       req1(neib), ierr)
enddo
```

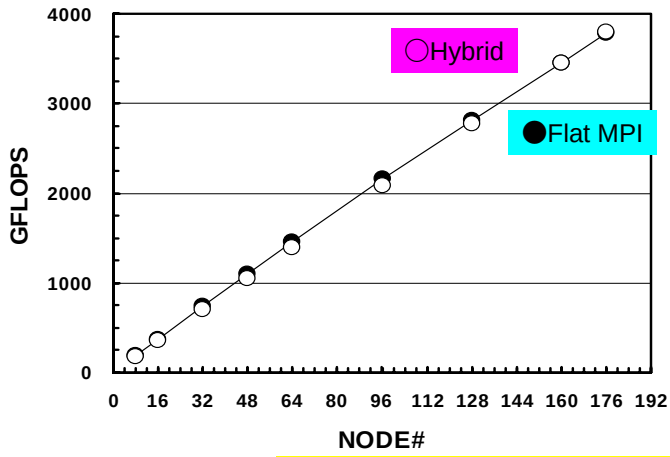
RECEIVE

```
do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  call MPI_IRECV
      (WR(istart+1), inum, MPI_DOUBLE_PRECISION, &
       NEIBPE(neib), 0, SOLVER_COMM,             &
       req2(neib), ierr)
enddo

call MPI_WAITALL (NEIBPETOT, req2, sta2, ierr)

do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    X(IMPORT_NODE(k))= WR(k)
  enddo
enddo

call MPI_WAITALL (NEIBPETOT, req1, sta1, ierr)

