並列有限要素法による
三次元定常熱伝導解析プログラム
並列可視化

中島 研吾
東京大学情報基盤センター

自動チューニング機構を有するアプリケーション開発・実行環境 ppOpen-HPC

中島研吾
東京大学情報基盤センター

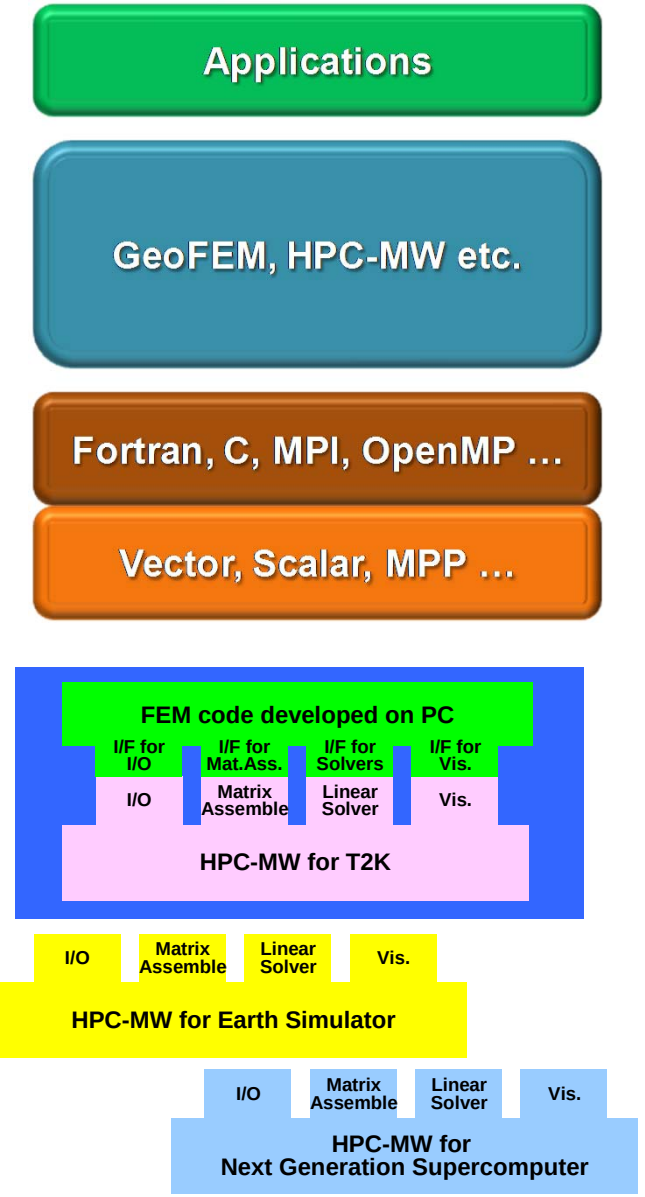
佐藤正樹（東大・大気海洋研究所），奥田洋司（東大・新領域創成科学研究科），
古村孝志（東大・情報学環/地震研），岩下武史（京大・学術情報メディアセンター），
阪口秀（海洋研究開発機構）

背　景(1/2)

- 大規模化，複雑化，多様化するハイエンド計算機環境の能力を充分に引き出し，効率的なアプリケーションプログラムを開発することは困難
- 有限要素法等の科学技術計算手法：
  - プリ・ポスト処理，行列生成，線形方程式求解等の一連の共通プロセスから構成される。
  - これら共通プロセスを抽出し，ハードウェアに応じた最適化を施したライブラリとして整備することで，アプリケーション開発者から共通プロセスに関わるプログラミング作業，並列化も含むチューニング作業を隠蔽できる。
  - アプリケーションMW，HPC-MW，フレームワーク

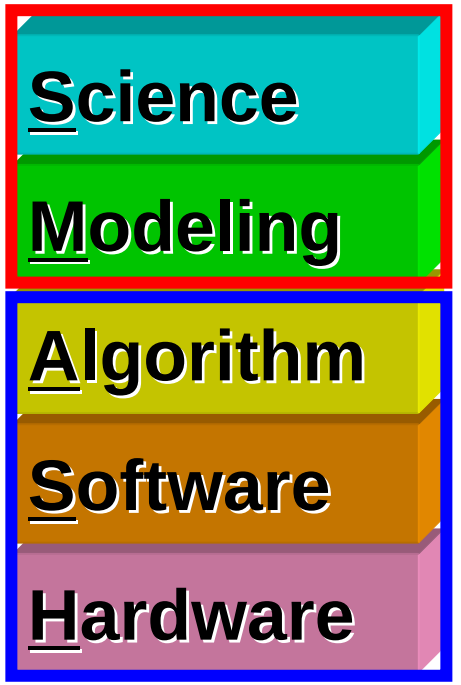
背　景(2/2)

- A.D.2000年前後
  - GeoFEM，HPC-MW
  - 地球シミュレータ，Flat MPI，FEM
- 現在：より多様，複雑な環境
  - マルチコア，GPU
  - ハイブリッド並列
    - MPIまでは何とかたどり着いたが・・・
    - 「京」でも重要
  - CUDA，OpenCL，OpenACC
  - ポストペタスケールからエクサスケールへ
    - より一層の複雑化



HPCミドルウェア：何がうれしいか

- アプリケーション開発者のチューニング（並列，単体）からの解放
  - **SM**ASHの探求に専念
    - 一生SMASHと付き合うのはきつい
  - SM**ASH**をカバー
- コーディングの量が減る
- 教育にも適している
- 問題点
  - ハードウェア，環境が変わるたびに最適化が必要となる



東大情報基盤センターのスパコン

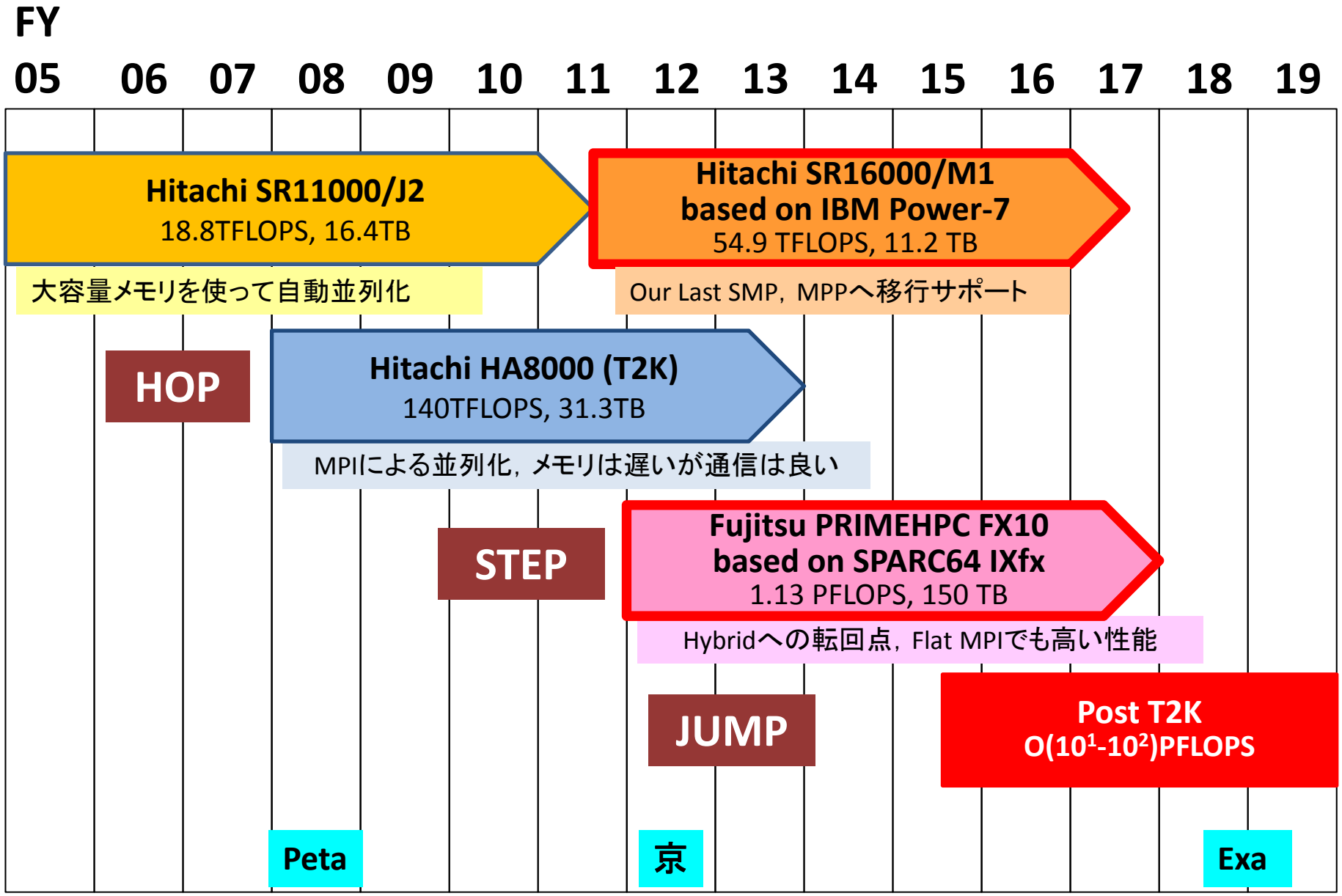
1システム～6年，3年周期でリプレース

Oakleaf-FX (Fujitsu PRIMEHPC FX10)		T2K-Todai（2014年3月退役）（Hitachi HA8000-tc/RS425 ）		Yayoi (Hitachi SR16000/M1)	
Total Peak performance	: 1.13 PFLOPS	Total Peak performance	: 140 TFLOPS	Total Peak performance	: 54.9 TFLOPS
Total number of nodes	: 4800	Total number of nodes	: 952	Total number of nodes	: 56
Total memory	: 150 TB	Total memory	: 32000 GB	Total memory	: 11200 GB
Peak performance / node	: 236.5 GFLOPS	Peak performance / node	: 147.2 GFLOPS	Peak performance / node	: 980.48 GFLOPS
Main memory per node	: 32 GB	Main memory per node	: 32 GB, 128 GB	Main memory per node	: 200 GB
Disk capacity	: 1.1 PB + 2.1 PB	Disk capacity	: 1 PB	Disk capacity	: 556 TB
SPARC64 Ixfx 1.84GHz		AMD Quad Core Opteron 2.3GHz		IBM POWER 7 3.83GHz	

“Oakbridge-fx” with 576 nodes installed in April 2014 (separated) (136TF)

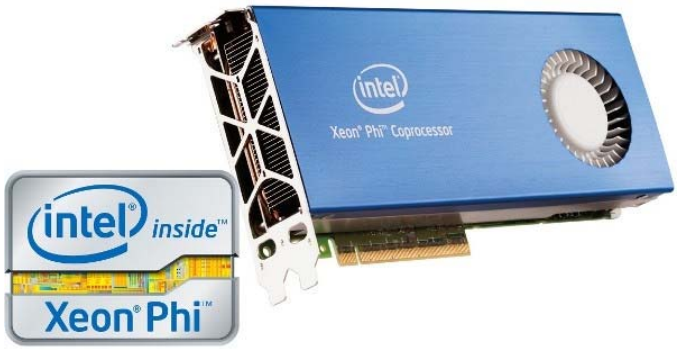
Total Users > 2,000

東大情報基盤センターのスパコン



Post T2K System

- 20-30 PFLOPS, FY.2015
- Many-core based (e.g. (only) Intel MIC/Xeon Phi)
- Joint Center for Advanced High Performance Computing (最先端共同HPC基盤施設，JCAHPC, http://jcahpc.jp/)
  - 筑波大学計算科学研究センター，東京大学情報基盤センター
- Programming is still difficult, although Intel compiler works.
  - (MPI + OpenMP)
  - Tuning for performance (e.g. prefetching) is essential
  - Some framework for helping users needed

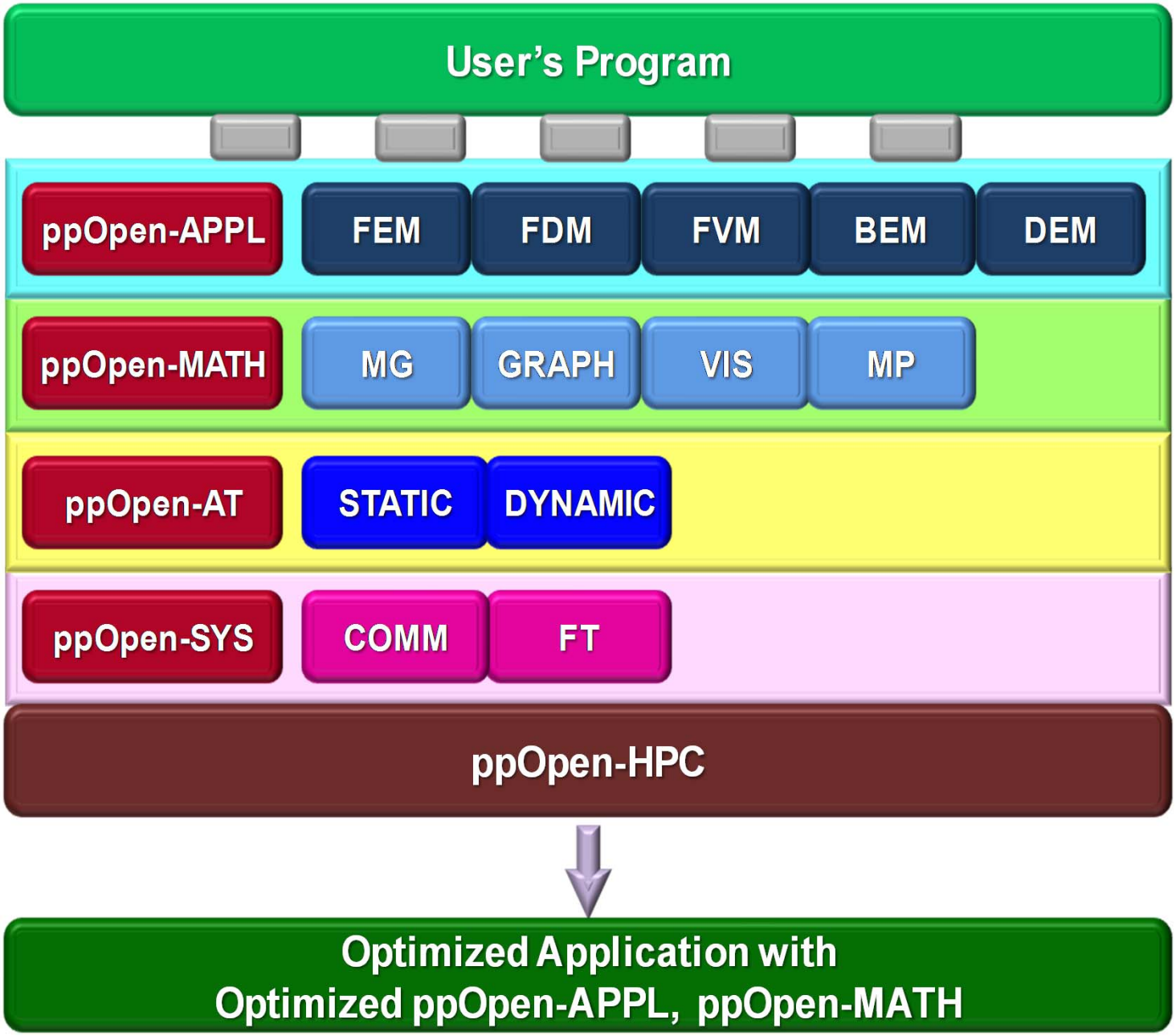


ppOpen-HPC

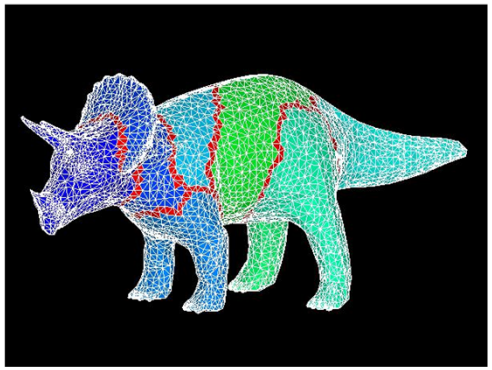
- 東京大学情報基盤センターでは，メニィコアに基づく計算ノードを有するポストペタスケールシステムの処理能力を充分に引き出す科学技術アプリケーションの効率的な開発，安定な実行に資する「自動チューニング機構を有するアプリケーション開発・実行環境：ppOpen-HPC」を開発中。
  - 科学技術振興機構戦略的創造研究推進事業（CREST）研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出（Post-Peta CREST）」（2011～2015年度）（領域統括：米澤明憲教授（理化学研究所計算科学研究機構））
  - PI：中島研吾（東京大学情報基盤センター）
  - 東大（情報基盤センター，大気海洋研究所，地震研究所，大学院新領域創成科学研究科），京都大学術情報メディアセンター，北海道大学情報基盤センター，海洋研究開発機構
  - 様々な分野の専門家によるCo-Design

