3D Parallel FEM (III) Parallel Visualization using ppOpen-MATH/VIS

Kengo Nakajima

Programming for Parallel Computing (616-2057)
Seminar on Advanced Computing (616-4009)

ppOpen-HPC

Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications with Automatic Tuning (AT)

Kengo Nakajima

Information Technology Center

Masaki Satoh (AORI/U. Tokyo), Takashi Furumura (ERI/U. Tokyo)

Hiroshi Okuda (GS Frontier Sciences/U. Tokyo), Takeshi Iwashita (ACCMS/Kyoto U.)

Hide Sakaguchi (JAMSTEC)

Post T2K System

- Will be installed FY.2014-2015, O(10^1-10^2) PFLOPS
  - under collaboration with U. Tsukuba
- Heterogeneous computing node will be adopted
  - best performance and well balanced memory-computation under limited power consumption.
- Multi-core CPU+GPU, Multi-core CPU+Many-core（e.g. Intel MIC/Xeon Phi）
  - TSUBAME 2.0 (Tokyo Tech)
  - HA-PACS (U.Tsukuba)
  - We are mainly thinking about MIC/Xeon-Phi-based system.
- Programming is difficult
  - (MPI+OpenMP) is already difficult
    - Explicit method is rather easier
  - OpenACC, CUDA, OpenCL

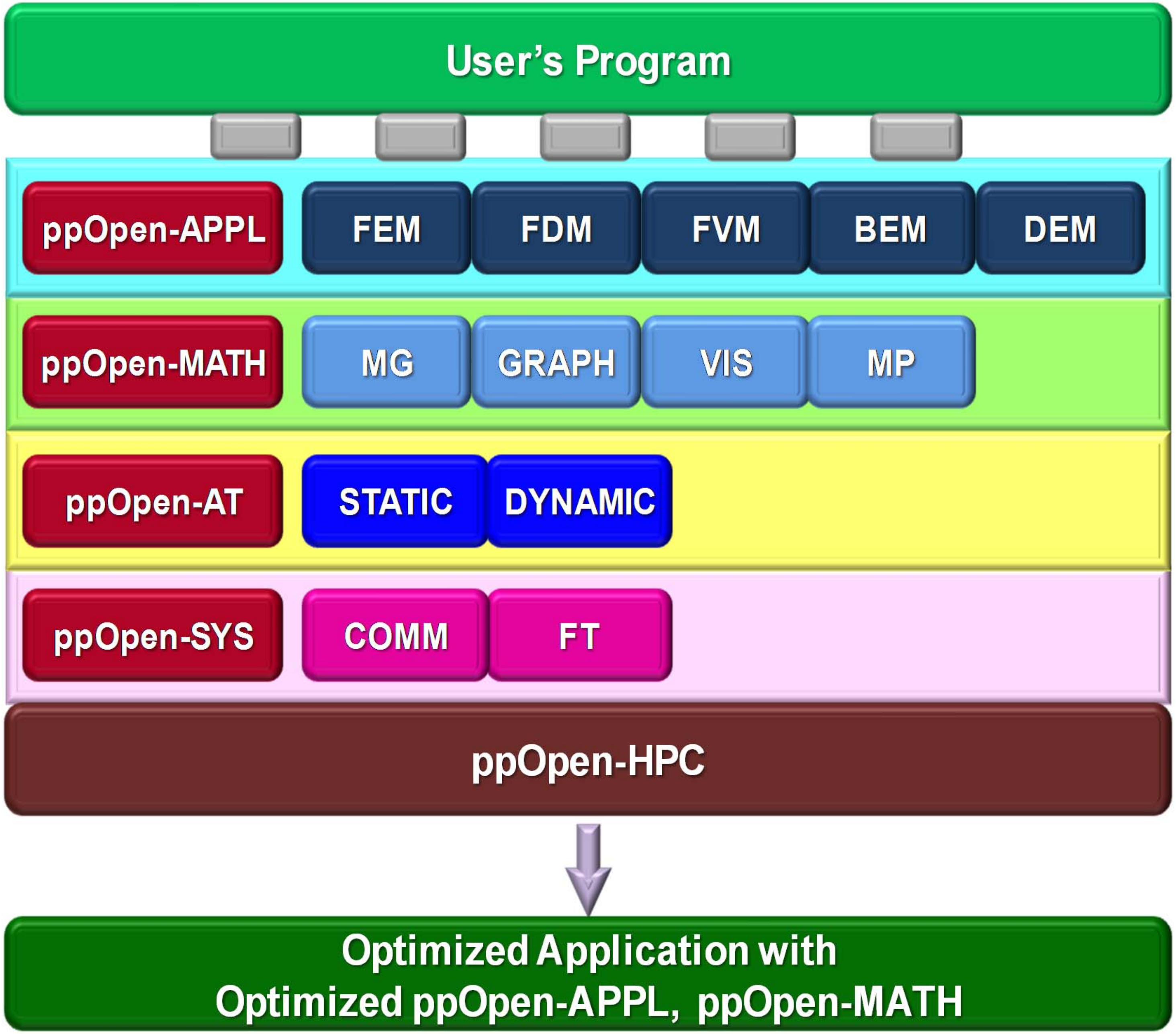
Key-Issues towards Appl./Algorithms on Exa-Scale Systems