return
end
```

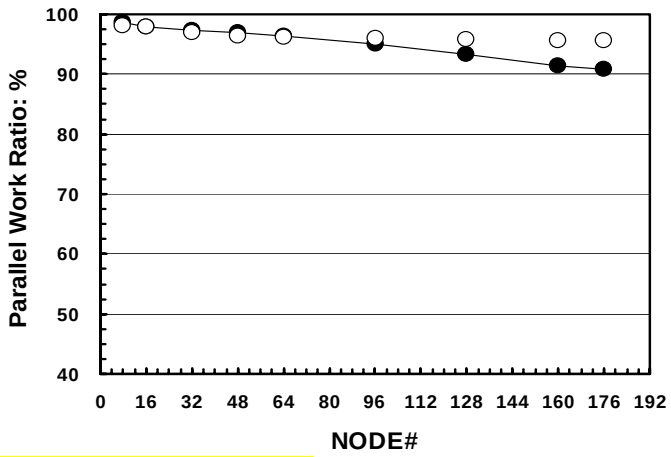
3D Elastic Model (Large Case)

256x128x128/SMP node, up to 2,214,592,512 DOF

GFLOPS rate



Parallel Work Ratio



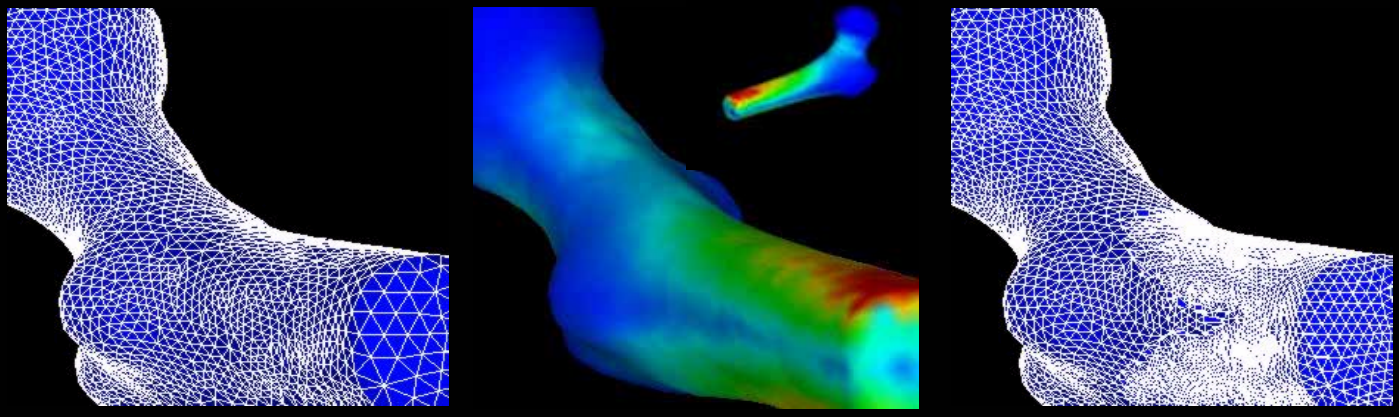
3.8TFLOPS for 2.2G DOF
176 nodes (33.8% of peak)

HPC-MW (FY.2002-FY.2007)

Successor of GeoFEM

- Parallel I/O : I/F for NASTRAN, ABAQUS etc.
- Adaptive Mesh Refinement (AMR)
- Dynamic Load-Balancing using **pMETIS** (DLB)
- Parallel Visualization
- Linear Solvers
- FEM Operations (Matrix Assembling etc.)
- Coupling I/F
- Utility for Mesh Partitioning

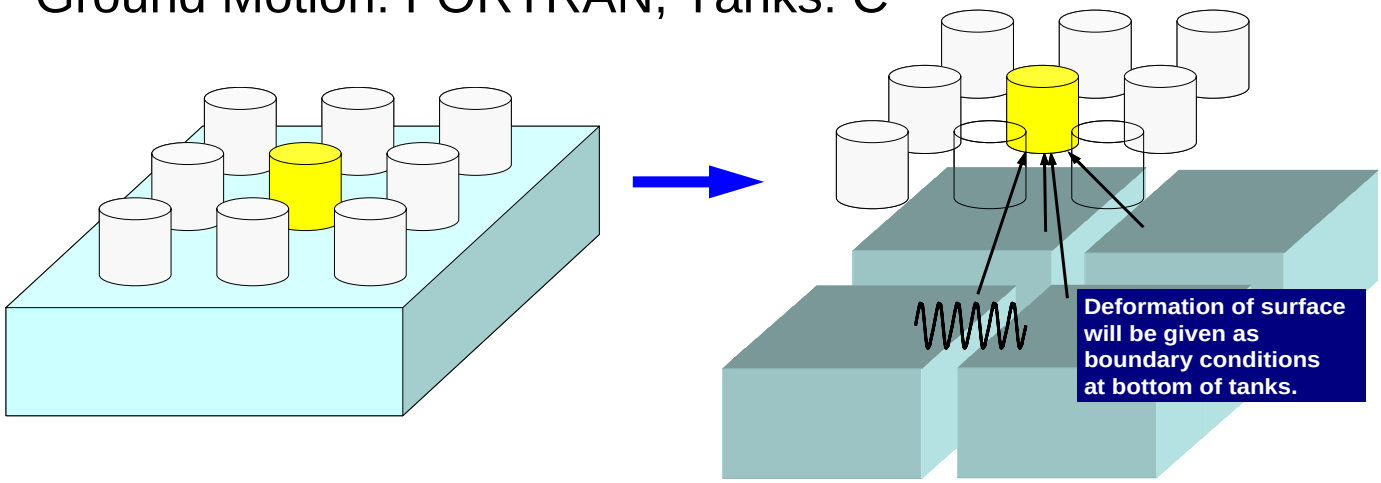
AMR for Solid Mechanics Simulation of Bone



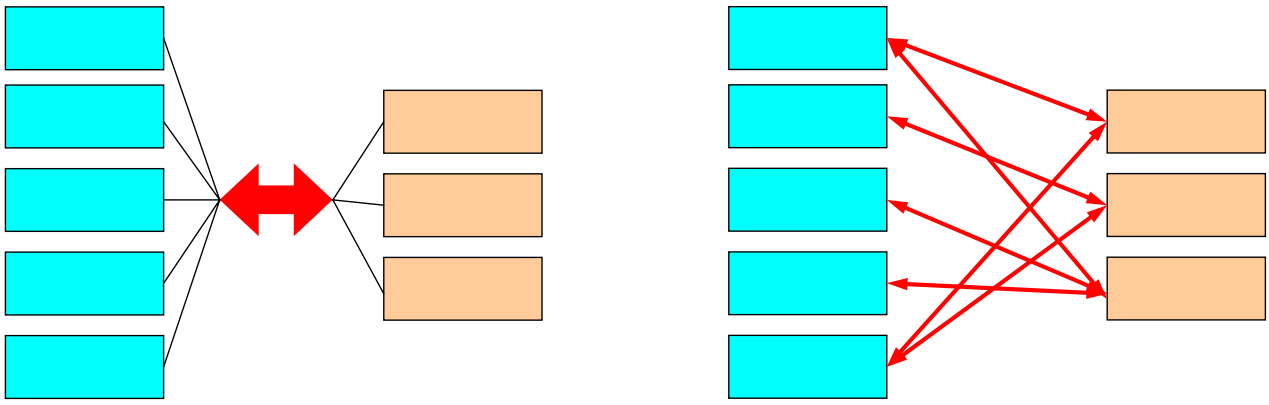
Initial Mesh | Mises Stress | Adapted Mesh

An Example of Coupled Simulation

- Coupling between “Ground Motion” and “Tanks for Oil-Storage”
  - “One-way” coupling from “Ground Motion” to “Tanks”.
  - Displacement of ground surface is given as forced displacement of bottom surface of tanks.
- Ground Motion: FORTRAN, Tanks: C

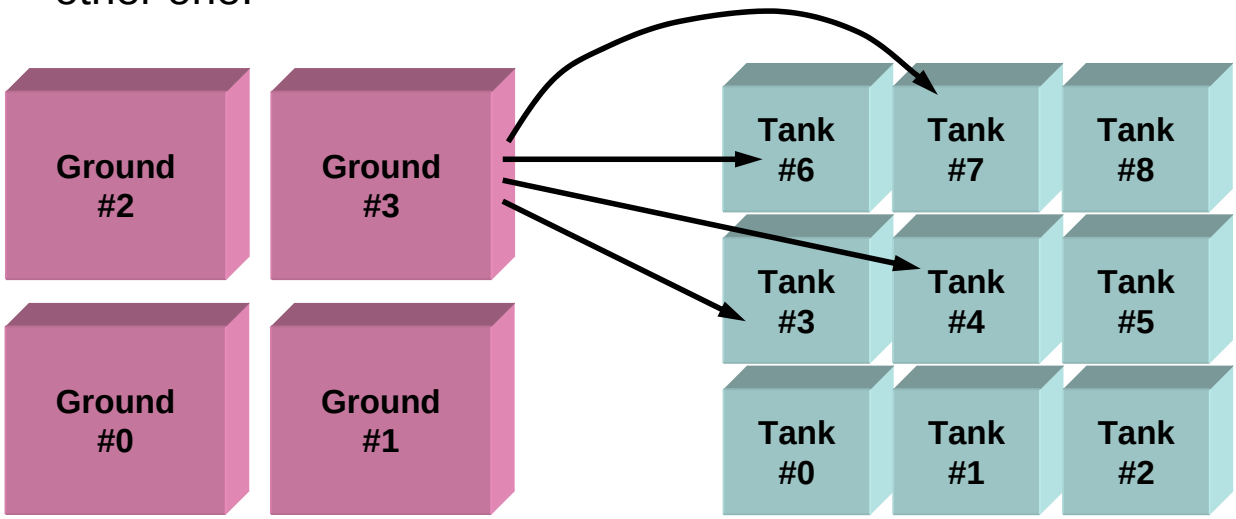


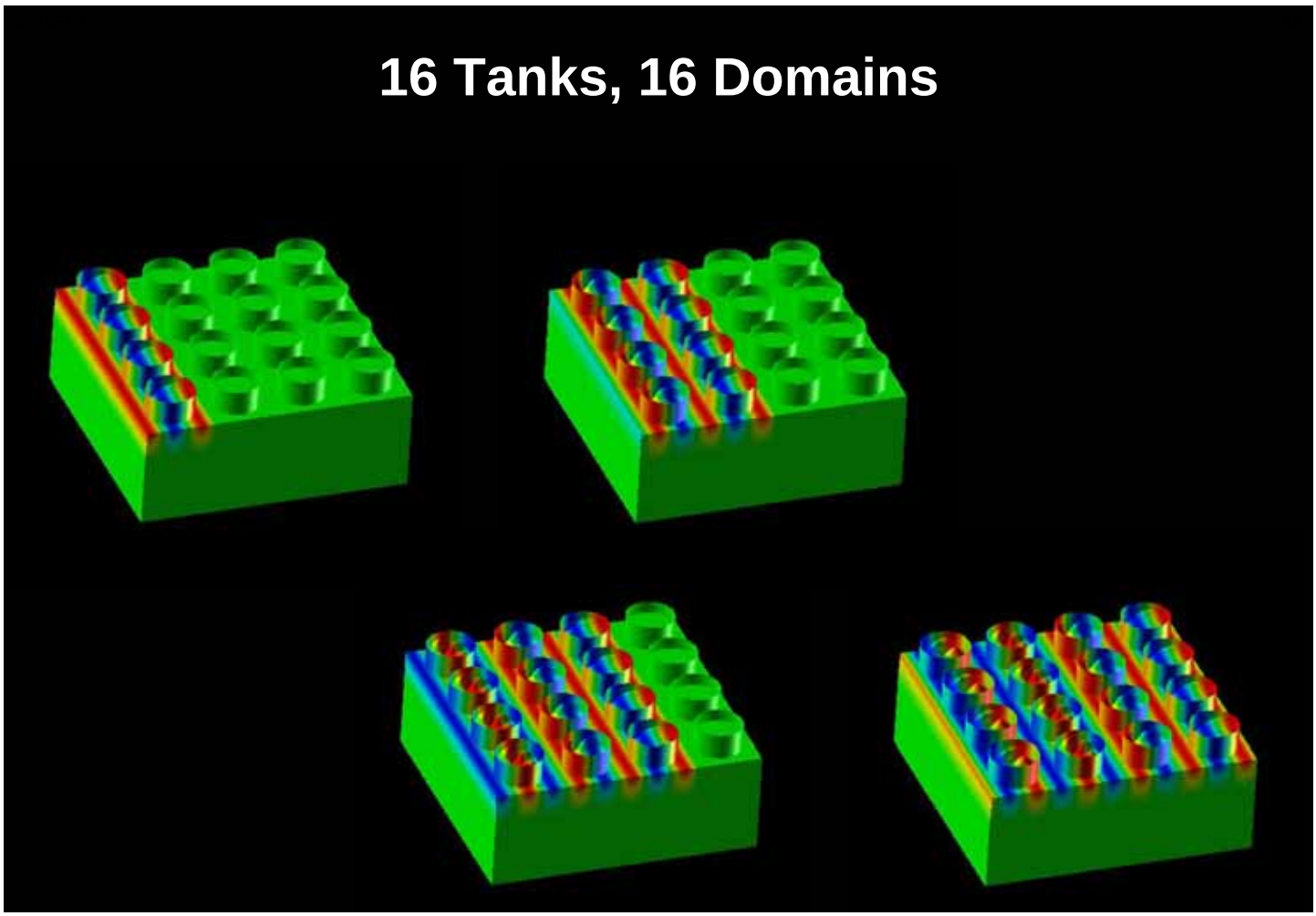
“Centralized” or “Distributed” in Coupled Simulations

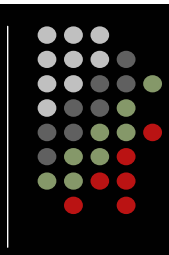


M✕N Parallel Data Redistribution

- The transfer of data from a parallel program running on M processors to another parallel program running on N processors.
  - Ideally neither program knows the number of processes on the other one.







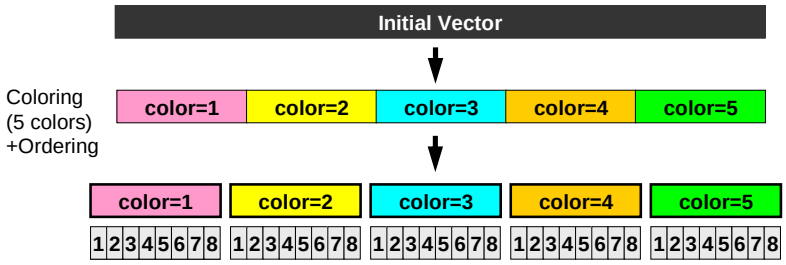
- Introduction
  - 3-min Introduction to Finite Element Method
- Parallel Finite-Element Methods
  - GeoFEM
  - Various Capabilities
- Critical Problems in Parallel FEM's
  - Expectations to ***XcalableMP***