概 要(1/3)

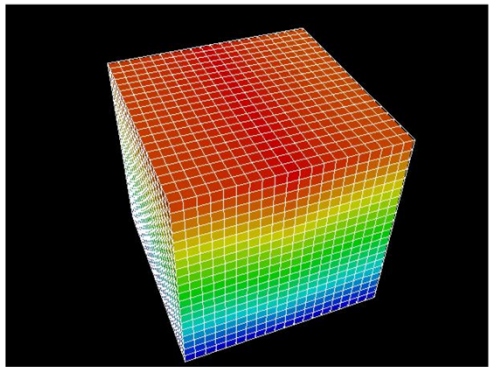
- メニーコアクラスタによるポストペタスケールシステム上での科学技術アプリケーションの効率的開発，安定な実行に資するppOpen-HPCの研究開発を計算科学，計算機科学，数理科学各分野の緊密な協力のもとに実施している。
  - 6 Issues in Post-Peta/Exascale Computingを考慮
  - “pp”: Post Peta
- 東大情報基盤センターに平成27年度導入予定のO(10)PFLOPS級システム(ポストT2K，Intel MIC/Xeon-Phiベース)をターゲット:
  - スパコンユーザーの円滑な移行支援
- 大規模シミュレーションに適した5種の離散化手法に限定し，各手法の特性に基づいたアプリケーション開発用ライブラリ群，耐故障機能を含む実行環境を実現する。
  - ppOpen-APPL:各手法に対応した並列プログラム開発のためのライブラリ群
  - ppOpen-MATH:各離散化手法に共通の数値演算ライブラリ群
  - ppOpen-AT:科学技術計算のための自動チューニング(AT)機構
  - ppOpen-SYS:ノード間通信，耐故障機能に関連するライブラリ群


User's Program
ppOpen-APPL
FEM
FDM
FVM
BEM
DEM
ppOpen-MATH
MG
GRAPH
VIS
MP
ppOpen-AT
STATIC
DYNAMIC
ppOpen-SYS
COMM
FT
ppOpen-HPC
Optimized Application with
Optimized ppOpen-APPL, ppOpen-MATH

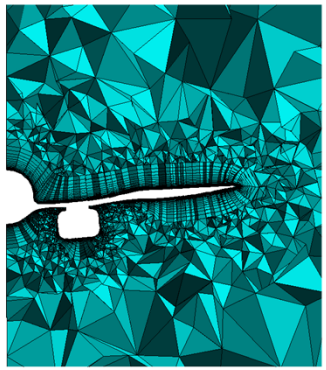
対象とする離散化手法
局所的，隣接通信中心，疎行列



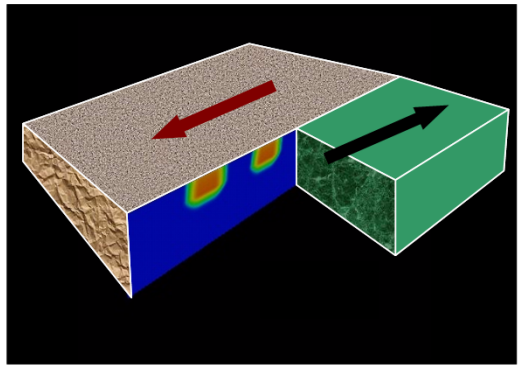
有限要素法
Finite Element Method
FEM



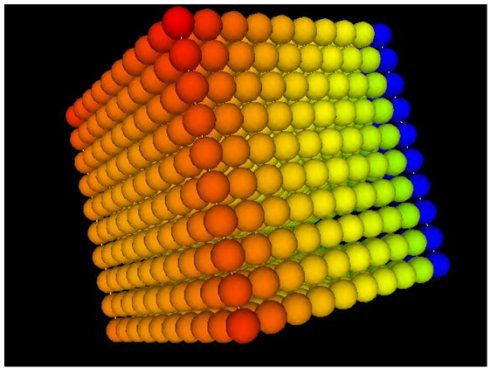
差分法
Finite Difference Method
FDM



有限体積法
Finite Volume Method
FVM



境界要素法
Boundary Element Method
BEM



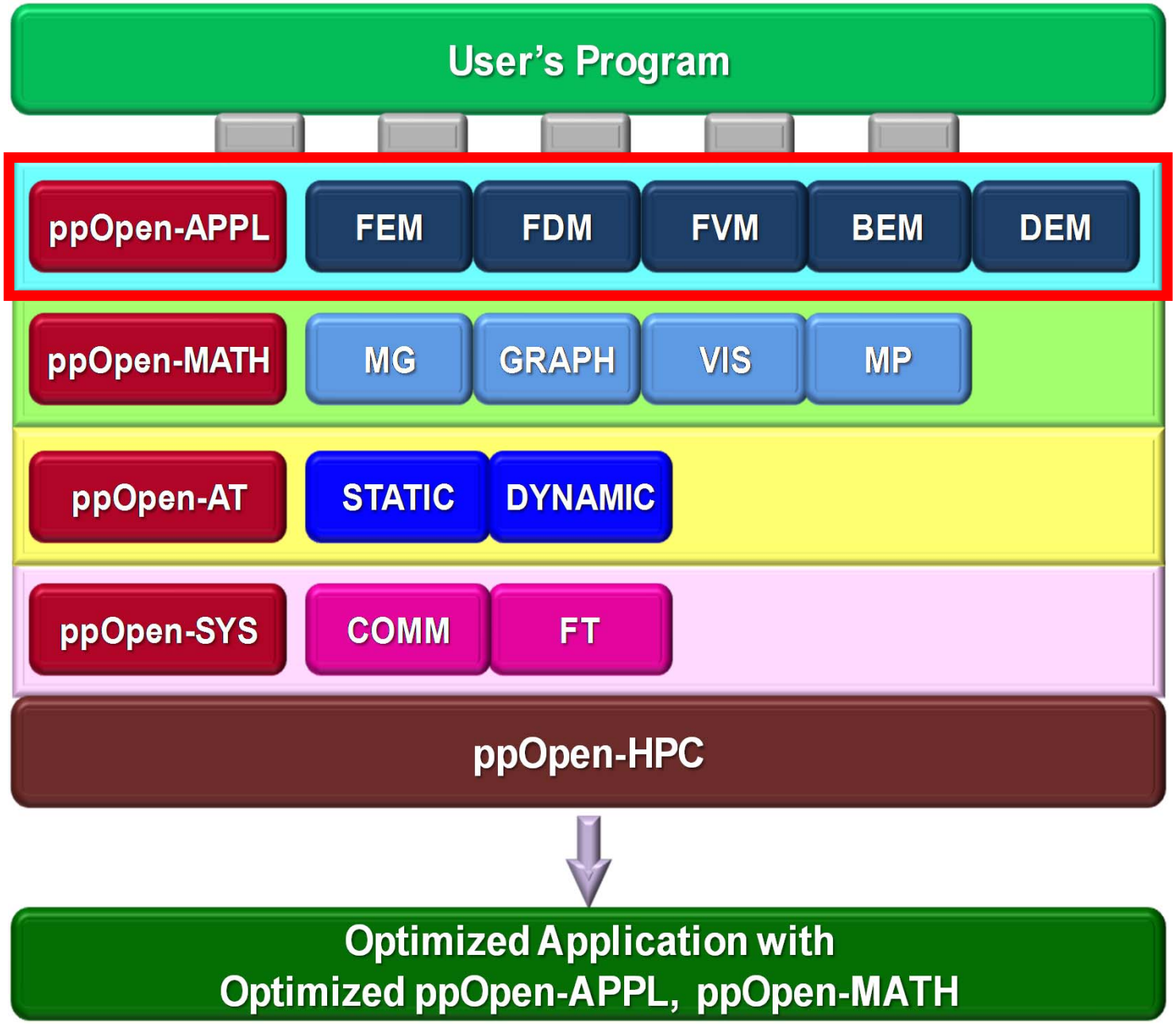
個別要素法
Discrete Element Method
DEM

概 要(2/3)

- 先行研究において各メンバーが開発した大規模アプリケーションに基づきppOpen-APPLの各機能を開発，実装
  - 各離散化手法の特性に基づき開発・最適化
    - 共通データ入出力インタフェース，領域間通信，係数マトリクス生成
    - 離散化手法の特性を考慮した前処理付き反復法
    - 適応格子，動的負荷分散
  - 実際に動いているアプリケーションから機能を切り出す
  - 各メンバー開発による既存ソフトウェア資産の効率的利用
    - GeoFEM，HEC-MW，HPC-MW，DEMIGLACE，ABCLibScript
- ppOpen-ATはppOpen-APPLの原型コードを対象として研究開発を実施し，その知見を各ppOpen-APPLの開発，最適化に適用
  - 自動チューニング技術により，様々な環境下における最適化ライブラリ・アプリケーション自動生成を目指す

概 要（3/3）

- 平成24年11月にマルチコアクラスタ向けに各グループの開発したppOpen-APPL，ppOpen-AT，ppOpen-MATHの各機能を公開（Ver.0.1.0）
  - http://ppopenhpc.cc.u-tokyo.ac.jp/
  - 平成25年11月にVer.0.2.0公開
- 現在は各機能の最適化，機能追加，ppOpen-APPLによるアプリケーション開発とともに，Intel Xeon/Phi等メニーコア向けバージョンを開発中


User's Program
ppOpen-APPL
FEM
FDM
FVM
BEM
DEM
ppOpen-MATH
MG
GRAPH
VIS
MP
ppOpen-AT
STATIC
DYNAMIC
ppOpen-SYS
COMM
FT
ppOpen-HPC
Optimized Application with
Optimized ppOpen-APPL, ppOpen-MATH

ppOpen-APPL

- A set of libraries corresponding to each of the five methods noted above (FEM, FDM, FVM, BEM, DEM), providing:
  - I/O
    - netCDF-based Interface
  - Domain-to-Domain Communications
  - Optimized Linear Solvers (Preconditioned Iterative Solvers)
    - Optimized for each discretization method
  - H-Matrix Solvers in ppOpen-APPL/BEM
  - Matrix Assembling
  - AMR and Dynamic Load Balancing
- Most of components are extracted from existing codes developed by members

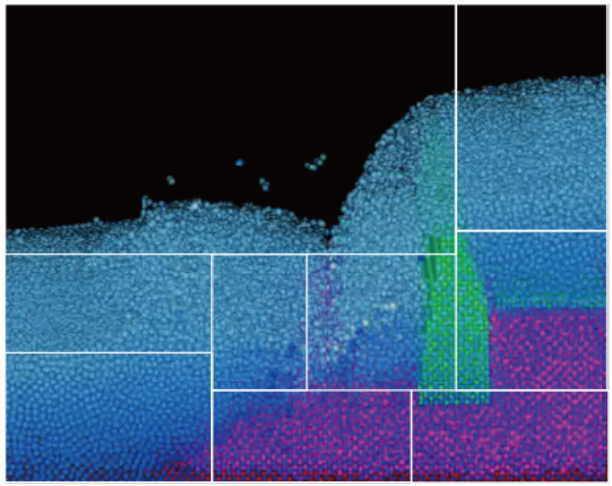
FEM Code on ppOpen-HPC

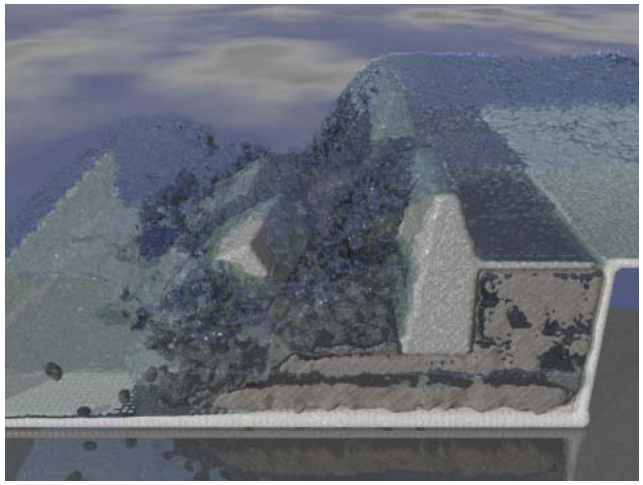
Optimization/parallelization could be hidden from application developers

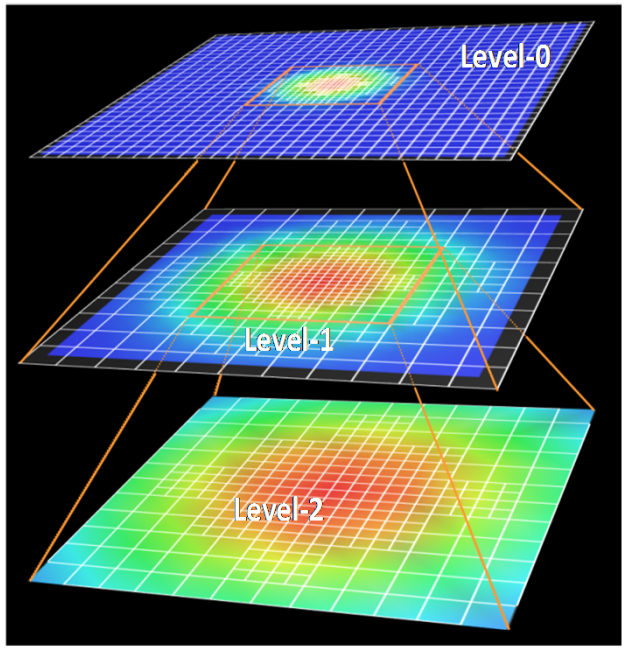
```
Program My_pFEM
use ppOpenFEM_util
use ppOpenFEM_solver

call ppOpenFEM_init
call ppOpenFEM_cntl
call ppOpenFEM_mesh
call ppOpenFEM_mat_init

do
  call Users_FEM_mat_ass
  call Users_FEM_mat_bc
  call ppOpenFEM_solve
  call ppOPenFEM_vis
  Time= Time + DT
enddo

call ppOpenFEM_finalize
stop
end
```

