

3D Parallel FEM (III) Parallel Visualization and ppOpen-HPC

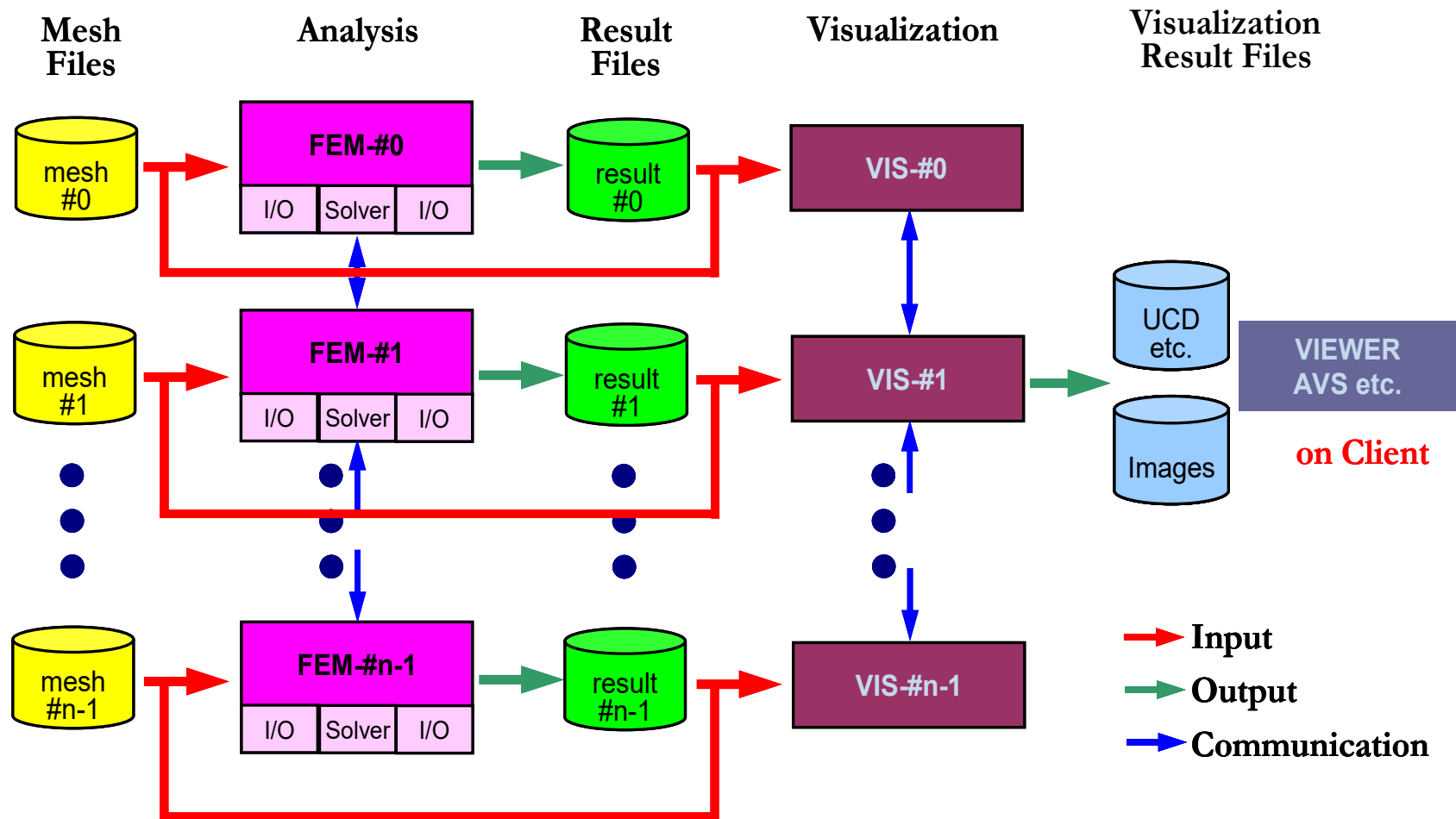
Kengo Nakajima