Current Status (1/2)

- Almost everything can be done with MPI-based utilities, tools, libraries, and frameworks (open source, commercial)
  - GeoFEM, HPC-MW, PETSc, AZTEC
  - ZOLTAN (SNL) for AMR/DLB
  - MCT (Model Coupling Toolkit, ANL) etc.
  - Parallel Version of AVS
- It is said that programming with MPI is difficult and complicated for application people…
  - Experiences in education at earth science department of UT.
  - "Data Structure = Abstraction" is important
- Development of everything (tools, libraries, applications) is definitely difficult

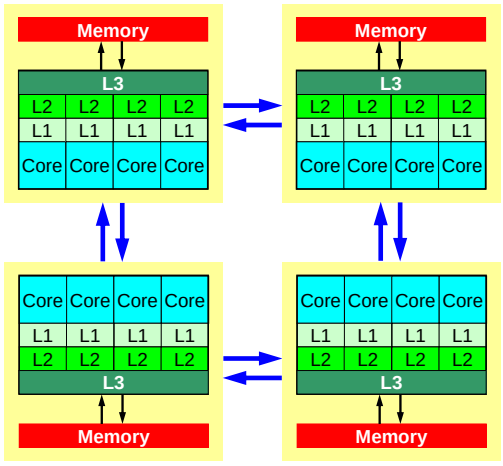
Current Status (2/2)

- It is said that programming with OpenMP is easy and straightforward for application people…
  - Reordering for ILU factorization
  - Data Localization and NUMA control issues for Multi-Socket/Multi-Core Architecture (e.g. T2K Open Supercomputers)



Elements in each color are independent, therefore parallel processing is possible. => divided into OpenMP threads (8 threads in this case)

Because all arrays are numbered according to "color", discontinuous memory access may happen on each thread.



Lessons from GeoFEM

- Tools/Libraries extracted from "real" applications are practically useful.
- Careful design of local data structure for SPMD is important.

- I am not sure whether programming language can be designed in this way or not.
  - but several types of target application should be considered.

“Local View” in XcalableMP

- Users must understand SPMD, local/global data
- Fundamental idea is not so different from MPI version
- Programming is easier
  - !$XMP reflect XXX
  - !$XMP sum XXX

Sparse Matrix-Vector Products

MPI