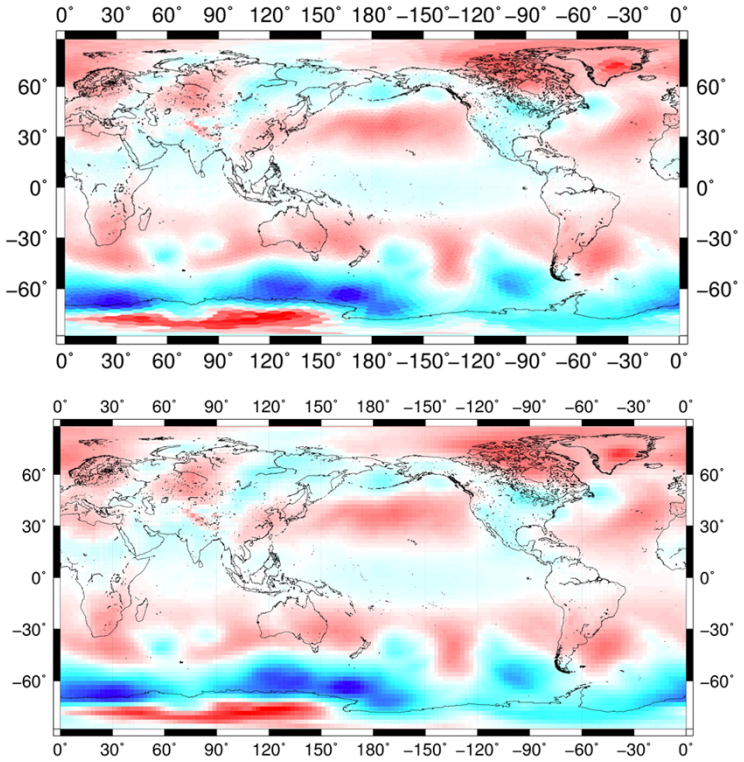
Target Applications

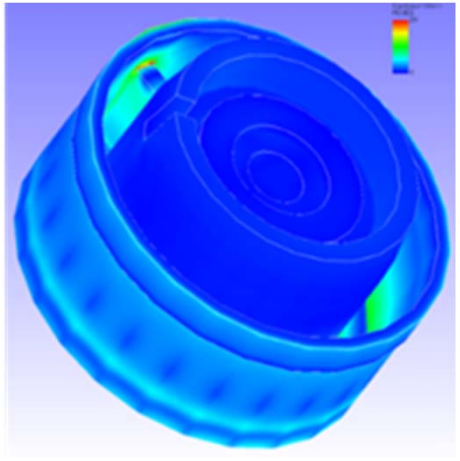
- Our goal is not development of applications, but we need some target appl. for evaluation of ppOpen-HPC.
- ppOpen-APPL/FEM
  - Incompressible Navier-Stokes
  - Heat Transfer, Solid Mechanics (Static, Dynamic)
- ppOpen-APPL/FDM
  - Incompressible Navier-Stokes
  - Transient Heat Transfer, Solid Mechanics (Dynamic)
- ppOpen-APPL/FVM
  - Compressible Navier-Stokes, Heat Transfer
- ppOpen-APPL/BEM
  - Electromagnetics, Solid Mechanics (Quasi Static) (Earthquake Generation Cycle)
- ppOpen-APPL/DEM
  - Incompressible Navier-Stokes, Solid Mechanics (Dynamic)

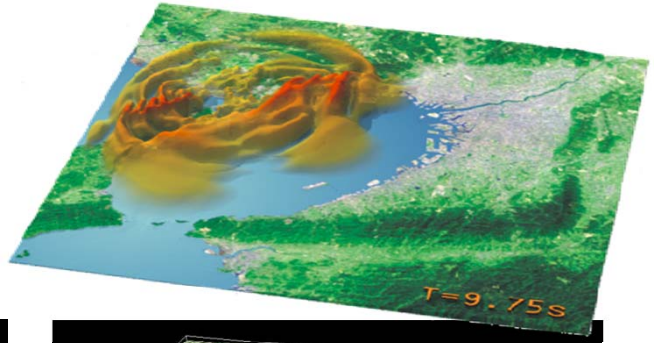


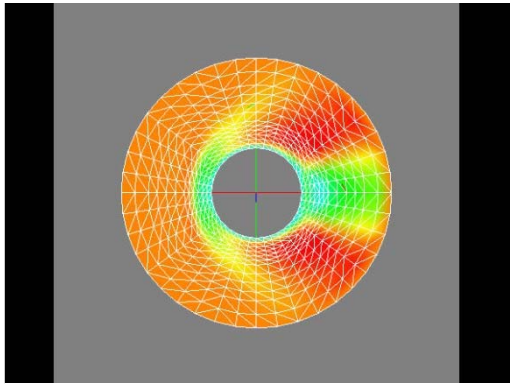












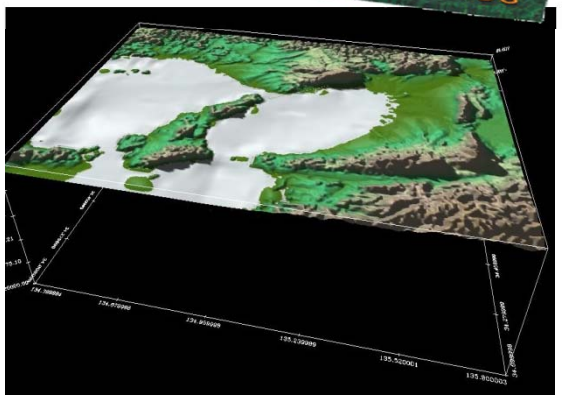
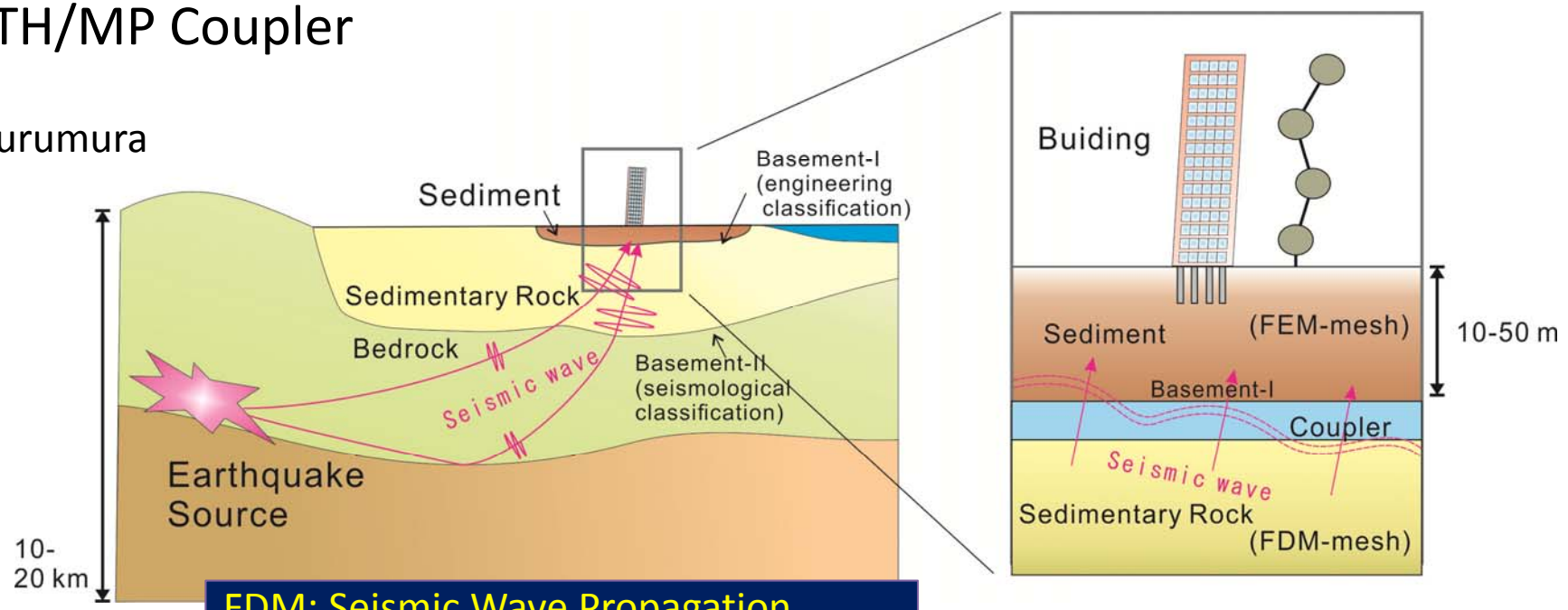


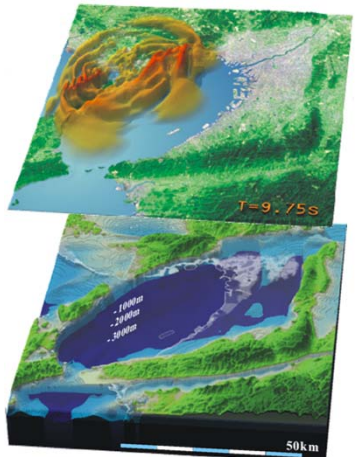
図 1　NICAM と IO モジュールの海面気圧

Large-Scale Coupled Simulations in FY.2013

Challenge (FY2013) : A test of a coupling simulation of FDM (regular grid) and FEM (unconstructed grid) using newly developed ppOpen-MATH/MP Coupler

c/o T.Furumura





FDM: Seismic Wave Propagation
Model size: 80x80x400 km
Time : 240 s
Resolution (space) : 0.1 km (regular)
Resolution (time) : 5 ms
(effective freq.<1 Hz)

FEM: Building Response
Model size : 400x400x200 m
Time : 60 s
Resolution (space) : 1 m
Resolution (time) : 1 ms

ppOpen-MATH/MP: Space-temporal interpolation, Mapping between FDM and FEM mesh, etc.

Schedule of Public Release

(with English Documents, MIT License)

We are now focusing on MIC/Xeon Phi

- 4Q 2012 (Ver.0.1.0)
  - ppOpen-HPC for Multicore Cluster (Cray, K etc.)
  - Preliminary version of ppOpen-AT/STATIC
- 4Q 2013 (Ver.0.2.0)
  - ppOpen-HPC for Multicore Cluster & Xeon Phi (& GPU)
  - available in SC'13
- 4Q 2014
  - Prototype of ppOpen-HPC for Post-Peta Scale System
- 4Q 2015
  - Final version of ppOpen-HPC for Post-Peta Scale System
  - Further optimization on the target system

ppOpen-HPC Ver.0.1.0

http://ppopenhpc.cc.u-tokyo.ac.jp/

- Released at SC12 (or can be downloaded)
- Multicore cluster version (Flat MPI, OpenMP/MPI Hybrid) with documents in English
- Collaborations with scientists

Component	Archive	Flat MPI	OpenMP/MPI	C	F
ppOpen-APPL/FDM	ppohFDM_0.1.0	○			○
ppOpen-APPL/FVM	ppohFVM_0.1.0	○	○		○
ppOpen-APPL/FEM	ppohFEM_0.1.0	○	○	○	○
ppOpen-APPL/BEM	ppohBEM_0.1.0	○	○		○
ppOpen-APPL/DEM	ppohDEM_0.1.0	○	○		○
ppOpen-MATH/VIS	ppohVIS_FDM3D_0.1.0	○		○	○
ppOpen-AT/STATIC	ppohAT_0.1.0	-	-	○	○

What is new in Ver.0.2.0 ?

http://ppopenhpc.cc.u-tokyo.ac.jp/

- Available in SC13 (or can be downloaded)

Component	New Development
ppOpen-APPL/FDM	• OpenMP/MPI Hybrid Parallel Programming Model • Intel Xeon/Phi Version • Interface for ppOpen-MATH/VIS-FDM3D
ppOpen-APPL/FVM	• Optimized Communication
ppOpen-APPL/FEM	• Sample Implementations for Dynamic Solid Mechanics • API for Linear Solver in Fortran
ppOpen-MATH/MP-PP	• Tool for Generation of Remapping Table in ppOpen-MATH/MP
ppOpen-MATH/VIS	• Optimized ppOpen-MATH/VIS-FDM3D
ppOpen-AT/STATIC	• Sequence of Statements, Loop Splitting (Optimized) • ppOpen-APPL/FVM • ppOpen-APPL/FDM・BEM

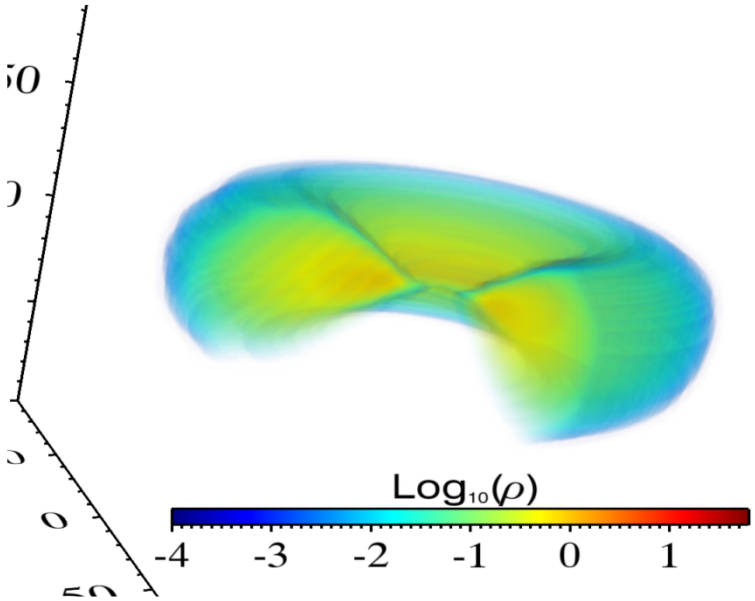
普及活動(1/2)

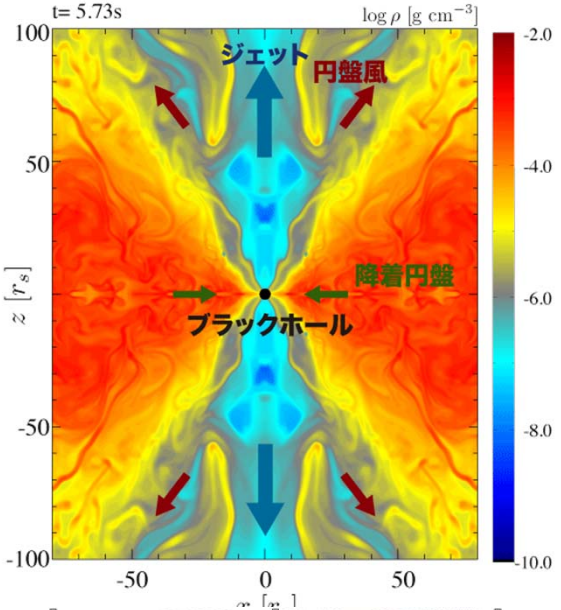
- ppOpen-AT関連共同研究
  - 工学院大学　田中研究室
    - 田中研究室開発のAT方式(d-spline方式)の適用対象としてppOpen-ATのAT機能を拡張
  - 東京大学　須田研究室
    - 電力最適化のため，須田研究室で開発中のAT方式と電力測定の共通APIを利用し，ppOpen-ATを用いた電力最適化方式を提案
- JHPCN共同研究課題
  - 高精度行列-行列積アルゴリズムにおける並列化手法の開発（東大，早稲田大）（H24年度）（研究としては継続）
    - 高精度行列-行列積演算における行列-行列積の実装方式選択に利用
  - 粉体解析アルゴリズムの並列化に関する研究（東大，法政大）（H25年度）
    - 粉体シミュレーションのための高速化手法で現れる性能パラメタのATで利用を検討

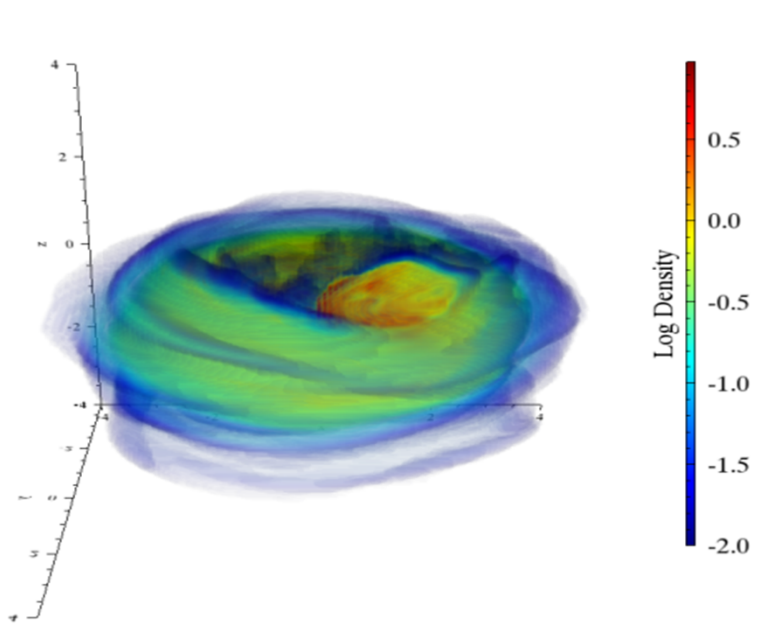
普及活動(2/2)