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

ppOpen-HPC

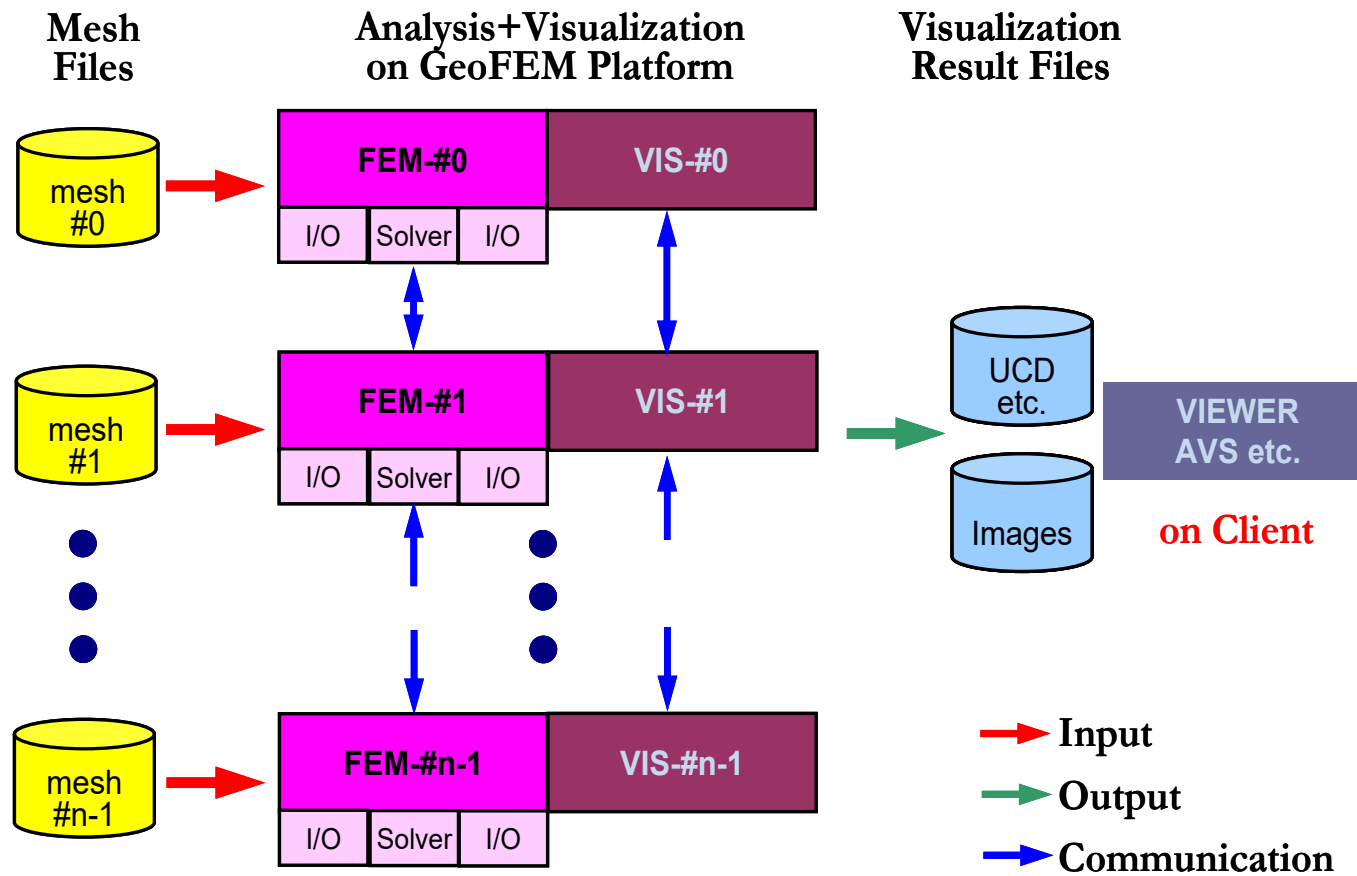
Framework for Parallel Visualization 1