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

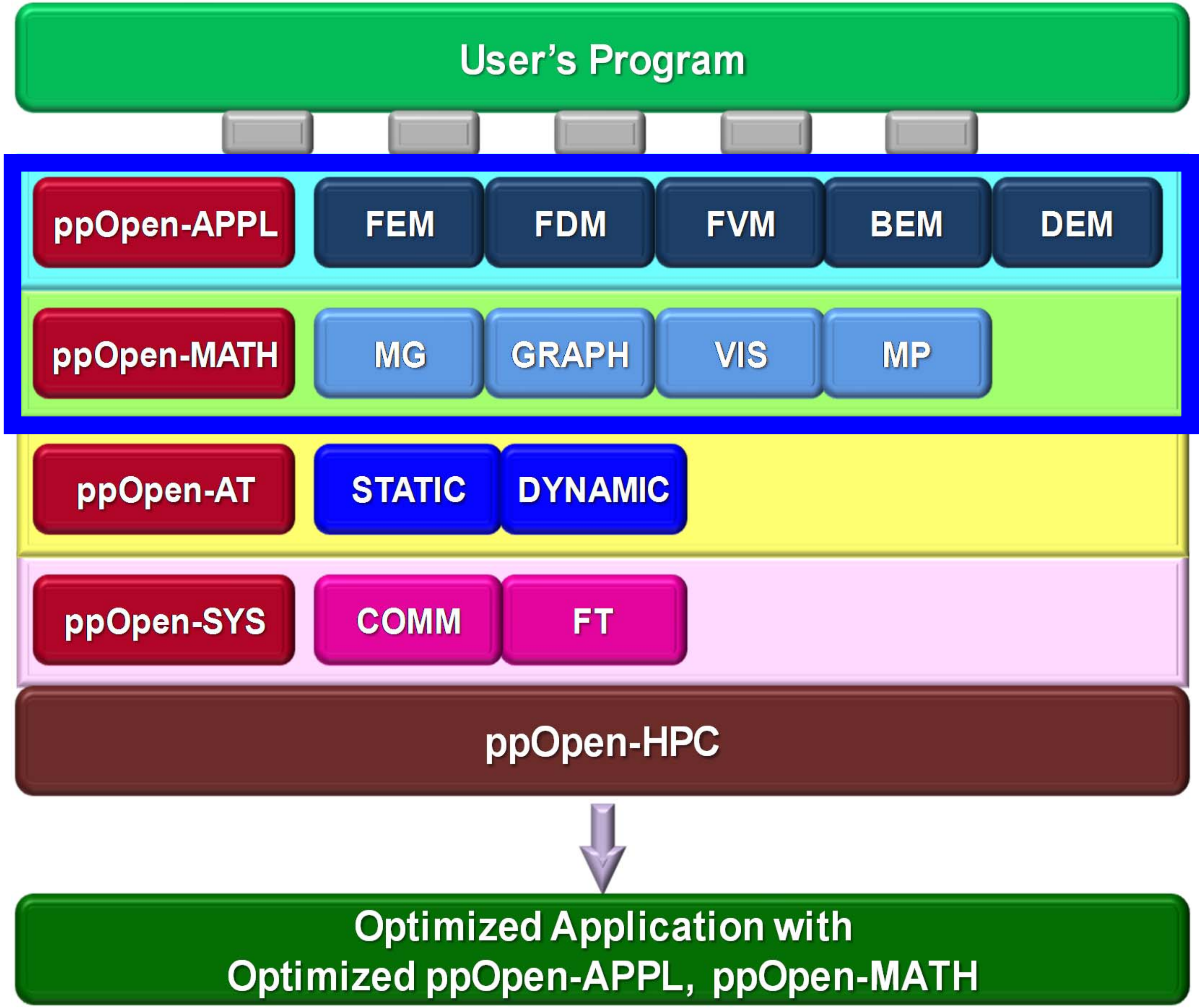
ppOpen-HPC (1/3)

- Open Source Infrastructure for development and execution of large-scale scientific applications on post-peta-scale supercomputers with automatic tuning (AT)
    - “pp” : post-peta-scale
- Five-year project (FY.2011-2015) (started in April 2011)
    - P.I.: Kengo Nakajima (ITC, The University of Tokyo)
    - Part of “Development of System Software Technologies for Post-Peta Scale High Performance Computing” funded by JST/CREST (Japan Science and Technology Agency, Core Research for Evolutional Science and Technology)
    - 4.5 M$ for 5 yr.
- Team with 6 institutes, >30 people (5 PDs) from various fields: Co-Desigin
    - ITC/U.Tokyo, AORI/U.Tokyo, ERI/U.Tokyo, FS/U.Tokyo
    - Kyoto U., JAMSTEC

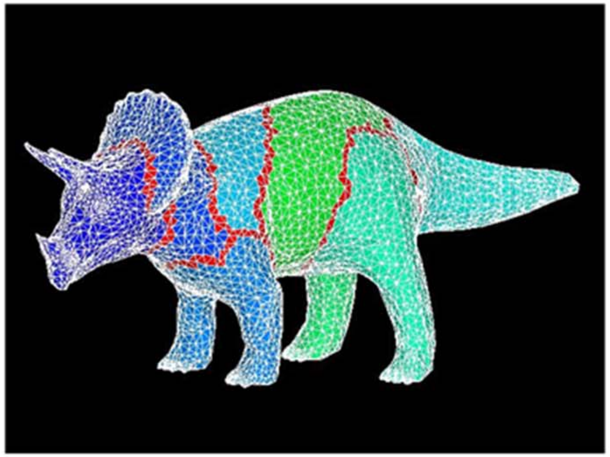

User's Program
ppOpen-APPL
FEM
FDM
FVM
BEM
DEM
ppOpen-MATH
MG
GRAPH
VIS
MP
ppOpen-AT
STATIC
DYNAMIC
ppOpen-SYS
COMM
FT
ppOpen-HPC
Optimized Application with
Optimized ppOpen-APPL, ppOpen-MATH

ppOpen-HPC (2/3)