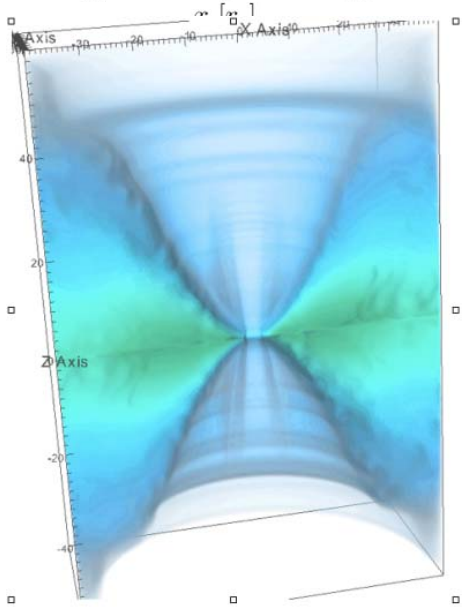
- JHPCN共同研究課題(続き)
  - 巨大地震発生サイクルシミュレーションの高度化(京大，東大他)(H24･25年度)
    - Hマトリクス，領域細分化
  - ポストペタスケールシステムを目指した二酸化炭素地中貯留シミュレーション技術の研究開発(大成建設，東大)(H25年度)
    - 疎行列ソルバー，並列可視化
  - 太陽磁気活動の大規模シミュレーション」(東大(地球惑星，情報基盤センター))(H25年度)
    - 疎行列ソルバー，並列可視化
- 講習会，講義
  - ppOpen-HPCの講習会を2014年3月から実施
  - 講義，講習会(並列有限要素法)でppOpen-MATH/VISを使用して可視化を実施する予定

3D MHD Simulations of Black Hole



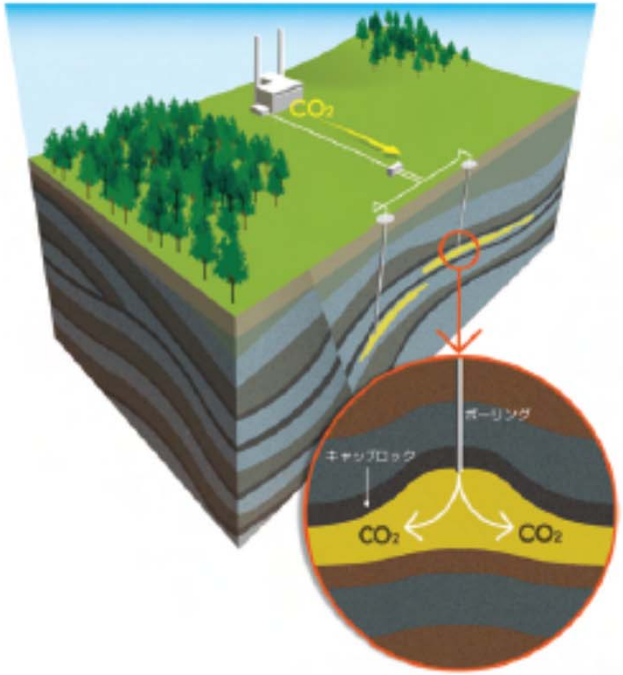






[Prof. Ryoji Matsumoto, Chiba U.]

CO_2地下貯留シミュレーション



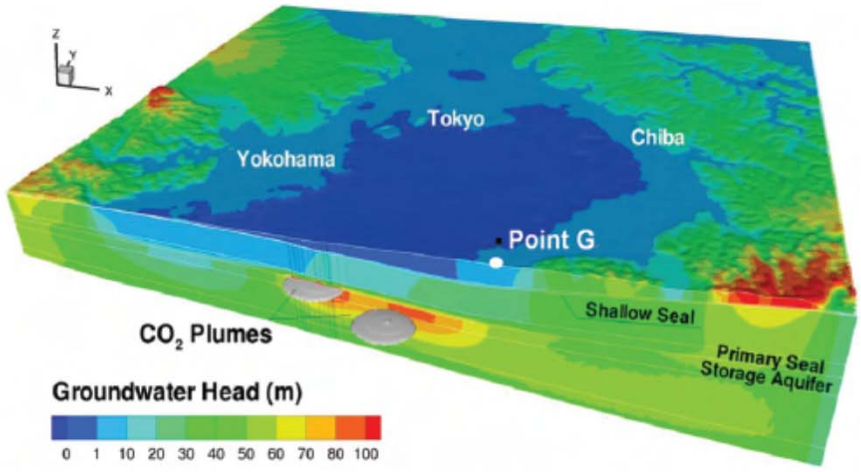
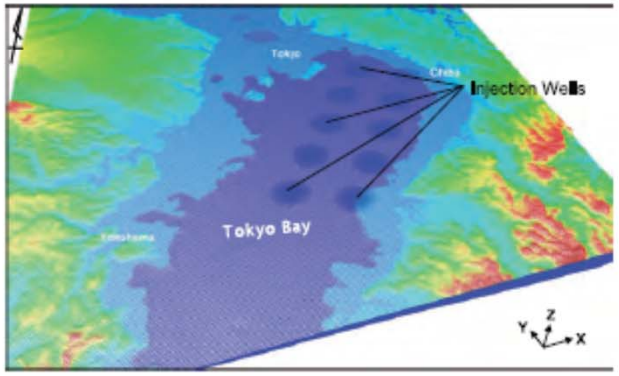
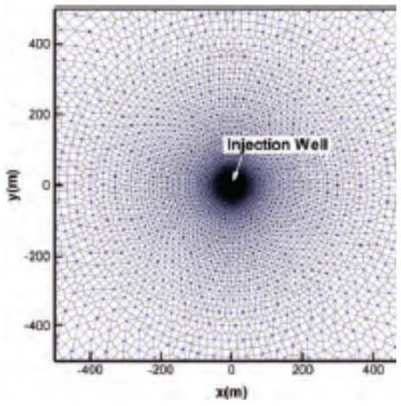
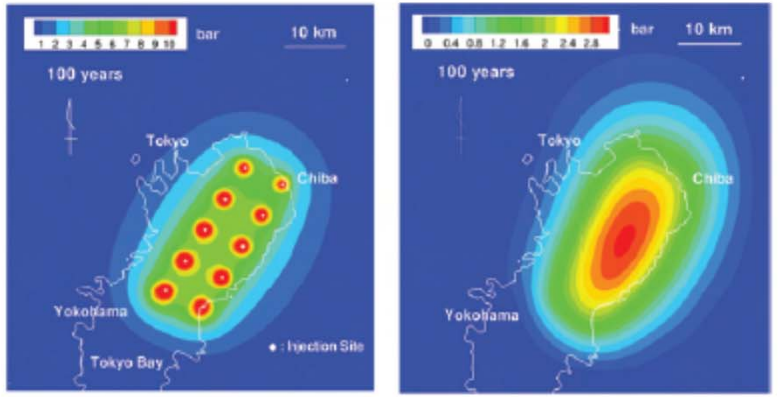


図-4　CO_2圧入後の地下水圧（全水頭換算）の分布（100年後）







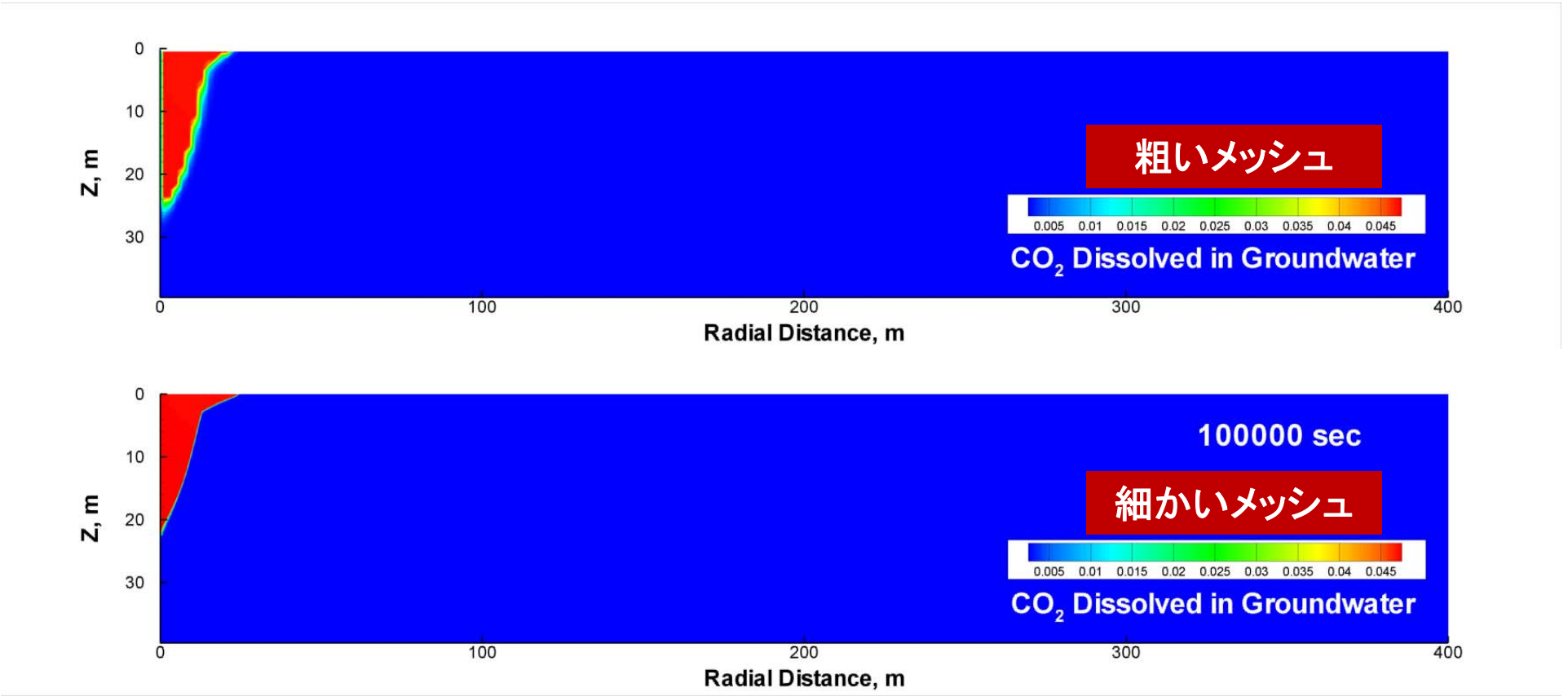
(a) 深部遮蔽層下面　　(b) 浅部遮蔽層下面

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）

〔画像提供：山本肇博士（大成建設）〕

CO_2が地下水に溶けていく様子

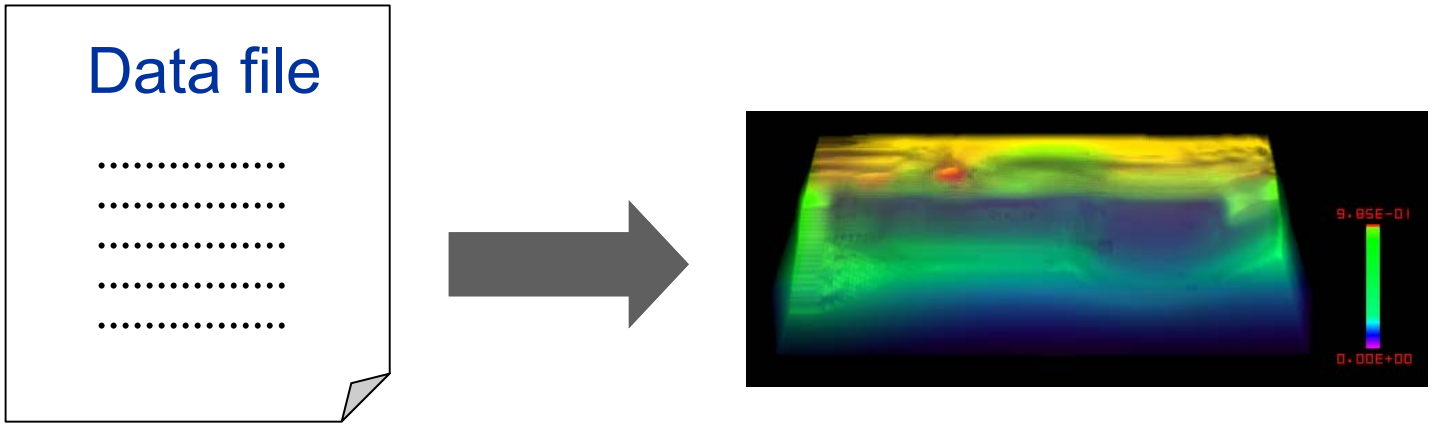
正確な予測のためには細かいメッシュが必要⇒大規模な計算モデル，連立一次方程式



〔画像提供：山本肇博士（大成建設）〕

可視化の意義

- シミュレーションや計測から得られた大規模数値データを視覚表現に変換し対象の直感的理解・効果的解析を支援
  - Controllable pictures are worth more than a thousand of words!



Seeing is Believing

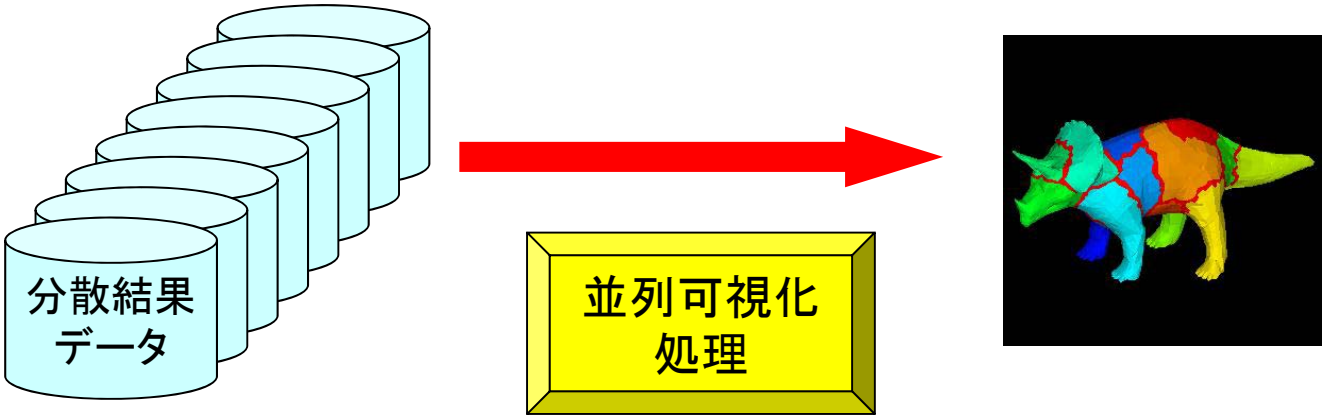
- 人間にとって画像や映像は，さまざまな情報の交換・保存・伝達等における最も重要なメディアとなっている。複雑な現象や実験結果等の各種の情報を，コンピュータグラフィックス（CG）を用いて人間に理解しやすい形で視覚化し，画像や映像として表現する技術がコンピュータービジュアリゼーション（Computer Visualization）（「ビジュアリゼーション」または「可視化」）である。
  - 中嶋正之，藤代一成編著「コンピュータビジュアリゼーション」，共立出版，2000.
- 可視化とは「CG」のことではない
  - CGに至るまでの様々な処理を「可視化」という

可視化の重要性

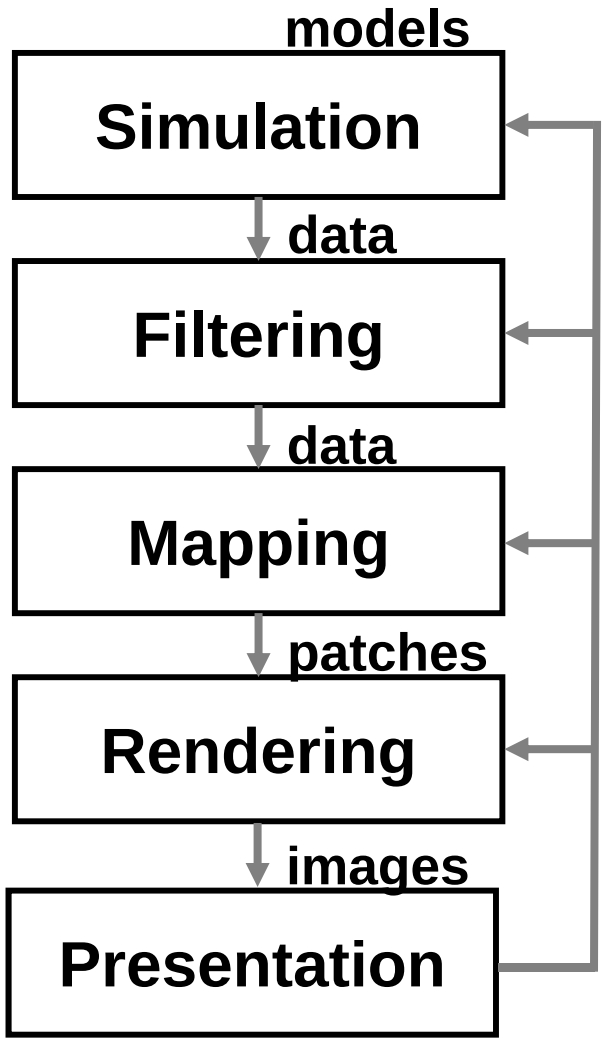
- 中島が社会人になったころ（1985年）は，シミュレーションは二次元が中心で，FEM（有限要素法）のモデルを使っても1,000メッシュ程度であった。
  - リストを出力し，それを「読む」ことによって結果を評価していた（モデルのチェックも含む）。
- 三次元，並列（分散）処理によるシミュレーションが主流になりつつある現在，可視化技術の重要性は30年前とは比較にならないくらい大きい。
  - 効率的に特徴をつかむ方法。
  - 「立体視」ができるにしても，あくまでも二次元画面への投影が中心。

「並列」可視化

- 並列シミュレーションの結果を，並列計算機を使用して可視化すること。
- ここでは，並列シミュレーションによって得られた分散データ(ファイルまたはメモリイメージ)を処理して，一枚の画像で見ることができるようにすること。
- 結果データは非常に大規模→単一データは不可能



Data-Flow Paradigm for Parallel Visualization (Fujishiro et al.)