Via-File

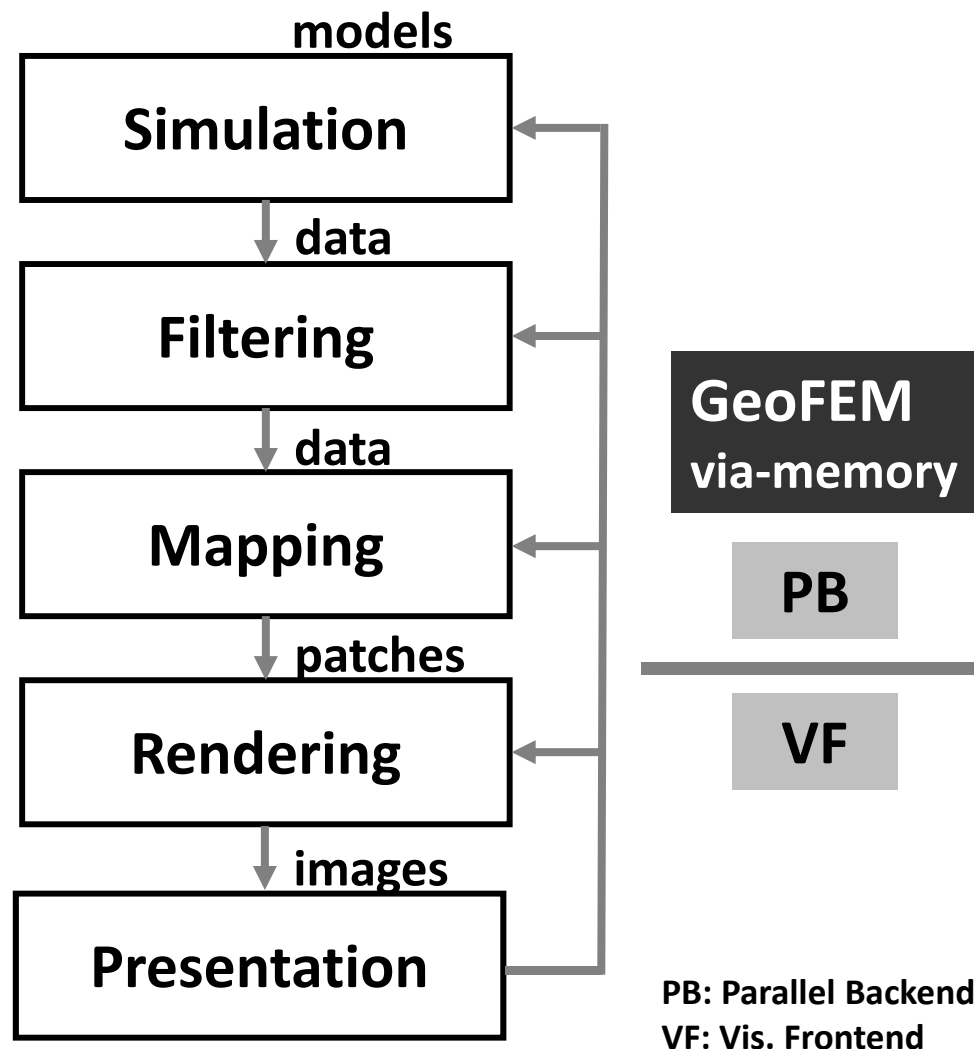


Framework for Parallel Visualization 2

Via-Memory (GeoFEM Project)