```
      call SOLVER_SEND_RECV                                             &
     &   ( NP, NEIBPETOT, NEIBPE, STACK_IMPORT, NOD_IMPORT,             &
     &     STACK_EXPORT, NOD_EXPORT, WS, WR, P, SOLVER_COMM, my_rank)

      do j= 1, N
        Q(j)= D(j)*P(j)
        do k= INL(j-1)+1, INL(j)
            i = IAL(k)
          Q(j)= Q(j) + AL(k)*P(i)
        enddo
        do k= INU(j-1)+1, INU(j)
            i = IAU(k)
          Q(j)= Q(j) + AU(k)*P(i)
        enddo
      enddo
```

XcalableMP

```
!$XMP reflect (P)

      do j= 1, N
        Q(j)= D(j)*P(j)
        do k= INL(j-1)+1, INL(j)
            i = IAL(k)
          Q(j)= Q(j) + AL(k)*P(i)
        enddo
        do k= INU(j-1)+1, INU(j)
            i = IAU(k)
          Q(j)= Q(j) + AU(k)*P(i)
        enddo
      enddo
```

SOLVER_SEND_RECV

```
subroutine SOLVER_SEND_RECV                          &
          (N, NEIBPETOT, NEIBPE,                     &
           IMPORT_INDEX, IMPORT_NODE,                &
           EXPORT_INDEX, EXPORT_NODE,                &
           WS, WR, X, SOLVER_COMM, my_rank)

implicit REAL*8 (A-H,O-Z)
include 'mpif.h'
parameter (KREAL= 8)
integer IMPORT_INDEX(0:NEIBPETOT), IMPORT_NODE(N)
integer EXPORT_INDEX(0:NEIBPETOT), EXPORT_NODE(N)
integer SOLVER_COMM, my_rank
integer req1(NEIBPETOT), req2(NEIBPETOT)
integer sta1(MPI_STATUS_SIZE, NEIBPETOT)
integer sta2(MPI_STATUS_SIZE, NEIBPETOT)
real(kind=KREAL)  X(N), NEIBPE(NEIBPETOT), WS(N), WR(N)
```

SEND

```
do neib= 1, NEIBPETOT
  istart= EXPORT_INDEX(neib-1)
  inum  = EXPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    WS(k)= X(EXPORT_NODE(k))
  enddo
  call MPI_ISEND
      (WS(istart+1), inum, MPI_DOUBLE_PRECISION,   &
       NEIBPE(neib), 0, SOLVER_COMM,               &
       req1(neib), ierr)
enddo
```

RECEIVE

```
do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  call MPI_IRECV
      (WR(istart+1), inum, MPI_DOUBLE_PRECISION, &
       NEIBPE(neib), 0, SOLVER_COMM,             &
       req2(neib), ierr)
enddo

call MPI_WAITALL (NEIBPETOT, req2, sta2, ierr)

do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    X(IMPORT_NODE(k))= WR(k)
  enddo
enddo

call MPI_WAITALL (NEIBPETOT, req1, sta1, ierr)

return
end
```


Six reasons why HPF failed

Brad Chamberlain (Cray/Chapel)

- Lack of good performance soon enough, lack of patience from the use community
- Lack of portable performance model, execution model
- Inability to drop the lower levels of parallel computation
- Lack of rich abstraction
  - reasonable support for dense multidimensional arrays
  - but not for sparse data structures
- Lack of general parallel programming models
- Lack of an open source implementation

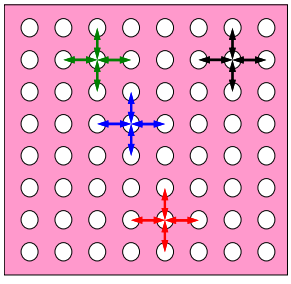
H.Murai (NEC/XcalableMP)

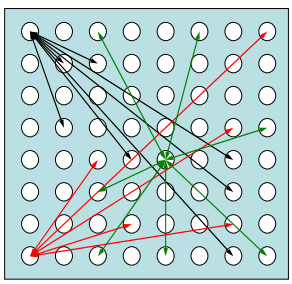
Important Issues for XcalableMP (1/2)

- Examples of REAL Applications
  - XcalableMP should support functions required by real appl's
- Util's/Tools/Libraries/Frameworks for Development of Various Types of Applications
  - Linear Solvers, Visualization, Coupling
  - Meshing, Partitioning, AMR/DLB
- Performance (Parallel, Serial), memory, latency
- Combination with MPI
  - explicit use of MPI
  - utilization a lot of existing MPI-based frameworks, libraries
- Combination with OpenMP
  - XcalableMP + OpenMP
- Multi-Level XcalableMP

Important Issues for XcalableMP (2/2)

- Simple Interface between FORTRAN & C
  - Example of Coupled Simulations
- Global Information
- Education
  - Idea of SPMD
    - especially in "Local View"
  - Development of Applications
    - course for each application
    - text book
    - NOT just for syntax





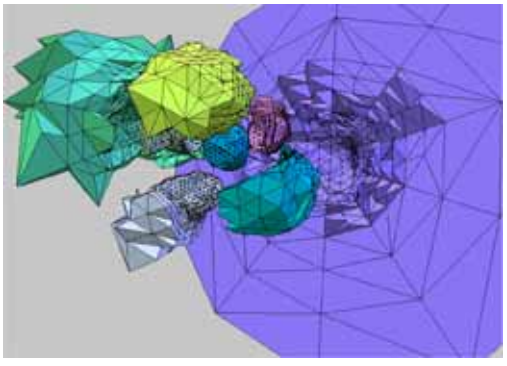
MY Expectations for XcalableMP

- as an Environment for Development of Parallel Tools by NON-Experts
  - Application people can develop tools for parallel computing (e.g. AMR, DLB, coupling etc.) without help of experts using XcalableMP.
  - XcalableMP can provide “global” information under distributed environment
    - by CFA-features, GA-like capabilities
    - NOT optimized, but easy
  - e.g. AMR/DLB for very special types of elements which are not supported by general libraries
    - This type of procedure is used not so frequently during computation (e.g. once for 1,000 time steps), therefore it is not required to be optimized
    - Experts of tool development can develop optimized version later

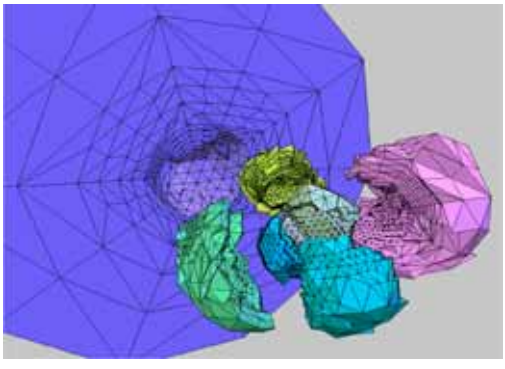
AMR+DLB

8 PE cases with 2-level adapted meshes
Final Mesh : 10,240 nodes, 69,462 tet.

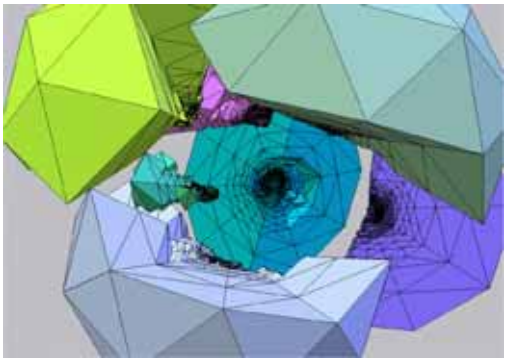
P-JOSTLE



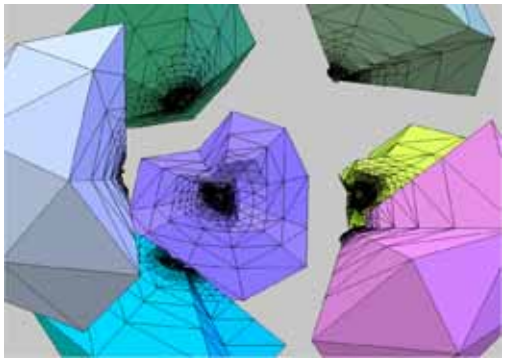
P-METIS k-way



RCB bucket



NO Repartition



[Nakajima, Fingberg and Okuda, 2001]

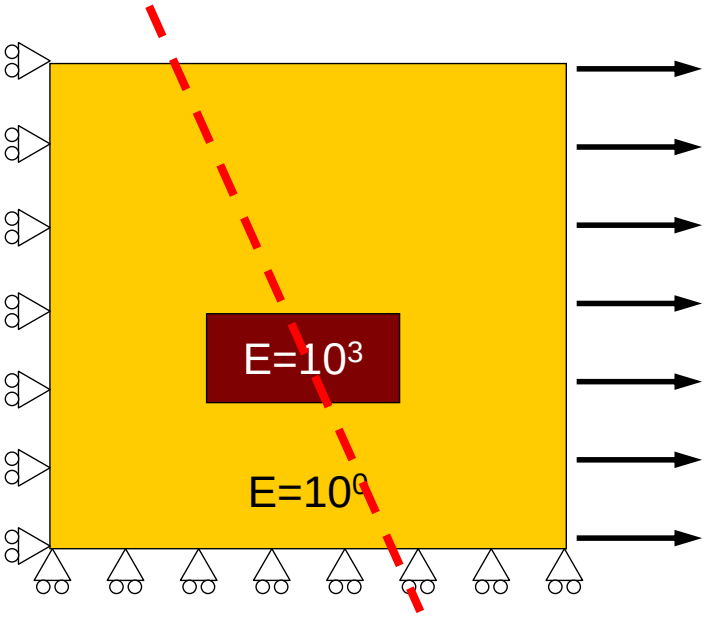
Example of “non-scalable” processes in “NEW Local ID” for DLB