Computational Efficiency

PB (parallel backend)
Supercomputer

VF (visualization front end)

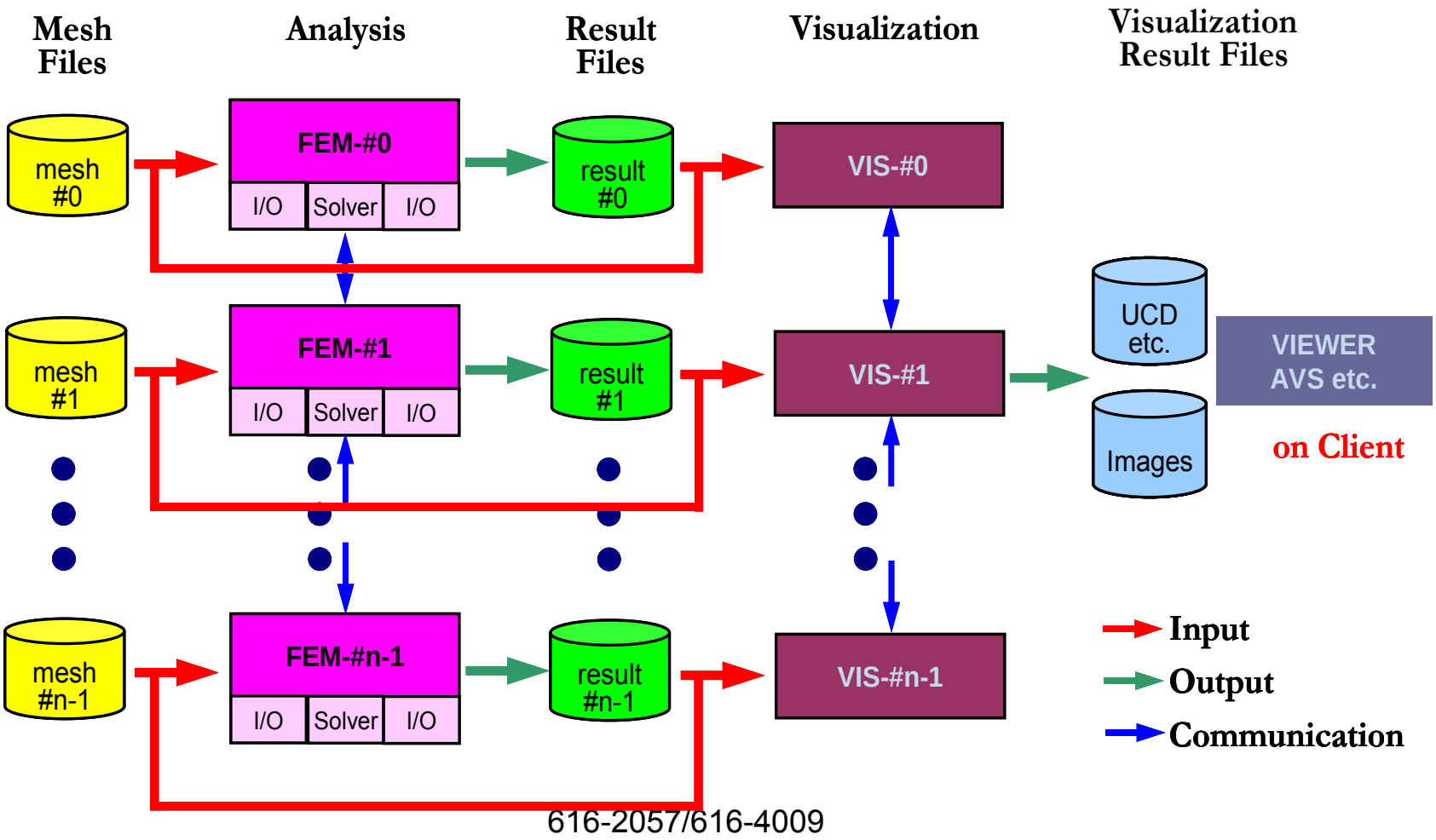
Interactivity

GeoFEM，HPC-MWにおける並列可視化機能の特徴（20世紀末～今世紀初頭）

- 様々な可視化手法，メッシュ体系をサポート
- 特殊なハードウェア，ライブラリは不要
- 高い並列性能
- 複雑形状への適用性
- 様々なハードウェアに対する最適化
- 使用法
  - ファイル渡し，または，メモリ渡し
  - Patch File（AVS）またはImage File（BMP）
  - メモリ渡しは結果ファイルを残さない

並列可視化フレームワーク 1

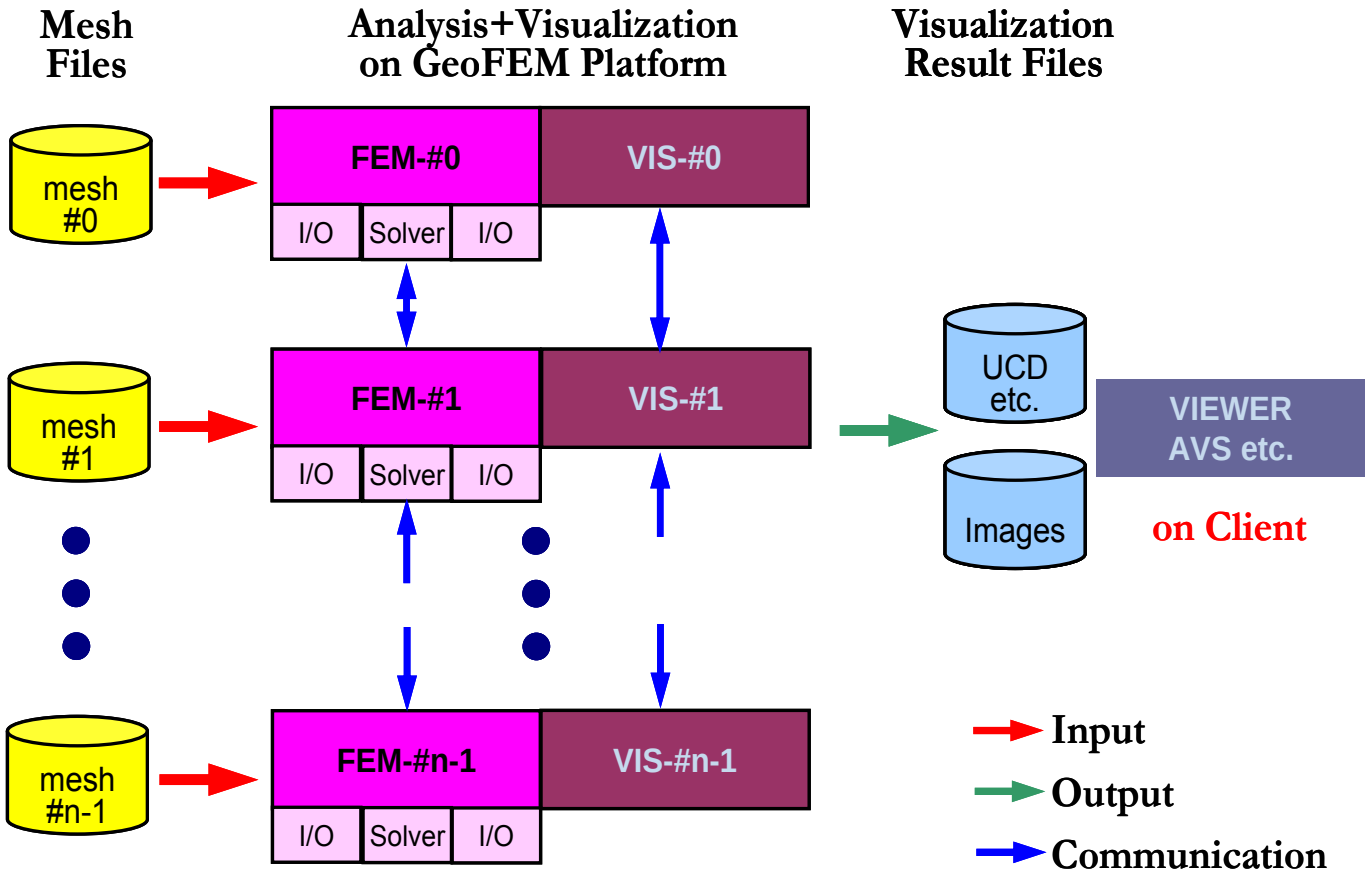
ファイル渡しバージョン



616-2057/616-4009

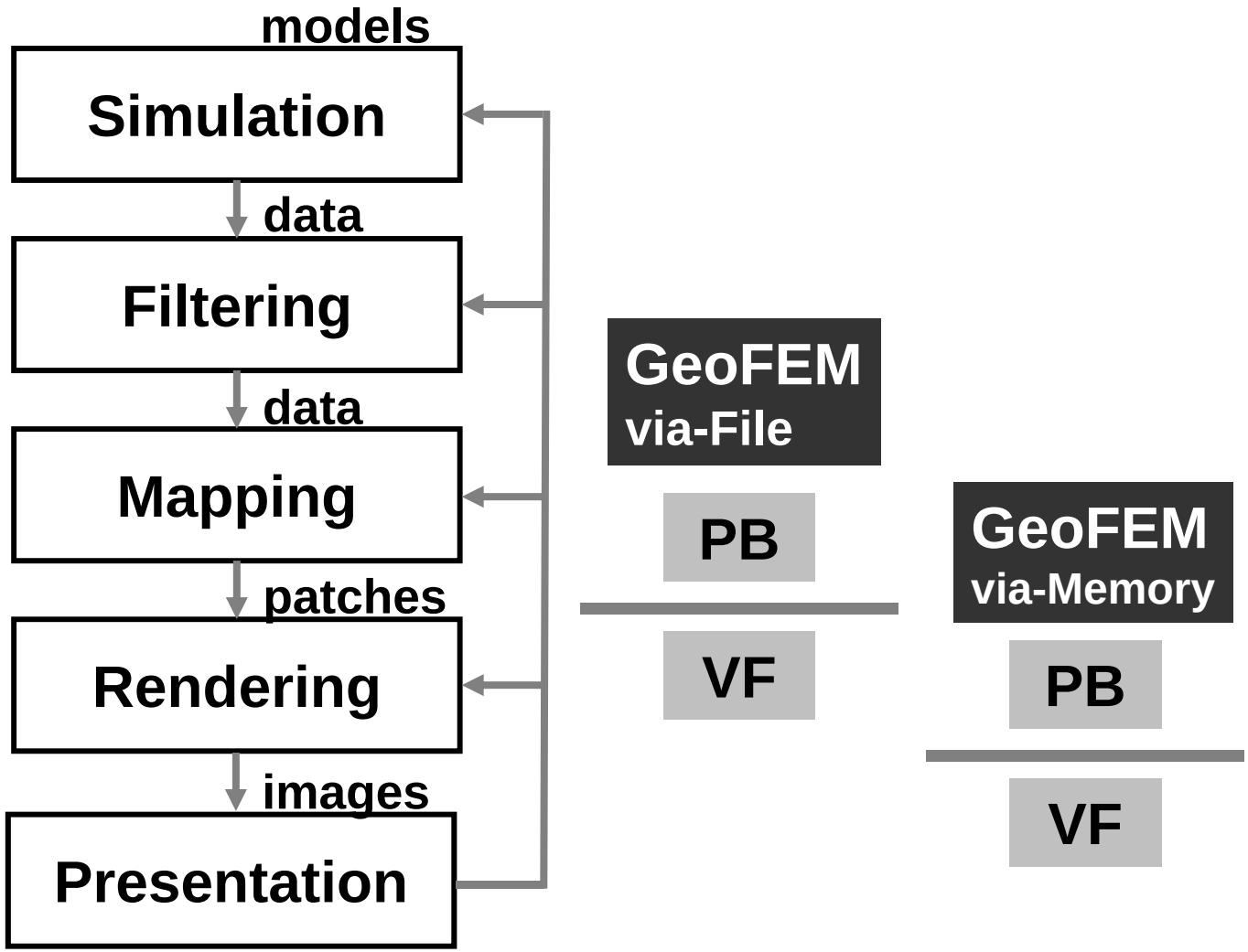
並列可視化フレームワーク 2

メモリ渡しバージョン



616-2057/616-4009

Data-Flow Paradigm for Parallel Visualization (Fujishiro et al.)