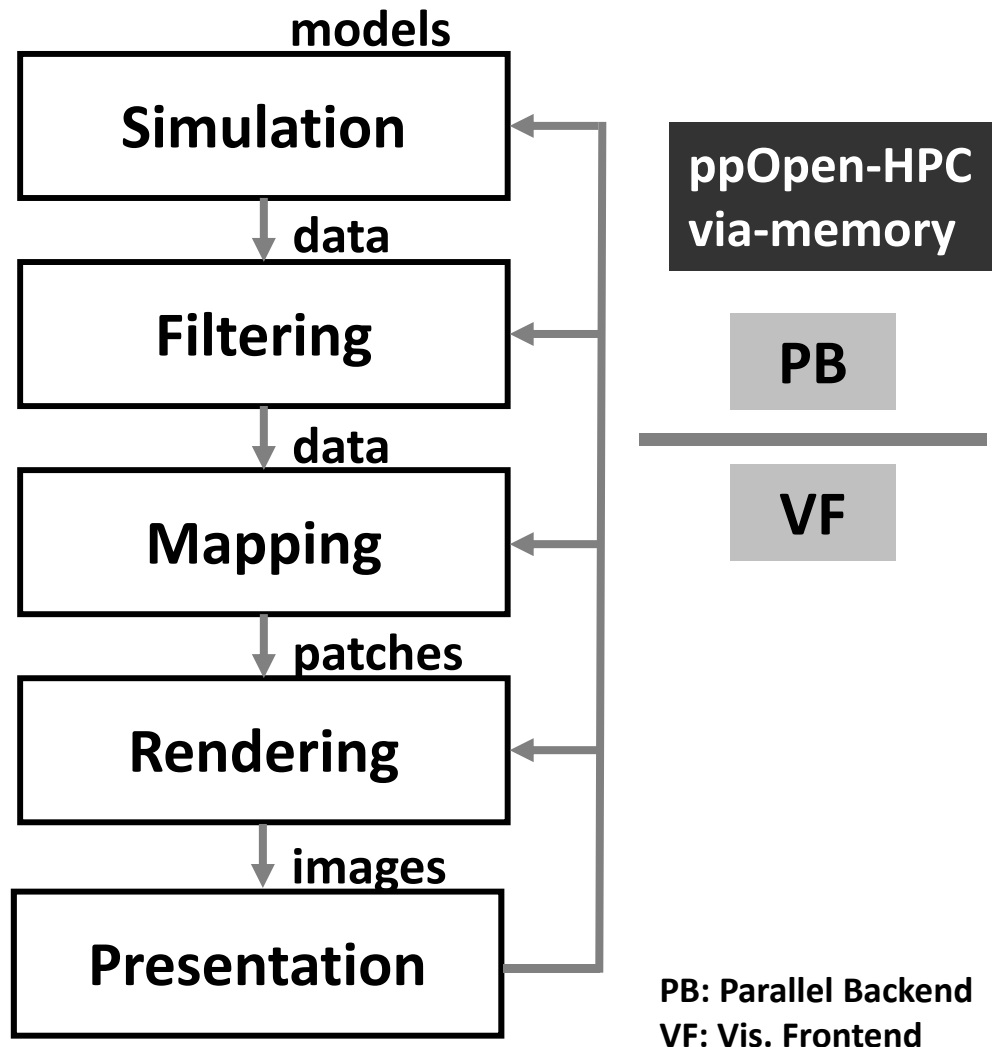


Visualization in ppOpen-HPC



- Concurrent Visualization-Computation
- Output files (single “self-contained(自己完結)” file) are browsed by MicroAVS & Paraview on a PC
- In GeoFEM (previous project), only patch files were obtained.
- Not detailed visualization.
- Just for understanding MIN-MAX, and peaks
- Detailed geometry is preferable