```
do ip= 1, PETOT
  if (ip-1.eq.my_rank) then
    do i= 1, NODEtotLorg
      nn= NODE_ID_NEW(i,2)
      gcount(nn+1)= gcount(nn+1) + 1
      NODE_ID_NEW(i,1)= gcount(nn+1)
    enddo
  endif
  call MPI_BCAST ( gcount, PETOT, MPI_INTEGER, ip-1, &
                   MPI_COMM_WORLD, ierr )
enddo
```


Critical Issues for Parallel FEM (1/2)

- Robust and Scalable Linear Solvers for Real-World Ill-Conditioned Problems

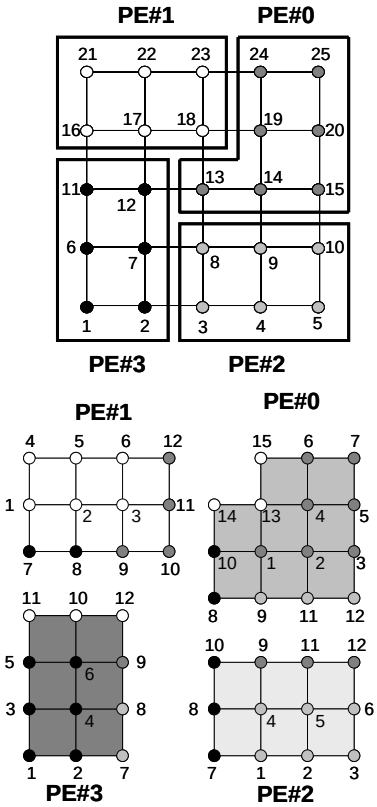
Technical Issues of “Parallel” Preconditioners for Iterative Solvers



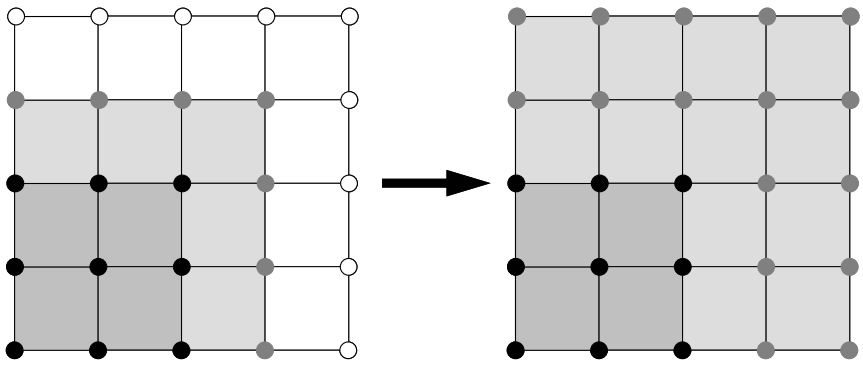
3D Solid Mechanics
E: Young’s Modulus

- If domain boundaries are on “stronger” elements, convergence is very bad.

Extension of Overlapped Zones

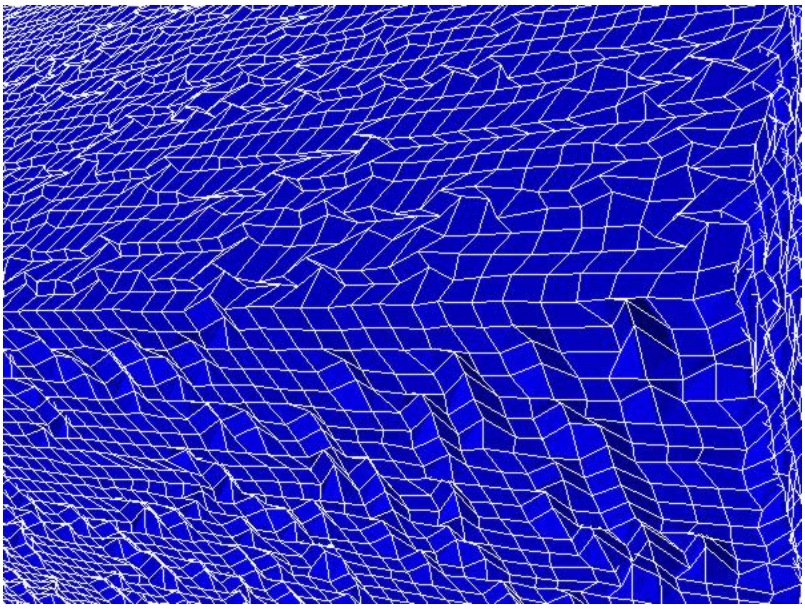


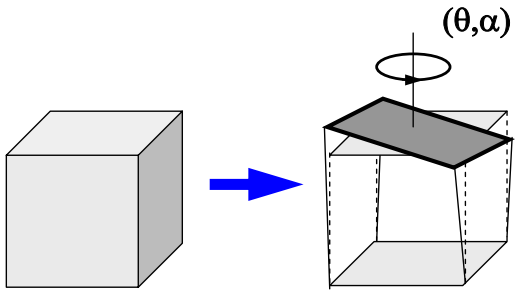
Cost for computation and communication may increase



●:Internal Nodes, ●:External Nodes
■:Overlapped Elements

Heterogeneous Field with Distorted Meshes