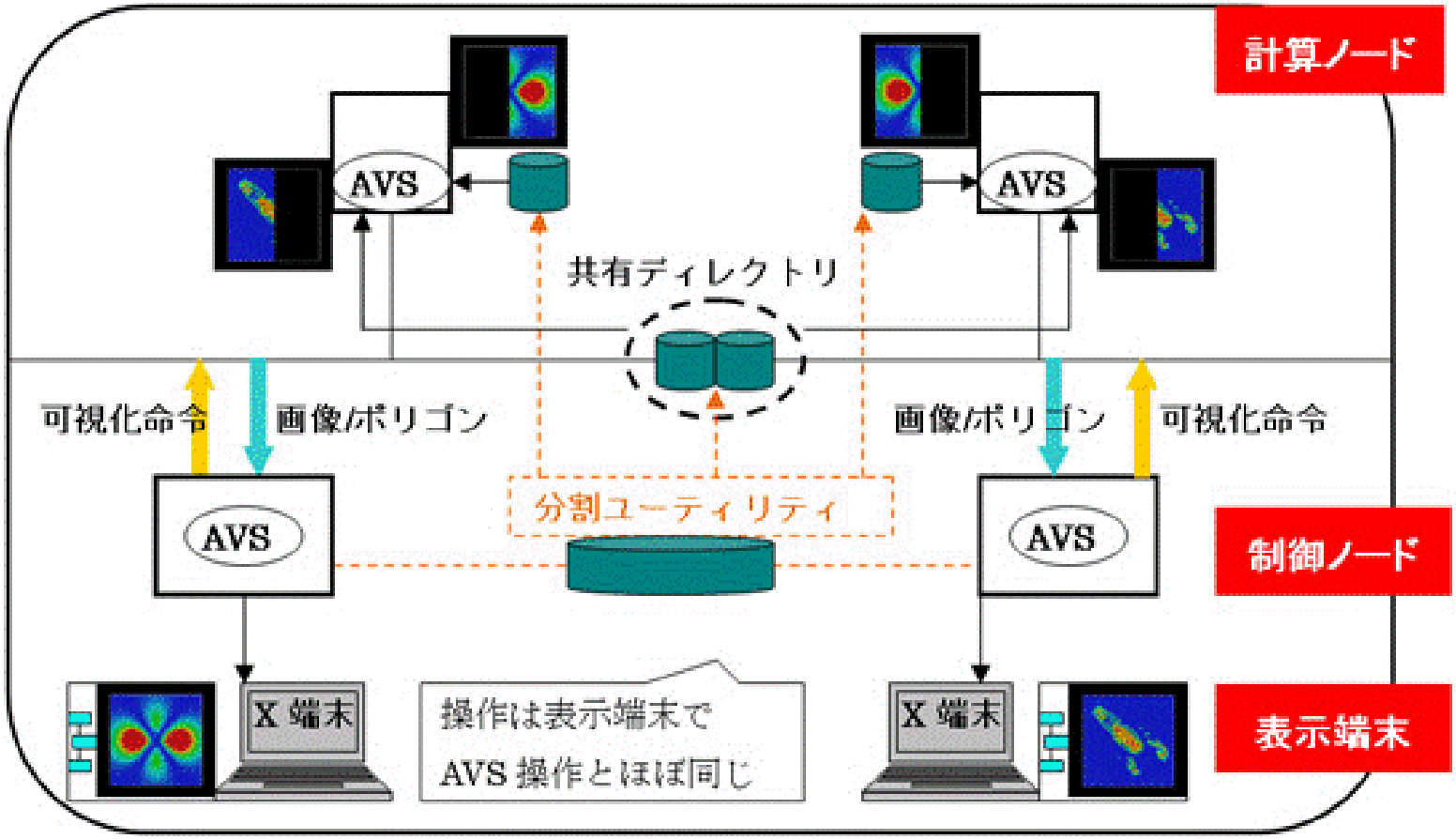
AVS/Express PCE Parallel Cluster Edition

http://www.cybernet.co.jp/avs/products/pce/

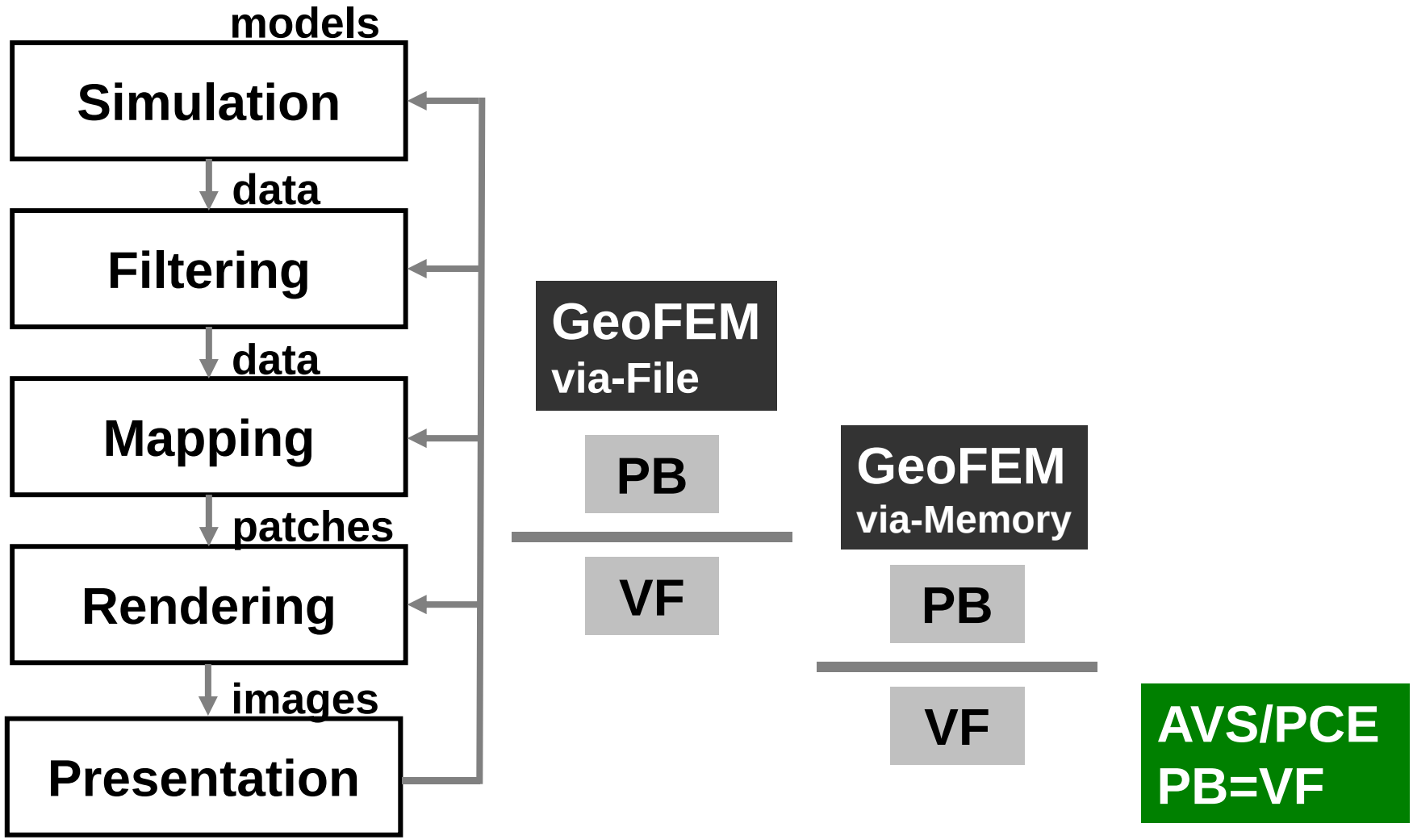
- AVS/Express PCEでは，クラスタ化された複数のLinuxマシンで，各計算ノードが持つ部分領域のみを可視化し，最終的な可視化結果のみ制御ノード上で表示するという構成になっている。
- 並列計算の結果，出力される大規模データを可視化する場合でも，高い精度を保ったまま，可視化処理を実現することが可能。
- 並列計算機上で対話処理可能
  - Windowsより制御可能
  - T2K東大に導入（～4ノードまで使用可能）：バッチ環境

AVS/Express PCE
Parallel Cluster Edition

（旧）KGT社　HPより



Data-Flow Paradigm for Parallel Visualization (Fujishiro et al.)

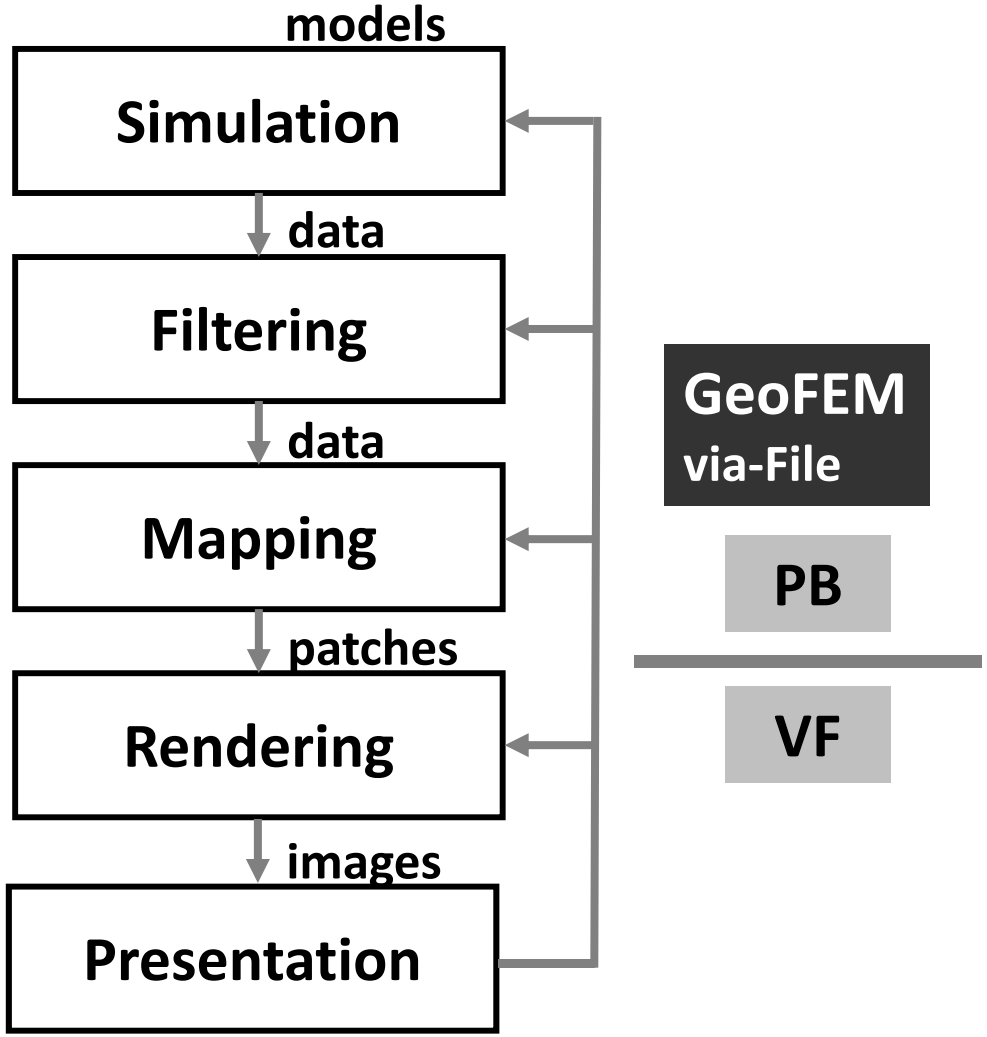


AVS/Express PCE Parallel Cluster Edition（cont.）

http://www.cybernet.co.jp/avs/products/pce/

- ノード数が増えた場合，部分領域を集めるプロセスがボトルネックとなる
  - MPI_Gather
  - アルゴリズムの改良が必要
- 小野謙二博士（理研AICS）らの研究
  - 京コンピュータ上での並列可視化システム

ppOpen-HPCにおける並列可視化の考え方



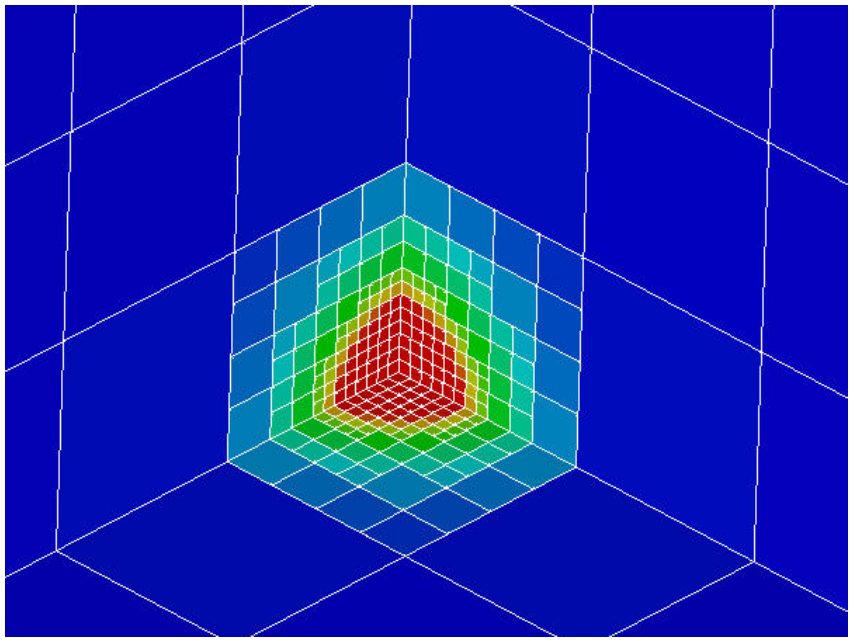
- 自己完結的なファイルを生成してPCで見る（e.g. ParaView，MicroAVS）。
- GeoFEMの場合はPatch抽出型で，例えば視点を変えることはできたが，可視化する変数，切り出す面等を変更することはできなかった。
- ピーク（最大，最小），分布を抑えることが大事，形状もある程度再現できていてほしい。

ppOpen-HPCにおける並列可視化の考え方

- 自己完結的なファイルを生成して「PC」で見る
  - GeoFEMの場合はPatch抽出型で，例えば視点を変えることはできたが，可視化する変数，切り出す面等を変更することはできなかった。
  - ピーク（最大，最小），分布を抑えることが大事，形状もある程度再現できていてほしい。
- 「見る」ためにスパコンは使わない
- 「絵を出すために計算をやり直す」という考え方も採らない
- 自己完結的ファイルができたら後はParaView，MicroAVSに任せる
- 大型計算機センターとしては，つぎ込めるだけの予算を計算エンジンにつぎ込みたい

ppOpen-MATH/VIS

- ボクセル型背景格子を使用した大規模並列可視化手法〔Nakajima & Chen 2006〕に基づく
  - 差分格子用バージョン公開：ppOpen-MATH/VIS-FDM3D
- UCD single file
- プラットフォーム
  - T2K，Cray
  - FX10
  - Flat MPI
    - Hybrid，非構造格子：開発中

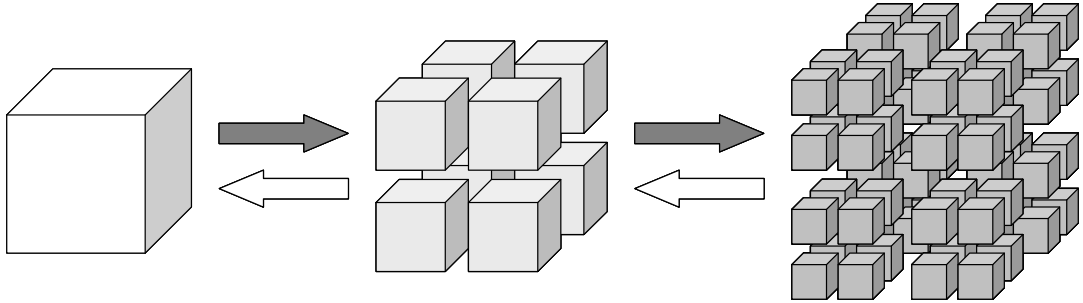


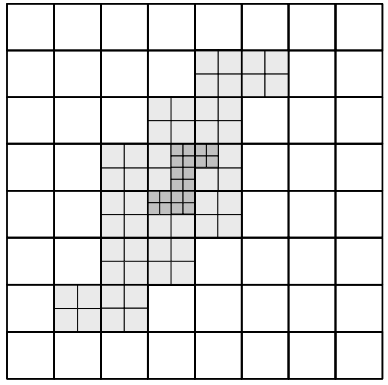
```
[Refine]
AvailableMemory = 2.0    Available memory size (GB), not available in this version.
MaxVoxelCount   = 500    Maximum number of voxels
MaxRefineLevel  = 20     Maximum number of refinement levels
```

Simplified Parallel Visualization using Background Voxels

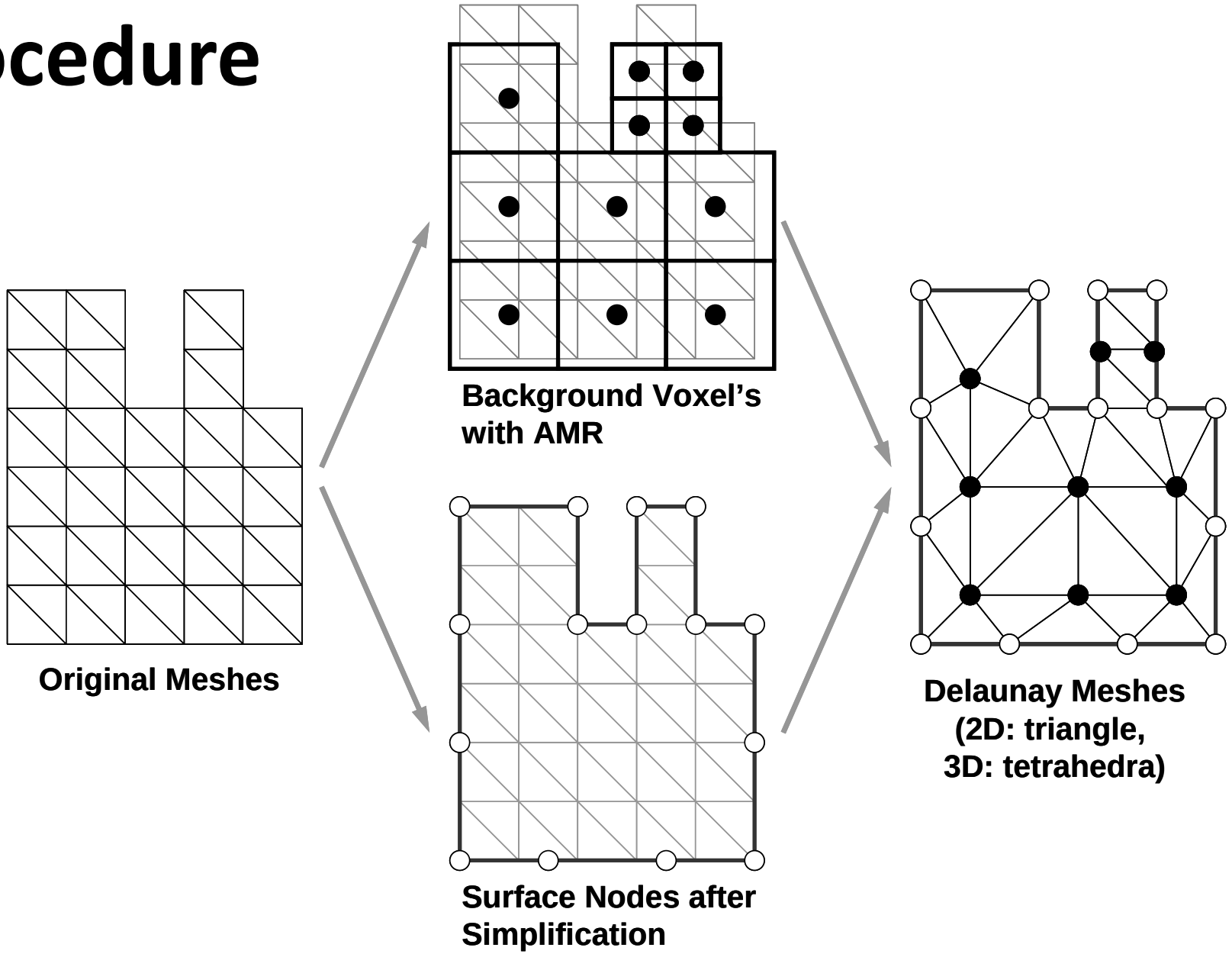
[KN, Chen 2006]

- Octree-based AMR
- AMR applied to the region where gradient of field values are large
  - stress concentration, shock wave, separation etc.
- If the number of voxels are controled, a single file with 10^5 meshes is possible, even though entire problem size is 10^9 with distributed data sets.