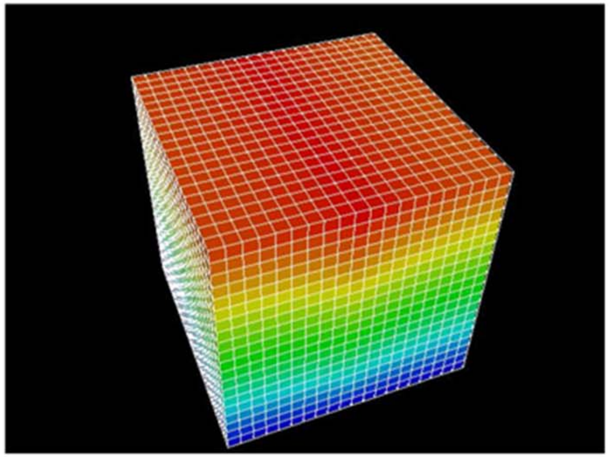
- ppOpen-HPC consists of various types of *optimized* libraries, which covers various types of procedures for scientific computations.
  - ppOpen-APPL/FEM, FDM, FVM, BEM, DEM
- Source code developed on a PC with a single processor is linked with these libraries, and generated parallel code is optimized for post-peta scale system.
- Users don’t have to worry about optimization tuning, parallelization etc.
  - CUDA, OpenGL etc. are hidden.
  - Part of MPI codes are also hidden.
  - OpenMP, OpenACC could be hidden


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

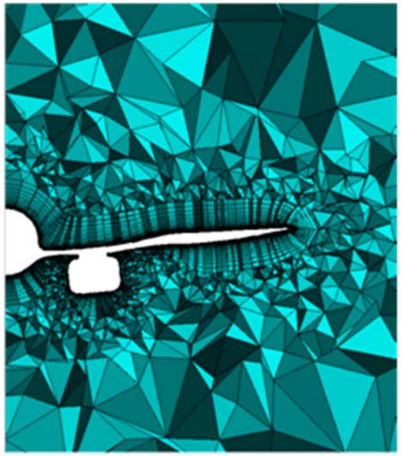
ppOpen-HPC covers ...



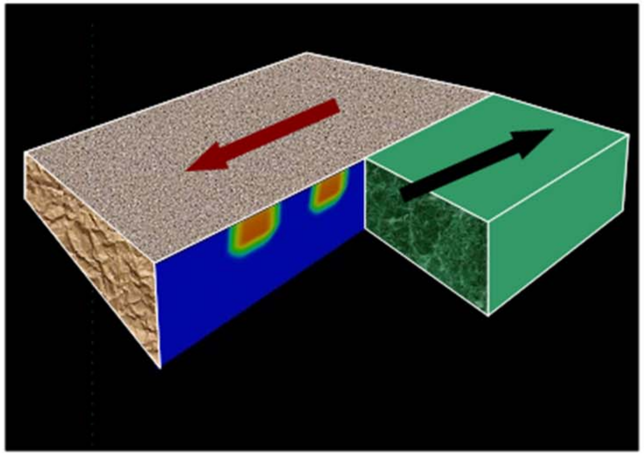
FEM
Finite Element Method



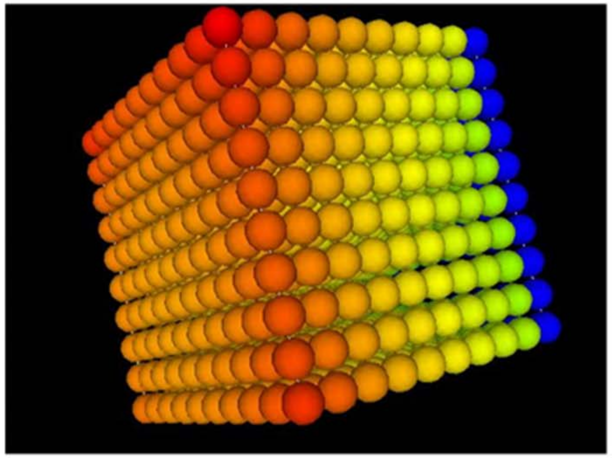
FDM
Finite Difference Method



FVM
Finite Volume Method



BEM
Boundary Element Method



DEM
Discrete Element Method

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
  - Matrix Assembling
  - AMR and Dynamic Load Balancing

Code developed on ppOpen-APPL/FEM

```
Program My_pFEM
use ppOpenFEM_util
use ppOpenFEM_solver

call ppOpenFEM_init
call ppOpenFEM_cntl
call ppOpenFEM_mesh
call ppOpenFEM_mat_init

do
  call ppOpenFEM_mat_ass
  call ppOpenFEM_mat_bc
  call ppOpenFEM_solve
  call ppOPenFEM_vis
  Time= Time + DT
enddo

call ppOpenFEM_finalize
stop
end
```

ppOpen-HPC (2/3)

- ppOpen-HPC consists of various types of *optimized* libraries, which covers various types of procedures for scientific computations.
  - ppOpen-APPL/FEM, FDM, FVM, BEM, DEM
- Source code developed on a PC with a single processor is linked with these libraries, and generated parallel code is optimized for post-peta scale system.
- Users don’t have to worry about optimization tuning, parallelization etc.
  - CUDA, OpenGL etc. are hidden.
  - Part of MPI codes are also hidden.
  - OpenMP, OpenACC could be hidden

ppOpen-HPC (3/3)