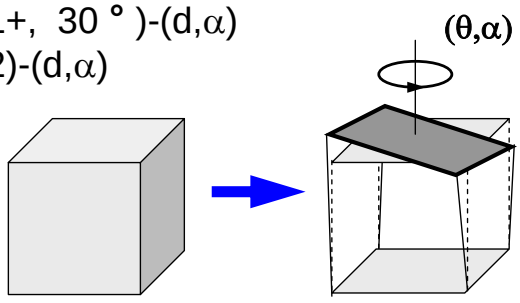
Results: 64 cores Distorted Meshes

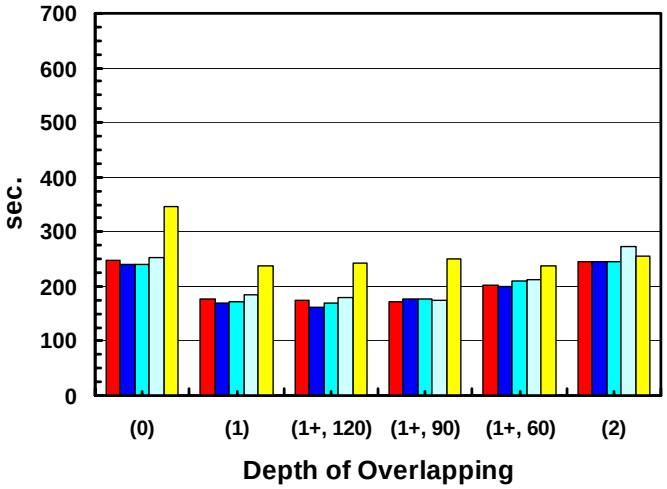
BILU(p,θ)-(d,α)

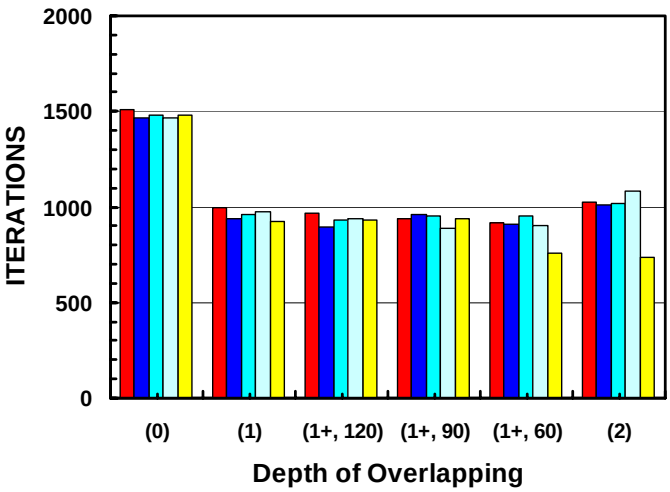
3,090,903 DOF, 64 cores

MAX distortion: **150-deg.**

BILU(1)-(d,α) GPBiCG
BILU(1+,120 °)-(d,α)
BILU(1+, 60 °)-(d,α)
BILU(1+, 30 °)-(d,α)
BILU(2)-(d,α)







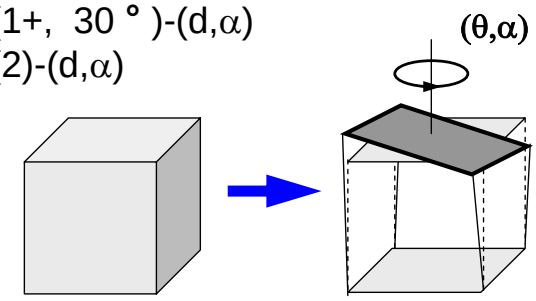
Results: 64 cores Distorted Meshes

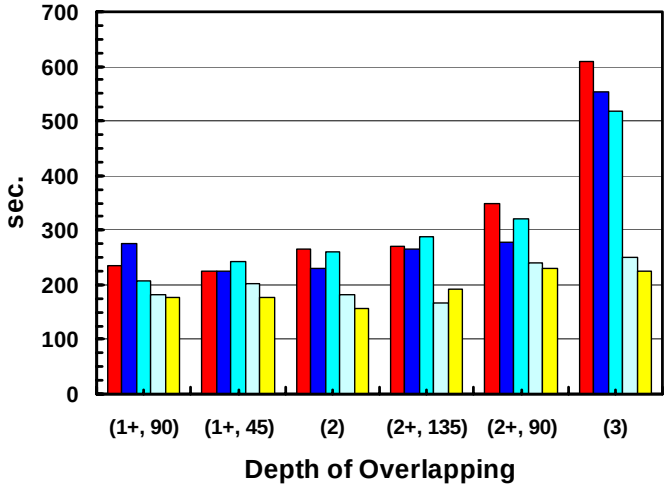
BILU(p,θ)-(d,α)

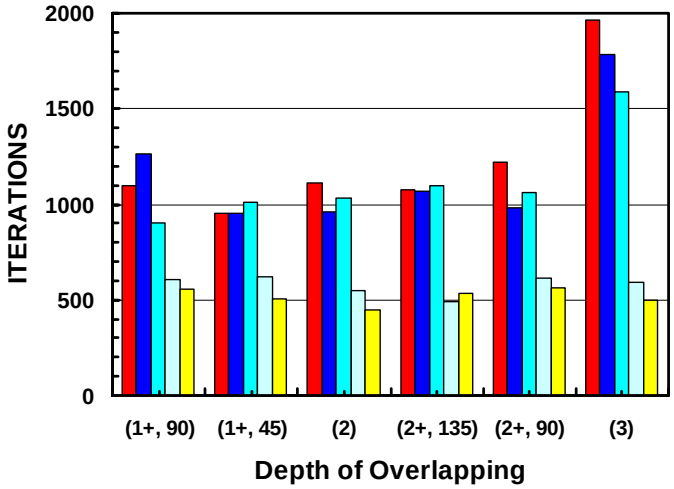
3,090,903 DOF, 64 cores

MAX distortion: **225-deg.**

BILU(1)-(d,α) GPBiCG
BILU(1+,120 °)-(d,α)
BILU(1+, 60 °)-(d,α)
BILU(1+, 30 °)-(d,α)
BILU(2)-(d,α)

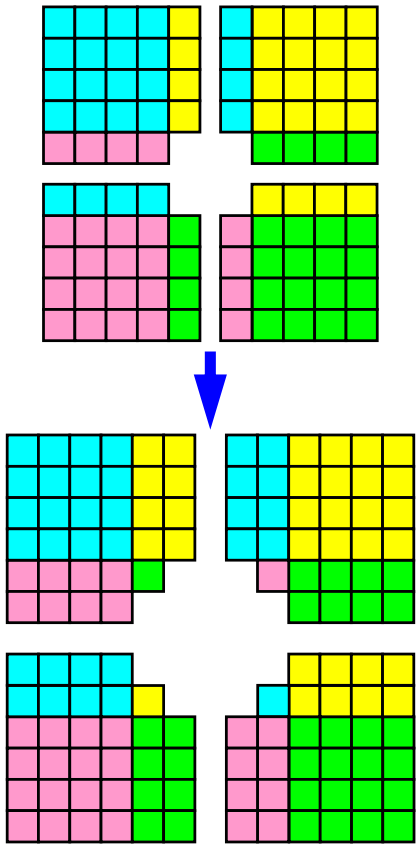






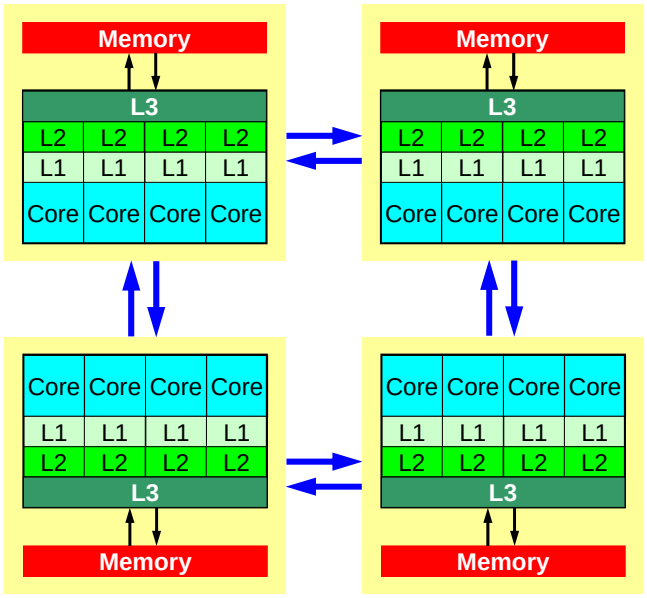
Automatic Tool for Optimum Depth of Overlapping

- Strategy for Optimum Depth
  - interesting research area
- Tools for “Just Moving Data” on Distributed Environment
  - NOT easy