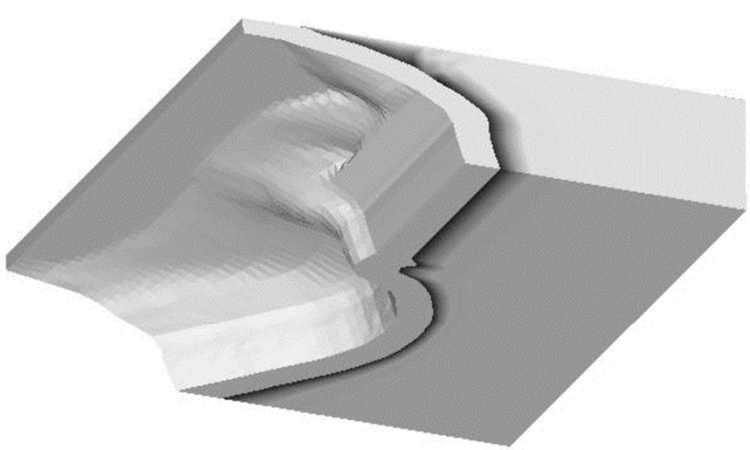


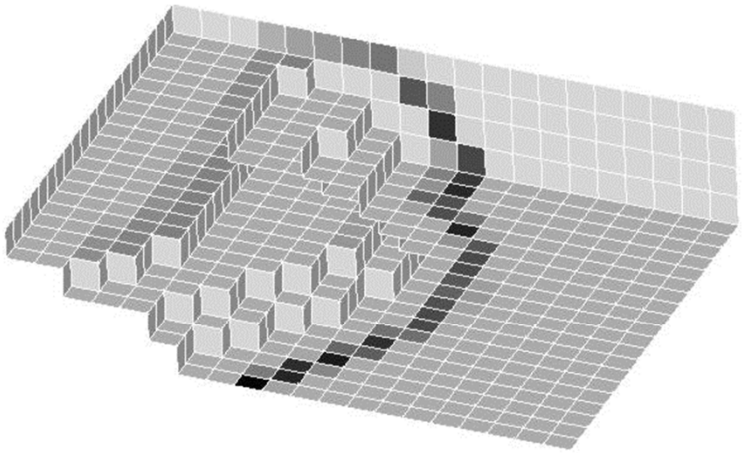


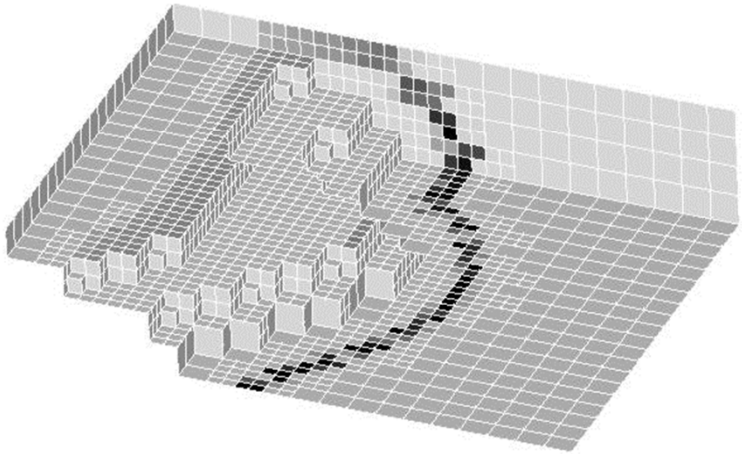
Procedure

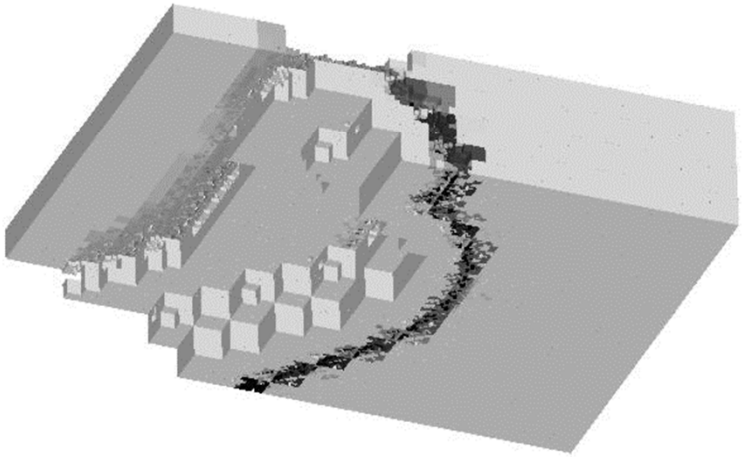

Original Meshes
Background Voxel's with AMR
Surface Nodes after Simplification
Delaunay Meshes (2D: triangle, 3D: tetrahedra)

Voxel Mesh (adapted)

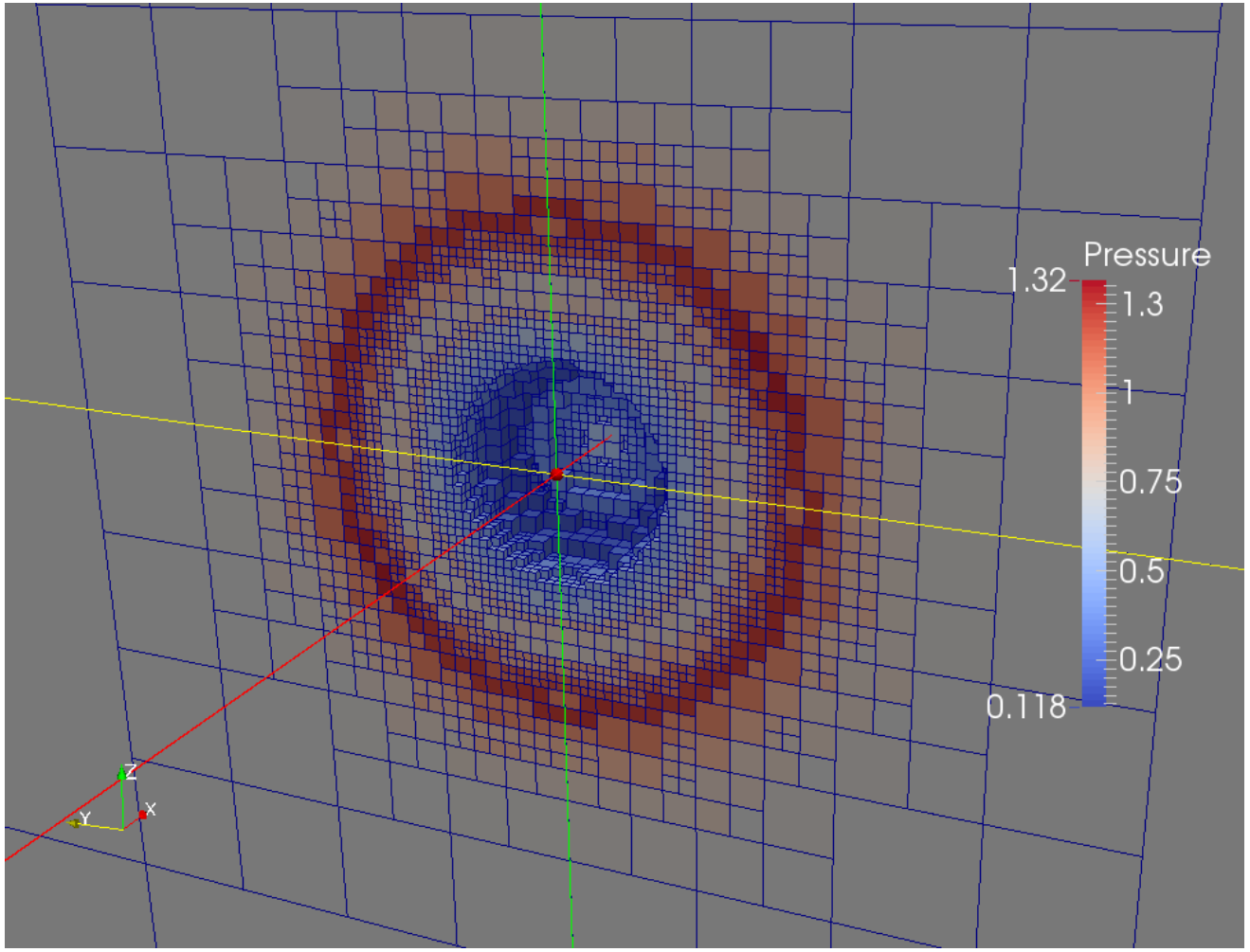






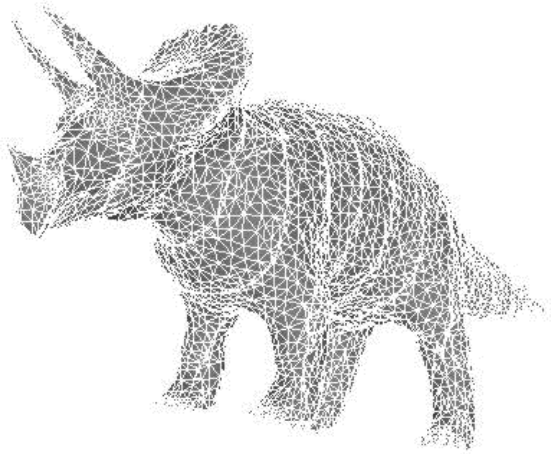


Flow around a sphere

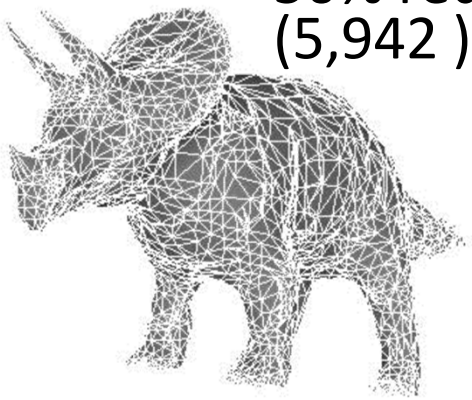


Example of Surface Simplification

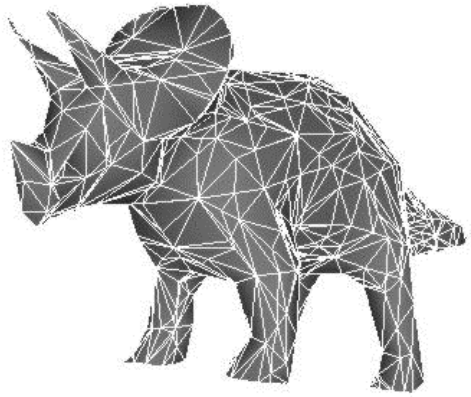
Initial
(11,884 tri's)



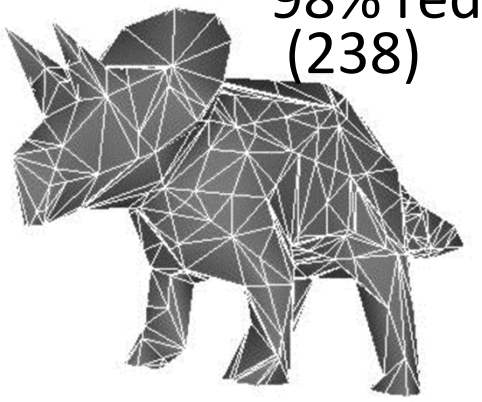
50% reduction
(5,942)



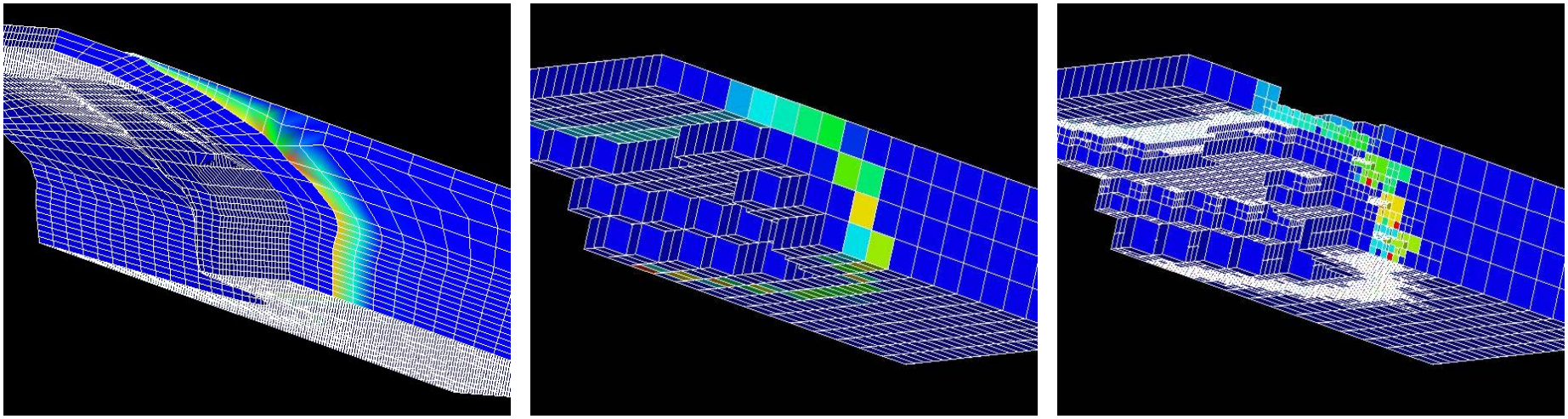
95% reduction
(594)



98% reduction
(238)



FEM Mesh (SW Japan Model)



pFEM3D + ppOpen-MATH/VIS

コピー

```
>$ cd ~/pFEM
>$ cp /home/ss/aics60/2014Summer/pVIS.tar .
>$ tar xvf pVIS.tar
```

FORTRANユーザー

```
>$ cd ~/pFEM/pVIS/F/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

Cユーザー