- Capability of automatic tuning (AT) enables development of optimized codes and libraries on emerging architecture based on results by existing architectures and machine parameters.
  - Mem. Access, Host/Co-Proc Balance, Comp/Comm Overlapping
  - Solvers & Libraries of ppOpen-HPC
  - OpenFOAM, PETSc
- Target system is post-peta-scale computer with heterogeneous computing nodes which consist of multicore CPU’s and accelerators, such as GPU’s and manycores.
  - Peak performance is $O(10^1\text{-}10^2)$ PFLOPS, and number of cores are $O(>10^6)$ cores.
  - Post T2K (MIC-based) to be installed in FY.2014-2015
  - ppOpen-HPC helps smooth transition of users to new system

Schedule of Public Release

(with English Documents)

- 4Q 2012
  - ppOpen-HPC for Multicore Cluster (Cray, K etc.)
  - Preliminary version of ppOpen-AT/STATIC
    - to be available in SC'12
- 3Q 2013
  - ppOpen-HPC for Multicore Cluster & Xeon Phi (& GPU)
- 3Q 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 v.0.1.0

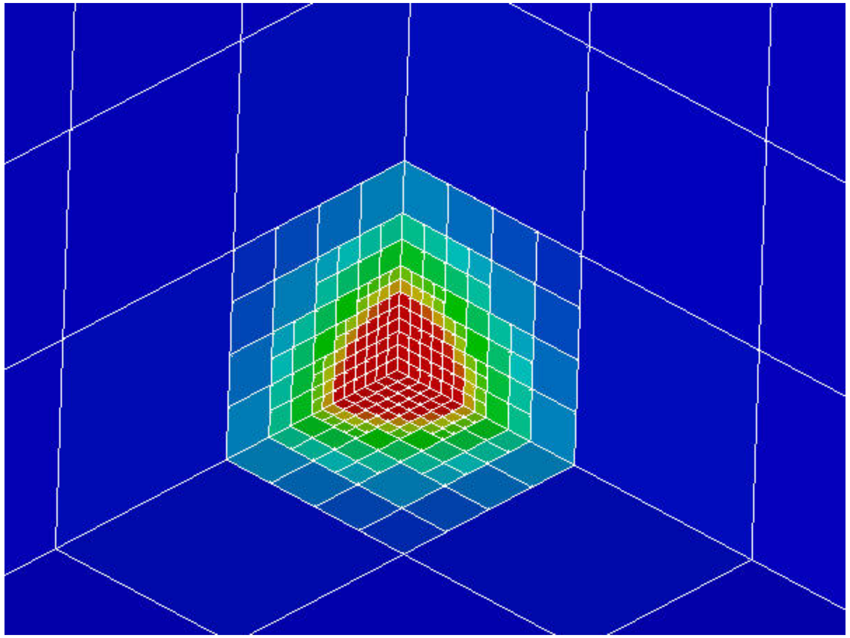
http://ppopenhpc.cc.u-tokyo.ac.jp/

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

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	-	-	○	○

ppOpen-MATH/VIS

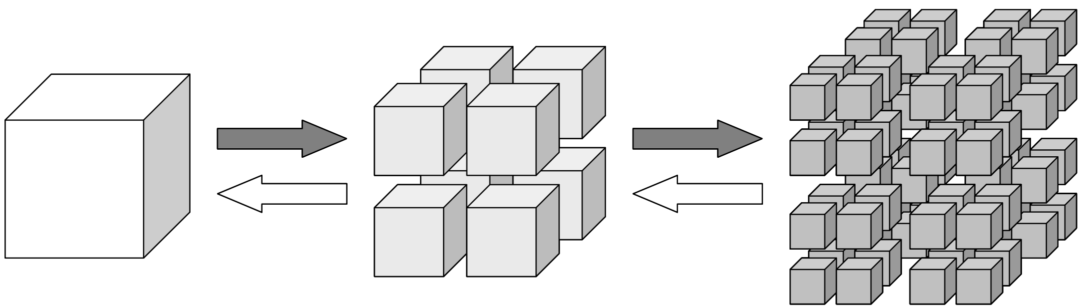
- Parallel Visualization using Information of Background Voxels [Nakajima & Chen 2006]
  - FDM version is released: ppOpen-MATH/VIS-FDM3D
- UCD single file
- Platform
  - T2K, Cray
  - FX10
  - Flat MPI
- Unstructured/Hybrid version
  - Next release

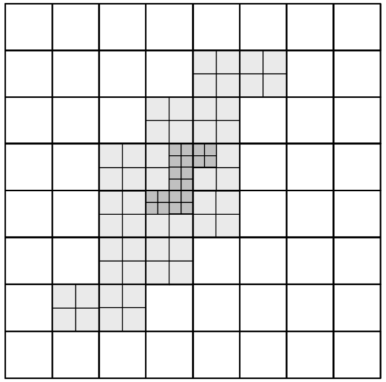


```
[Refine]
AvailableMemory = 2.0     Available memory size (GB), not available in this version.
MaxVoxelCount   = 500     Maximum number of voxels
MaxRefineLevel  = 20      Maximum number of refinement levels
```