  - Utilization of DLB tool is possible, but difficult for non-experts
  - XcalableMP



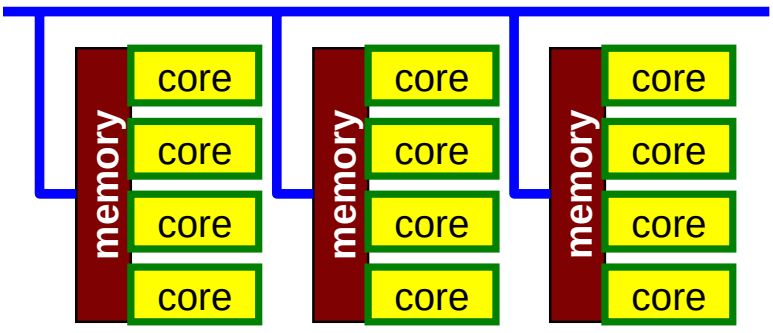
Critical Issues for Parallel FEM (2/2)

- Exascale Systems
  - $O(10^8)$ cores
  - Terrible Communication Overhead by MPI Latency for > 10^8-way MPI's
  - **Expectations for Hybrid**
    - **1/16 MPI processes for T2K/Tokyo**

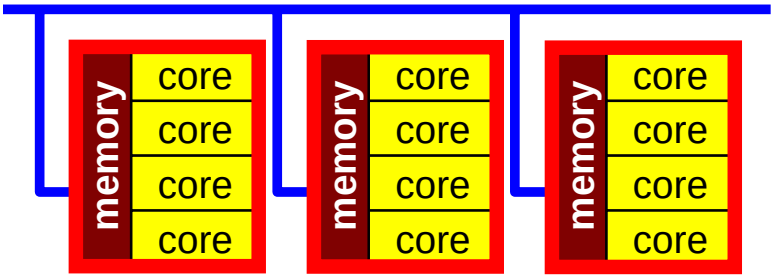


Flat MPI vs. Hybrid

Flat-MPI：Each PE -> Independent



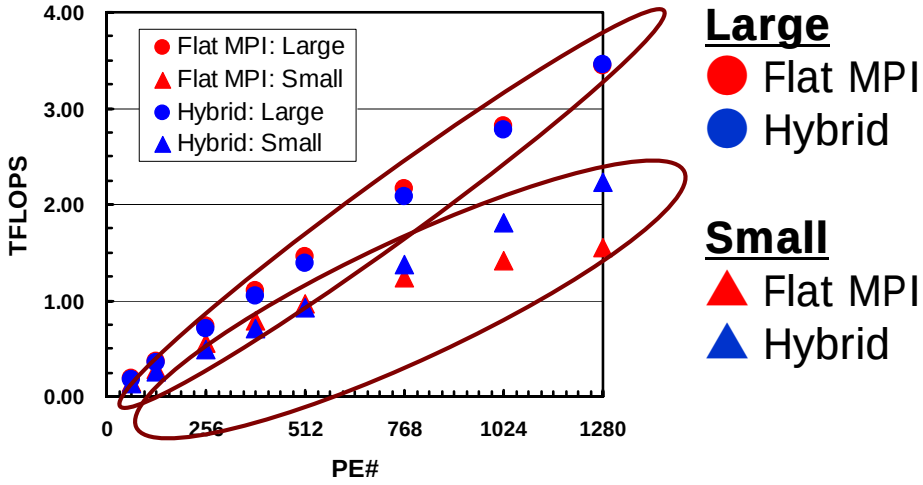
Hybrid：Hierarchal Structure



Weak Scaling Results on ES

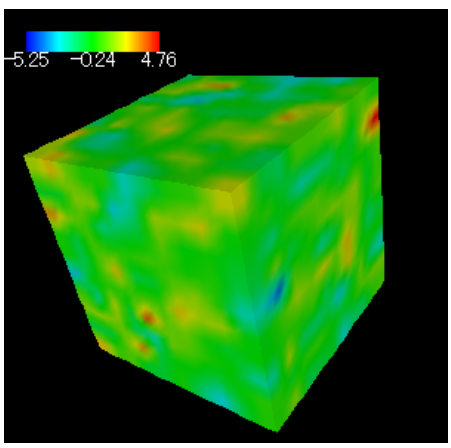
GeoFEM Benchmarks [KN 2003]

- Generally speaking, hybrid is better for large number of nodes
- especially for small problem size per node
  - "less" memory bound

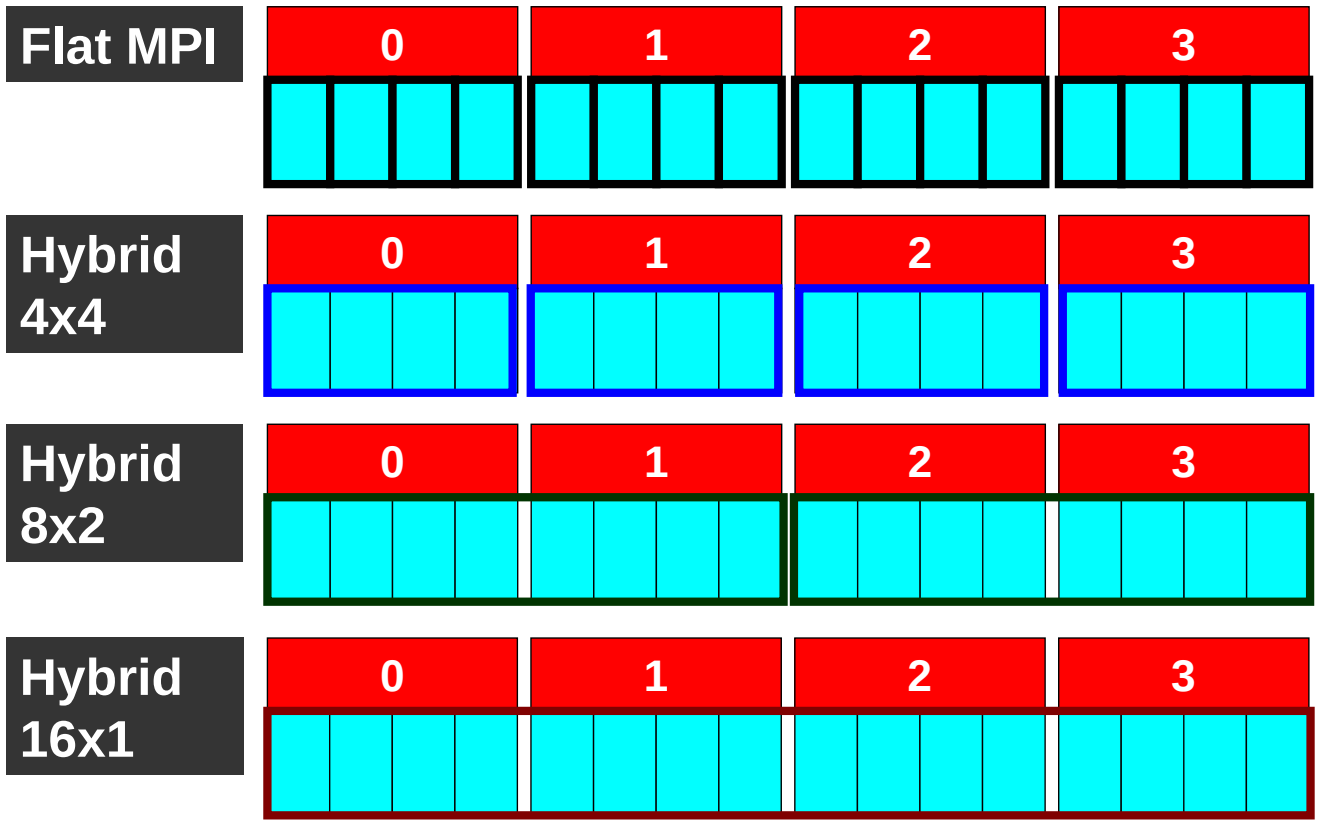


Target Application

- 3D Elastic Problems with Heterogeneous Material Property
  - $E_{max}=10^3$, $E_{min}=10^{-3}$, $\nu=0.25$