```
>$ cd ~/pFEM/pVIS/C/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

Makefile

```
CFLAGSL  = -I/home/ss/aics60/ppohFVM-tutorial/ppohFILES/include
LDFLAGSL = -L/home/ss/aics60/ppohFVM-tutorial/ppohFILES/lib
LIBSL    = -lppohvispfem3d

.SUFFIXES:
.SUFFIXES: .o .c

.c.o:
        $(CC) -c $(CFLAGS) $(CFLAGSL) $< -o $@

TARGET = ../run/pfem3d_test

OBJS = ¥
        test1.o ...

all: $(TARGET)

$(TARGET): $(OBJS)
        $(CC) -o $(TARGET) $(CFLAGS) $(CFLAGSL) $(OBJS)
$(LDFLAGSL) $(LIBS) $(LIBSL)
        rm -f *.o *.mod
```

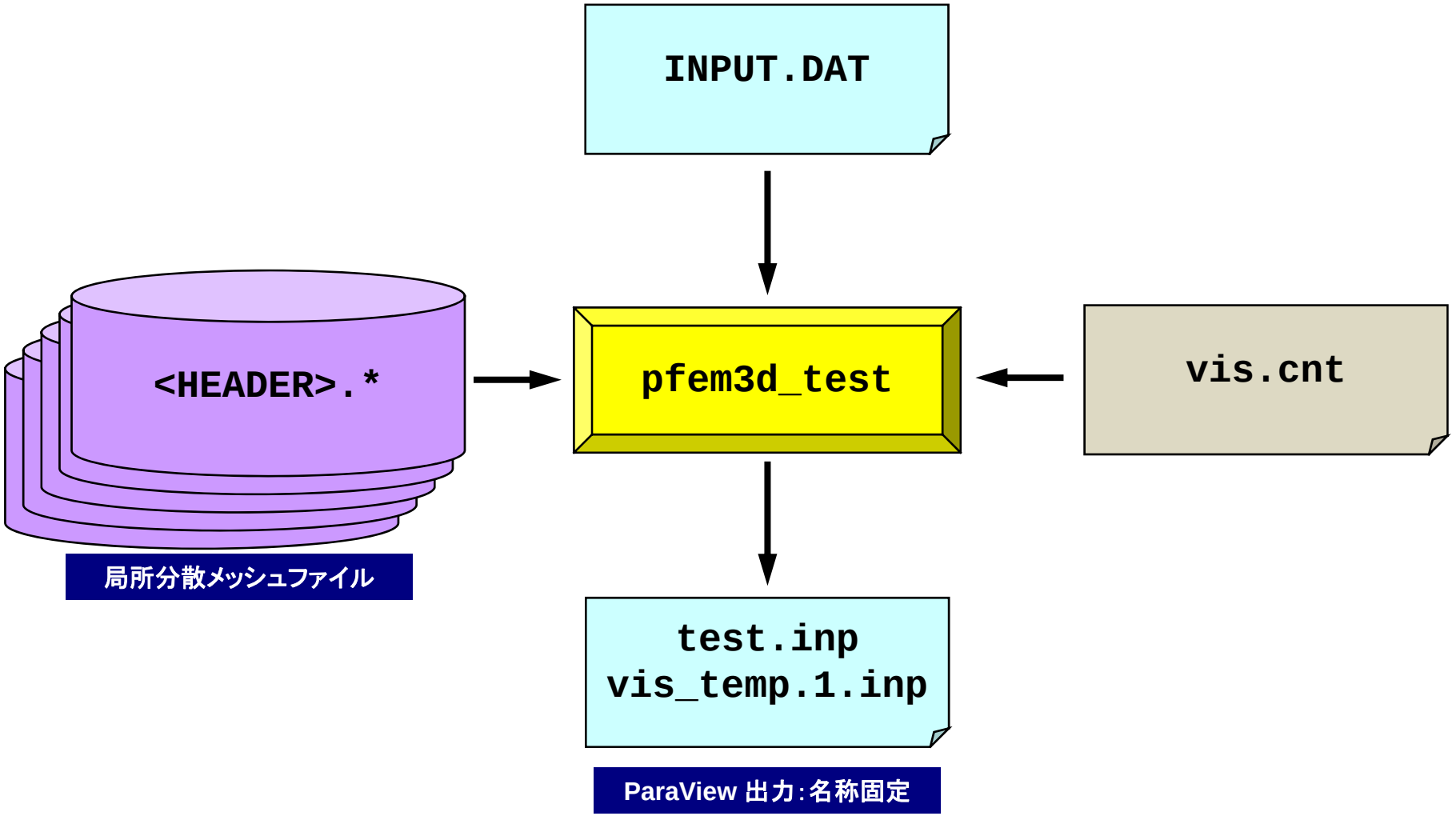
~/pFEM/pVIS/F(C)/run

```
cube_20x20x20_4pe_kmetis.0
cube_20x20x20_4pe_kmetis.1
cube_20x20x20_4pe_kmetis.2
cube_20x20x20_4pe_kmetis.3
cube_20x20x20_4pe.out

go.sh
INPUT.DAT
vis.cnt
vis_temp.1.inp
```

```
cube_20x20x20_4pe_kmetis
2000
1.0 1.0
1.0e-08
```

```
#!/bin/sh

#PJM -L "rscgrp=school"
#PJM -L "node=4"
#PJM --mpi "proc=4"
#PJM -L "elapse=00:10:00"
#PJM -j
#PJM -o "cube_20x20x20_4pe.out"

mpiexec ./pfem3d_test
```

pFEM3D + ppOpen-MATH/VIS



Fortran/main (1/2)

```
      use solver11
      use pfem_util
      use ppohvis_pfem3d_util

      implicit REAL*8(A-H,O-Z)
      type(ppohVIS_BASE_stControl)               :: pControl
      type(ppohVIS_BASE_stResultCollection)      :: pNodeResult
      type(ppohVIS_BASE_stResultCollection)      :: pElemResult
      character(len=PPOHVIS_BASE_FILE_NAME_LEN) :: CtrlName
      character(len=PPOHVIS_BASE_FILE_NAME_LEN) :: VisName
      character(len=PPOHVIS_BASE_LABEL_LEN)      :: ValLabel
      integer(kind=4)                             :: iErr

      CtrlName = ""
      CtrlName = "vis.cnt"

      VisName = ""
      VisName = "vis"

      ValLabel = ""
      ValLabel = "temp"

      call PFEM_INIT

      call ppohVIS_PFEM3D_Init(MPI_COMM_WORLD, iErr)
      call ppohVIS_PFEM3D_GetControl(CtrlName, pControl, iErr);
      call INPUT_CNTL
      call INPUT_GRID

      call ppohVIS_PFEM3D_SETMESHEX(                                    &
     &       NP,      N,           NODE_ID, XYZ,                      &
     &       ICELTOT, ICELTOT_INT, ELEM_ID, ICELNOD,                  &
     &       NEIBPETOT, NEIBPE, IMPORT_INDEX, IMPORT_ITEM,            &
     &                          EXPORT_INDEX, EXPORT_ITEM, iErr)
```

Fortran/main (2/2)

```
      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD

      pNodeResult%ListCount = 1
      pElemResult%ListCount = 0
      allocate(pNodeResult%Results(1))

      call ppohVIS_PFEM3D_ConvResultNodeItem1N(                          &
     &       NP, ValLabel, X, pNodeResult%Results(1), iErr)

      call ppohVIS_PFEM3D_Visualize(pNodeResult, pElemResult, pControl, &
     &                              VisName, 1, iErr)

      call PFEM_FINALIZE

      end program heat3Dp
```

C/main (1/2)

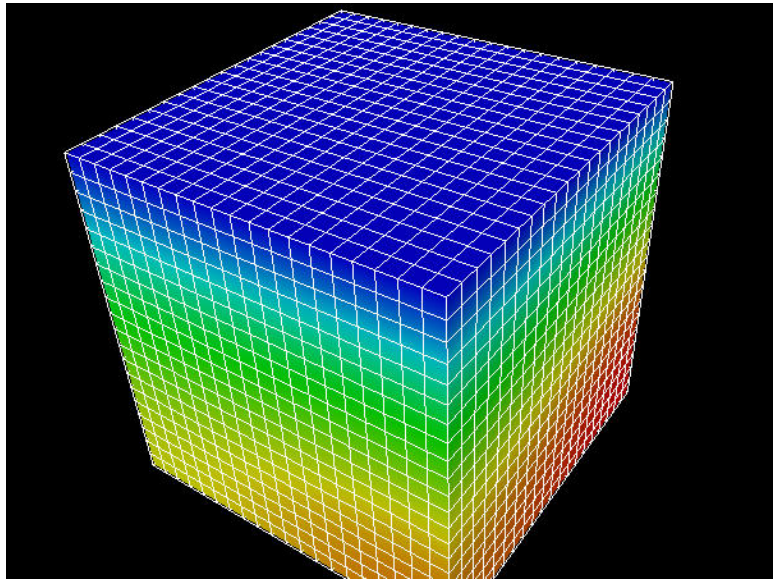
```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
#include "ppohVIS_PFEM3D_Util.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
extern void MAT_ASS_BC();
extern void SOLVE11();
extern void OUTPUT_UCD();
extern void PFEM_FINALIZE();
int main(int argc,char* argv[])
{
  double START_TIME,END_TIME;
  struct ppohVIS_FDM3D_stControl *pControl = NULL;
  struct ppohVIS_FDM3D_stResultCollection *pNodeResult = NULL;

  PFEM_INIT(argc,argv);
  ppohVIS_PFEM3D_Init(MPI_COMM_WORLD);
  pControl = ppohVIS_FDM3D_GetControl("vis.cnt");

  INPUT_CNTL();
  INPUT_GRID();

  if(ppohVIS_PFEM3D_SetMeshEx(
      NP,N,NODE_ID,XYZ,
      ICELTOT,ICELTOT_INT,ELEM_ID,ICELNOD,
      NEIBPETOT,NEIBPE,IMPORT_INDEX,IMPORT_ITEM,EXPORT_INDEX,EXPORT_ITEM)) {
                ppohVIS_BASE_PrintError(stderr);
                MPI_Abort(MPI_COMM_WORLD,errno);
  };
```

C/main (2/2)

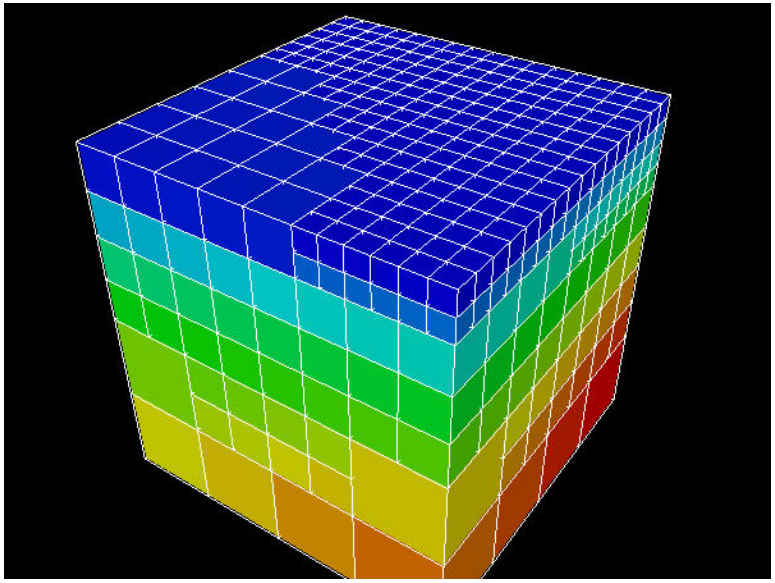
```
  MAT_CON0();
  MAT_CON1();

  MAT_ASS_MAIN();
  MAT_ASS_BC()  ;

  SOLVE11();

  OUTPUT_UCD();

  pNodeResult=ppohVIS_BASE_AllocateResultCollection();
        if(pNodeResult == NULL) {
                ppohVIS_BASE_PrintError(stderr);
                MPI_Abort(MPI_COMM_WORLD,errno);
        };
        if(ppohVIS_BASE_InitResultCollection(pNodeResult, 1)) {
                ppohVIS_BASE_PrintError(stderr);
                MPI_Abort(MPI_COMM_WORLD,errno);
        };

        pNodeResult->Results[0] =

  ppohVIS_PFEM3D_ConvResultNodeItemPart(NP,1,0,"temp",X);

  START_TIME= MPI_Wtime();
        if(ppohVIS_PFEM3D_Visualize(pNodeResult,NULL,pControl,"vis",1)) {
                ppohVIS_BASE_PrintError(stderr);
                MPI_Abort(MPI_COMM_WORLD,errno);
        };

  ppohVIS_PFEM3D_Finalize();

  PFEM_FINALIZE() ;
}
```

vis.cnt

```
[Refine]                    細分化制御情報セクション
AvailableMemory = 2.0       利用可能メモリ容量（GB）not in use
MaxVoxelCount = 1000        Max Voxel #
MaxRefineLevel = 20         Max Voxel Refinement Level
[Simple]                    簡素化制御情報セクション
ReductionRate = 0.0         表面パッチ削減率
```



1.52 MB
8,000 elements



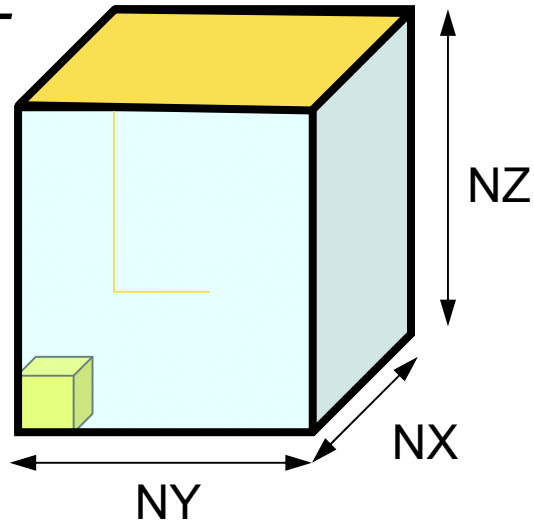
.385 MB,
813 elements

現状

- 実はまだ，最適化が進んでおらず，ノード数が増えると時間がかかる。

簡易可視化方法

- 各領域が規則正しい直方体構造となっていることを仮定
  - 一様形状である必要はない
  - pmeshで生成されるようなメッシュ
- 最終的に出力するParaView用出力ファイルの全体のメッシュ数を規定
- 各部分領域（MPIプロセス）の従属変数分布から，各領域に割り当てる「可視化用」メッシュ数の決定
  - 八分木で領域ごとに可視化用メッシュ生成
  - 値の変化の多い領域にメッシュ数を多く割当
    - ルールは色々と検討する必要がある
- 各領域で生成した可視化用メッシュを集める



代替法（プログラムはFortranのみ）

FORTRAN ユーザー

```
>$ cd ~/pFEM/pfem3dV/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

Cユーザー

```
>$ cd ~/pFEM/pfem3dV/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

Fortran/main

```
      program heat3Dp

      use solver11
      use pfem_util

      implicit REAL*8(A-H,O-Z)

      call PFEM_INIT
      call INPUT_CNTL
      call INPUT_GRID

      call MAT_CON0
      call MAT_CON1

      call MAT_ASS_MAIN
      call MAT_ASS_BC

      call SOLVE11

      call OUTPUT_UCD_REGULAR

      call PFEM_FINALIZE

      end program heat3Dp
```

制御ファイル：INPUT.DAT

```
../pmesh/pcube                    HEADER
2000                              ITER
1.0 1.0                           COND, QVOL
1.0e-08                           RESID
1000                              N_MESH_VIS
```

- HEADER：　局所分散ファイルヘッダ名，<HEADER>.my_rank
- ITER：　反復回数上限
- COND：　熱伝導率
- QVOL：　体積当たり発熱量係数
- RESID：　反復法の収束判定値
- N_MESH_VIS:　簡易可視化機能における表示メッシュ数の目安

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\dot{Q}(x,y,z)=0$$

$$\dot{Q}(x,y,z)=QVOL\left|x_C+y_C\right|$$

計算例

- 256×256×256節点（=16,777,216節点，16,581,375要素）
- 128コア
- 可視化
  - 2,970節点，834要素 Movie
- 各MPIプロセスで可視化データを生成してマージするので，MPIプロセス数が増えると重複する節点の数が増えてしまう⇒修正中