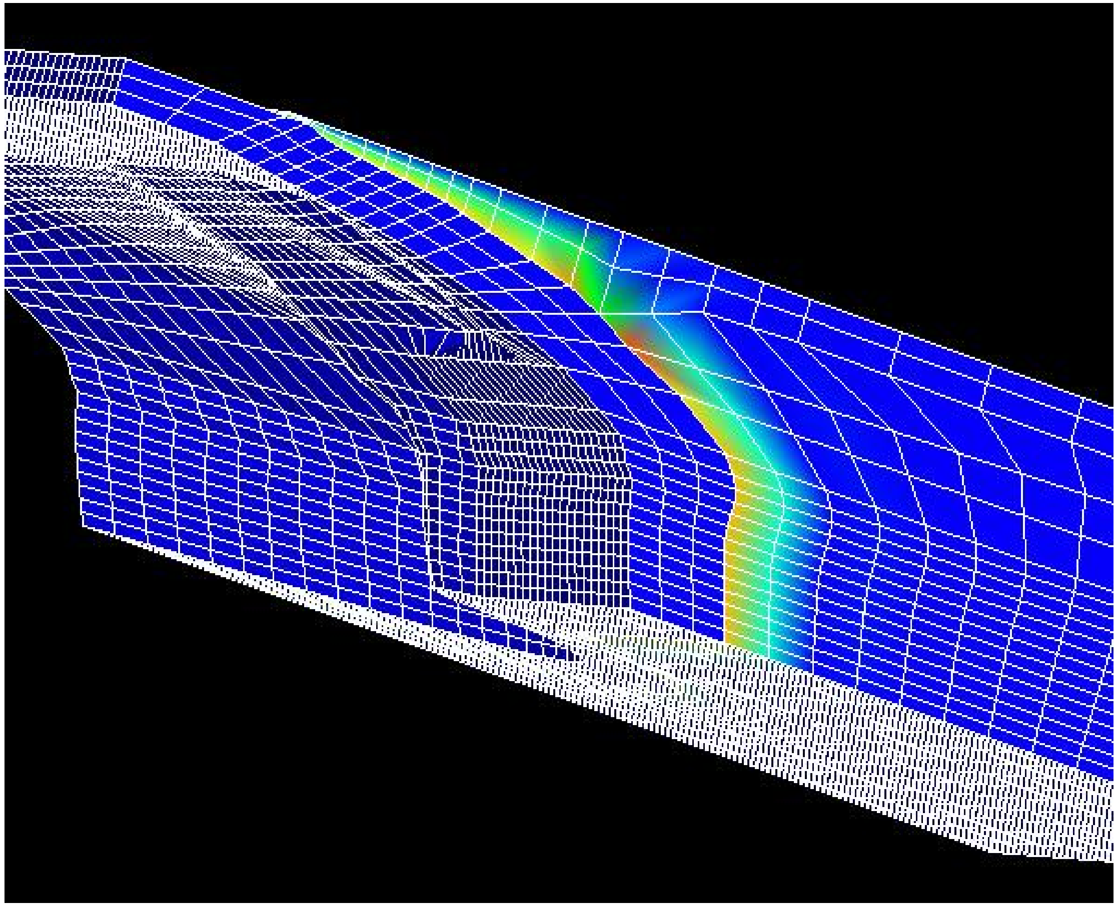
Simplified Parallel Visualization using Background Voxels

- 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.