    - generated by "sequential Gauss" algorithm for geo-statistics [Deutsch & Journel, 1998]
  - 128^3 tri-linear hexahedral elements, 6,291,456 DOF
    - Strong Scaling
- (SGS+CG) Iterative Solvers
  - Symmetric Gauss-Seidel
  - HID-based domain decomposition
- T2K/Tokyo
  - 512 cores (32 nodes)
- FORTARN90 (Hitachi) + MPI
  - Flat MPI, Hybrid (4x4, 8x2, 16x1)



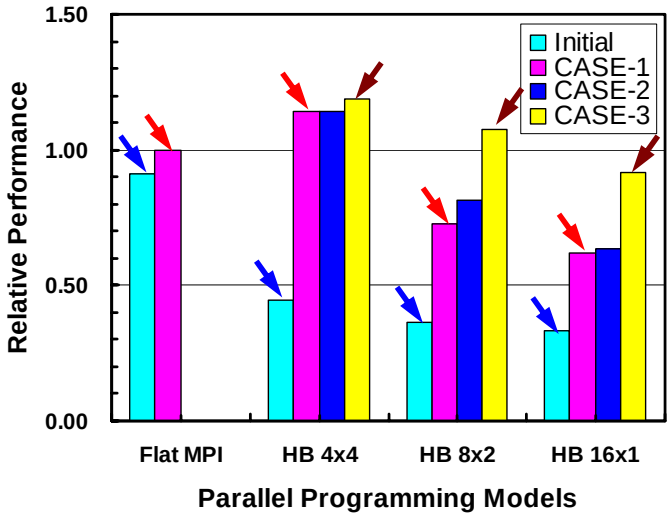
Flat MPI, Hybrid (4x4, 8x2, 16x1)



Improvement: CASE-1 ⇒ CASE-3

Normalized by the Best Performance of Flat MPI

32nodes, 512cores
196,608 DOF/node

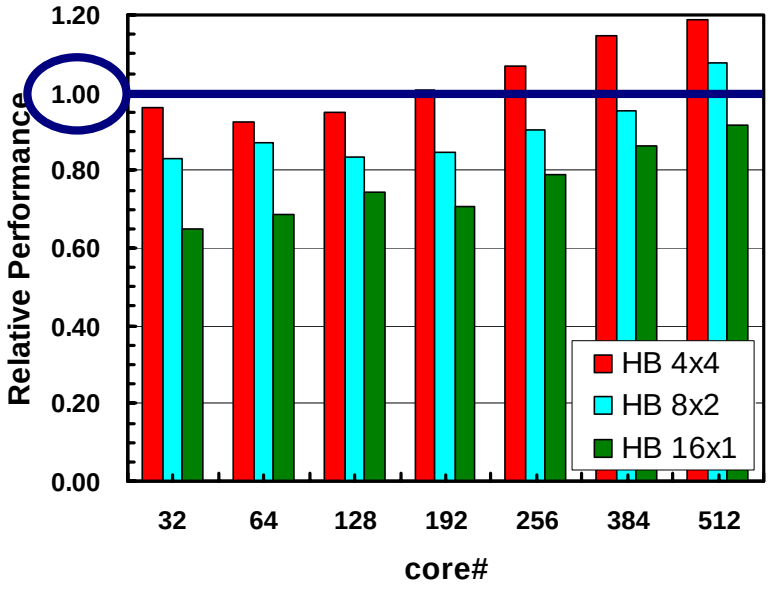


CASE-1: NUMA control
CASE-2: + F.T.
CASE-3: + Further Reordering

Relative Performance for Strong Scaling (Best Cases)

32~512 cores

Normalized by BEST Flat MPI at each core#



Optimum Parallel Programming Model ?

- Flat
  - MPI
  - XcalableMP
- Hybrid
  - MPI (inter-node) + OpenMP (intra-node)
  - MPI + MPI
  - XcalableMP + OpenMP
  - XcalableMP + XcalableMP
  - XcalableMP + MPI