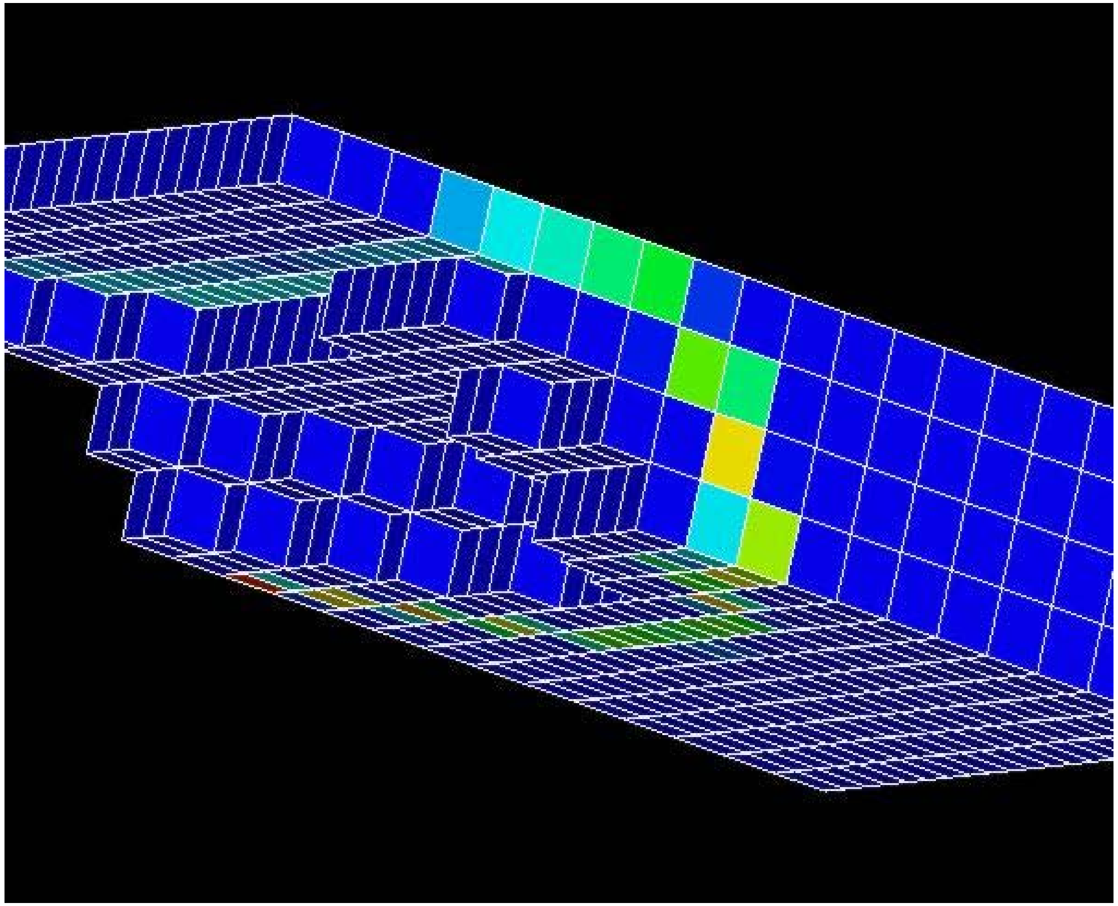




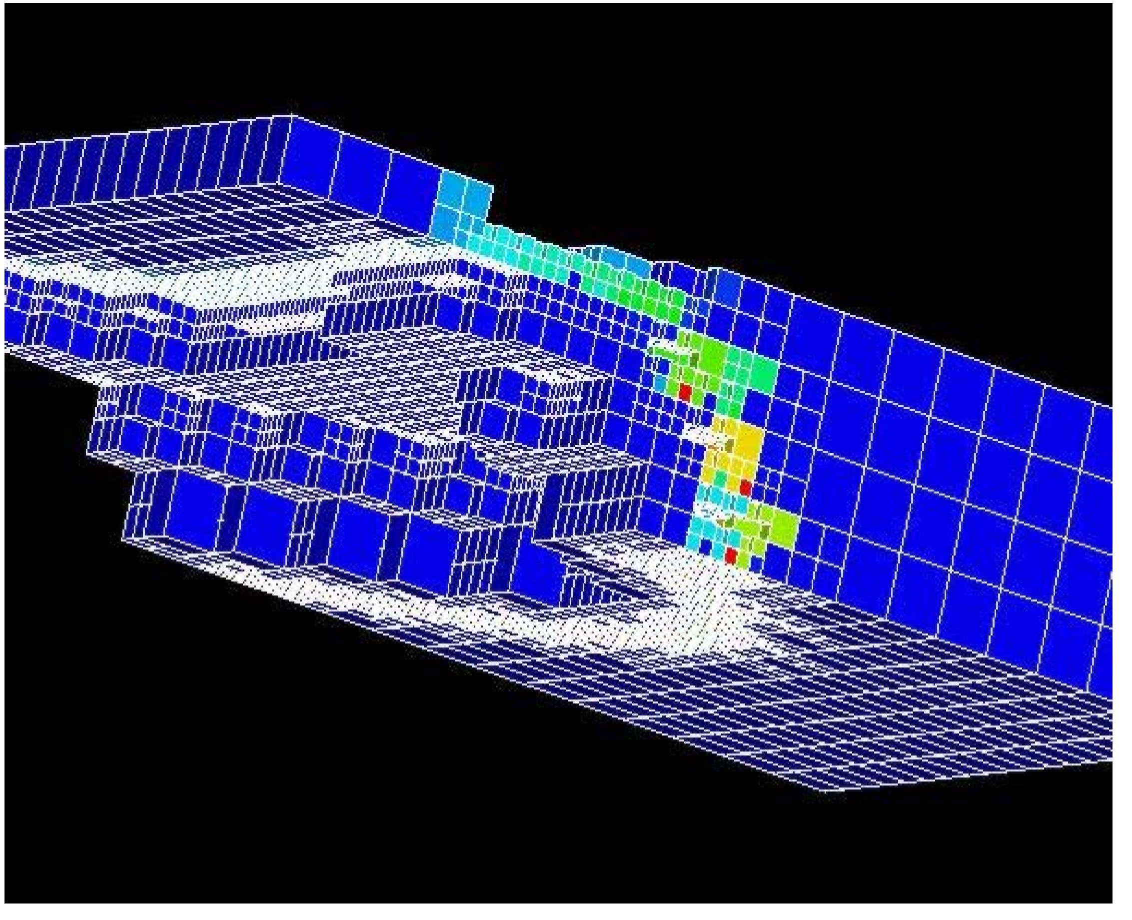
FEM Mesh (SW Japan Model)



Voxel Mesh (initial)

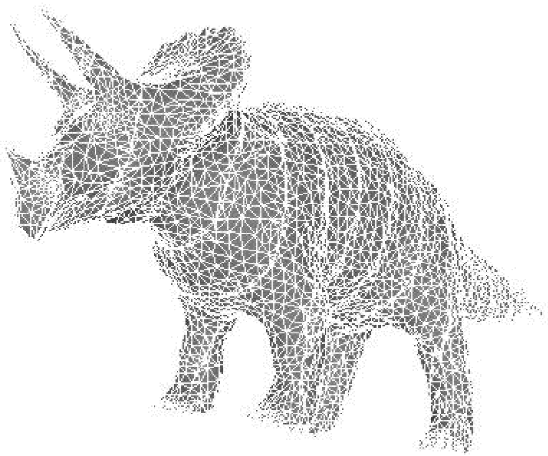


Voxel Mesh (2-level adapted)

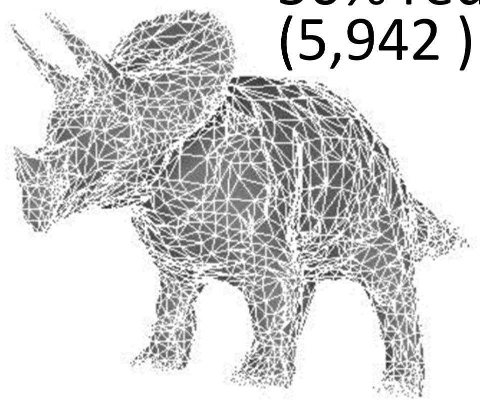


Example of Surface Simplification

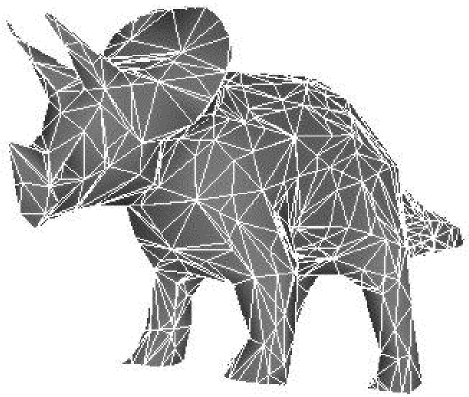
Initial
(11,884 tri's)



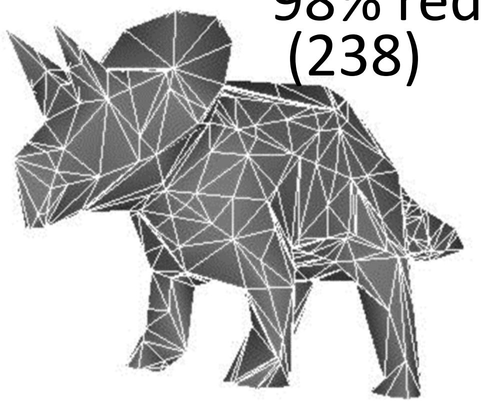
50% reduction
(5,942)



95% reduction
(594)



98% reduction
(238)



pFEM3D + ppOpen-MATH/VIS

Files

```
>$ cd <$O-TOP>
>$ cp /home/z30088/pVIS.tar .
>$ tar xvf pVIS.tar
```

FORTRAN

```
>$ cd <$O-TOP>/pVIS/F/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

C

```
>$ cd <$O-TOP>/pVIS/C/src
>$ make
>$ cd ../run
>$ pjsub go.sh
```

Makefile

```
CFLAGSL  = -I/home/z30088/ppohVIS_test/include
LDFLAGSL = -L/home/z30088/ppohVIS_test/lib
LIBSL    = -lppohvisfdm3d

.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
```

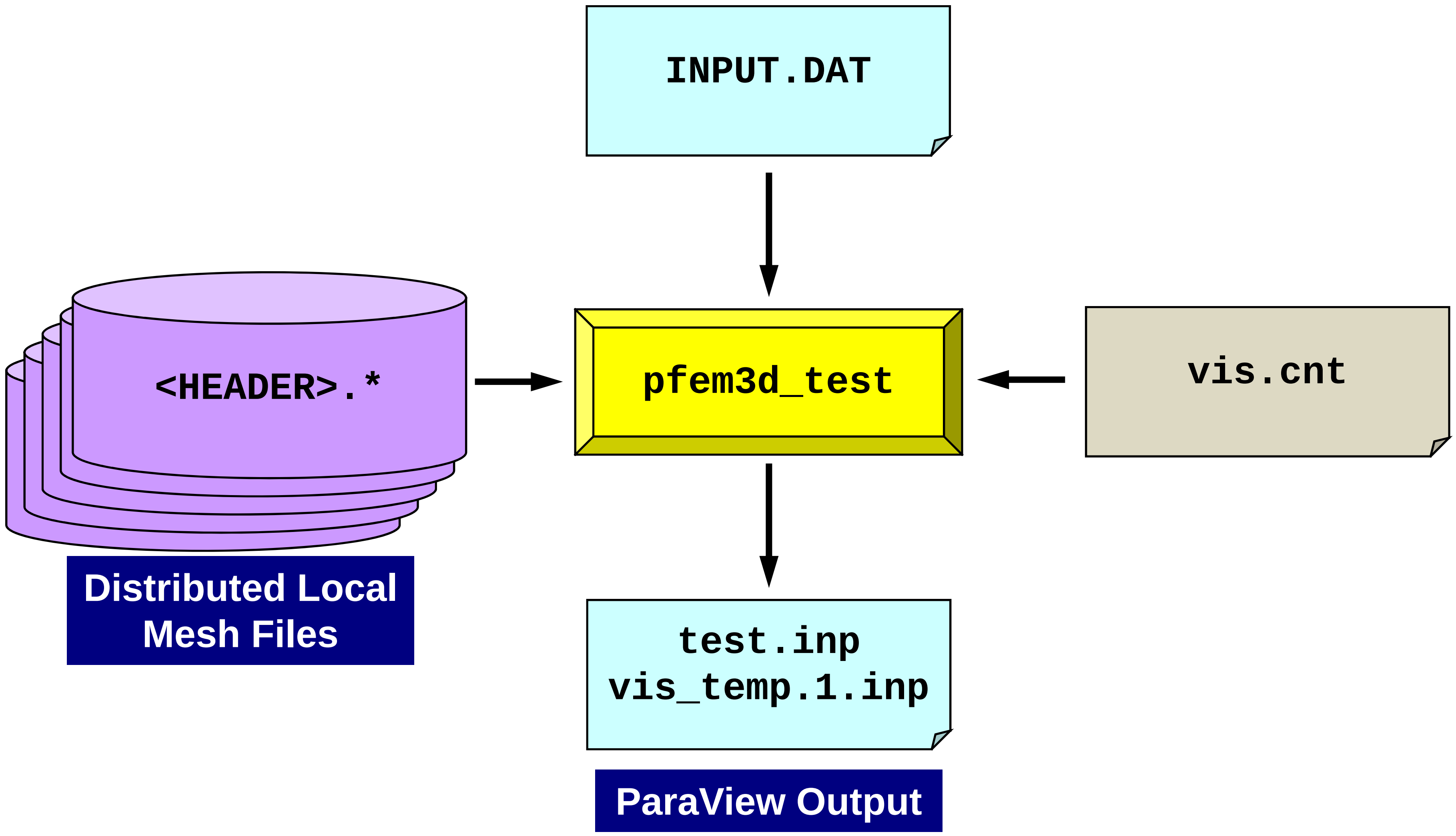
<$O-TOP>/pVIS/F(C)/run

```
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=lecture"
#PJM -L "node=4"
#PJM --mpi "proc=4"
#PJM -L "elapse=00:10:00"
#PJM -g "gt71"
#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_fdm3d_util

      implicit REAL*8(A-H,O-Z)
      type(ppohVIS_FDM3D_stControl)              :: pControl
      type(ppohVIS_FDM3D_stResultCollection)     :: pNodeResult
      type(ppohVIS_FDM3D_stResultCollection)     :: pElemResult
      character(len=PPOHVIS_FDM3D_FILE_NAME_LEN) :: CtrlName
      character(len=PPOHVIS_FDM3D_FILE_NAME_LEN) :: VisName
      character(len=PPOHVIS_FDM3D_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

 call ppohVIS_PFEM3D_ConvResult(N, ValLabel, X,                     &
&                               pNodeResult, pElemResult, iErr)
 call ppohVIS_PFEM3D_Visualize(pNodeResult, pElemResult, pControl, &
&                              VisName, 1, iErr)

 call ppohVIS_PFEM3D_Finalize(iErr)

 call PFEM_FINALIZE

 end program heat3Dp
```

C/main (1/2)

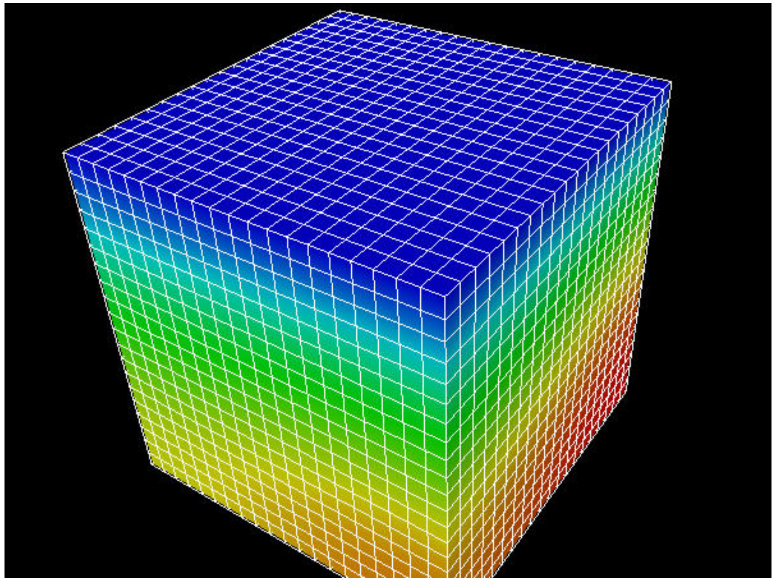
```
#include <stdio.h>
#include <stdlib.h>
FILE* fp_log;
#define GLOBAL_VALUE_DEFINE
#include "pfem_util.h"
#include "ppohVIS_FDM3D_Util.h"
extern void PFEM_INIT(int,char**);
extern void INPUT_CNTL();
extern void INPUT_GRID();
extern void MAT_CON0();
extern void MAT_CON1();
extern void MAT_ASS_MAIN();
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_FDM3D_PrintError(stderr);
  };
```

C/main (2/2)

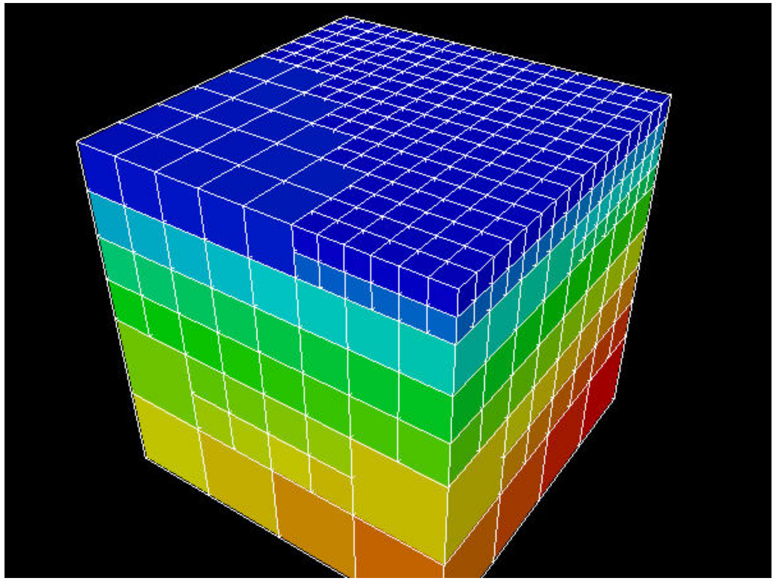
```
  MAT_CON0();
  MAT_CON1();

  MAT_ASS_MAIN();
  MAT_ASS_BC()  ;

  SOLVE11();

  OUTPUT_UCD();

  pNodeResult = ppohVIS_PFEM3D_ConvResult(N, "temp", X);

  if(ppohVIS_PFEM3D_Visualize(pNodeResult, NULL, pControl, "vis", 1)) {
    ppohVIS_FDM3D_PrintError(stderr);
  }

  ppohVIS_PFEM3D_Finalize();

  PFEM_FINALIZE() ;
}
```

vis.cnt

```
[Refine]                    Section for Refinement Control
AvailableMemory = 2.0        (GB) not in use
MaxVoxelCount = 1000        Max Voxel #
MaxRefineLevel = 20         Max Voxel Refinement Level
[Simple]                    Section for Simplification Control
ReductionRate = 0.0         Reduction Rate of Surf. Patches
```



8,000 elements, 10,334 nodes



813 elements, 1,236 nodes