Visualization in ppOpen-HPC



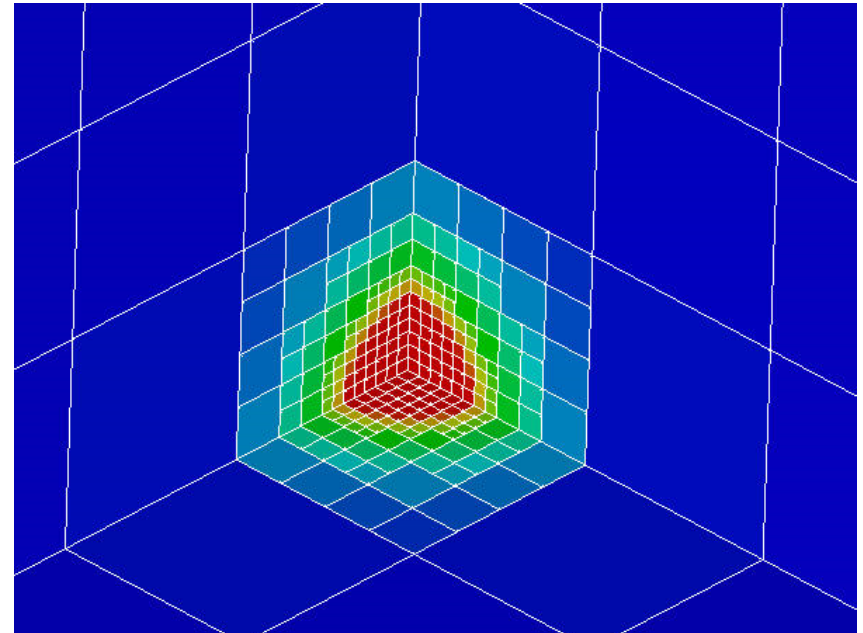
- Concurrent Visualization-Computation
- Output files (single “self-contained (自己完結)” file) are browsed by MicroAVS & Paraview on a PC
- In GeoFEM (previous project), only patch files were obtained.
- Not detailed visualization.
- Just for understanding MIN-MAX, and peaks
- Detailed geometry is preferable

Supercomputer System & Visualization

- Supercomputer is not for “displaying”, but for computing.
- “Re-calculation” for just visualization is not good.
- In supercomputer center, we spend money on computer itself (not for visualization system).

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

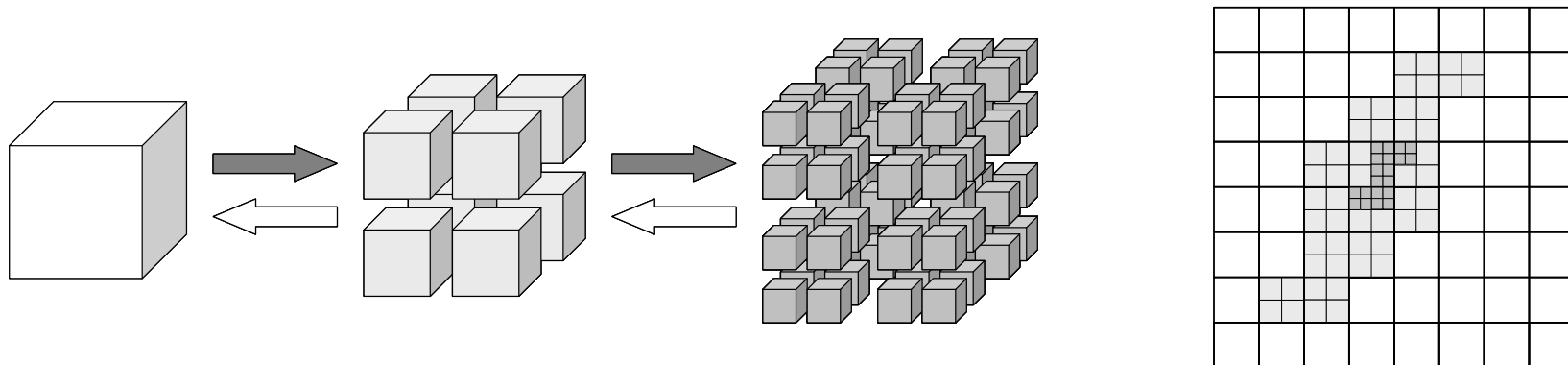
Maximum number of voxels

MaxRefineLevel = 20

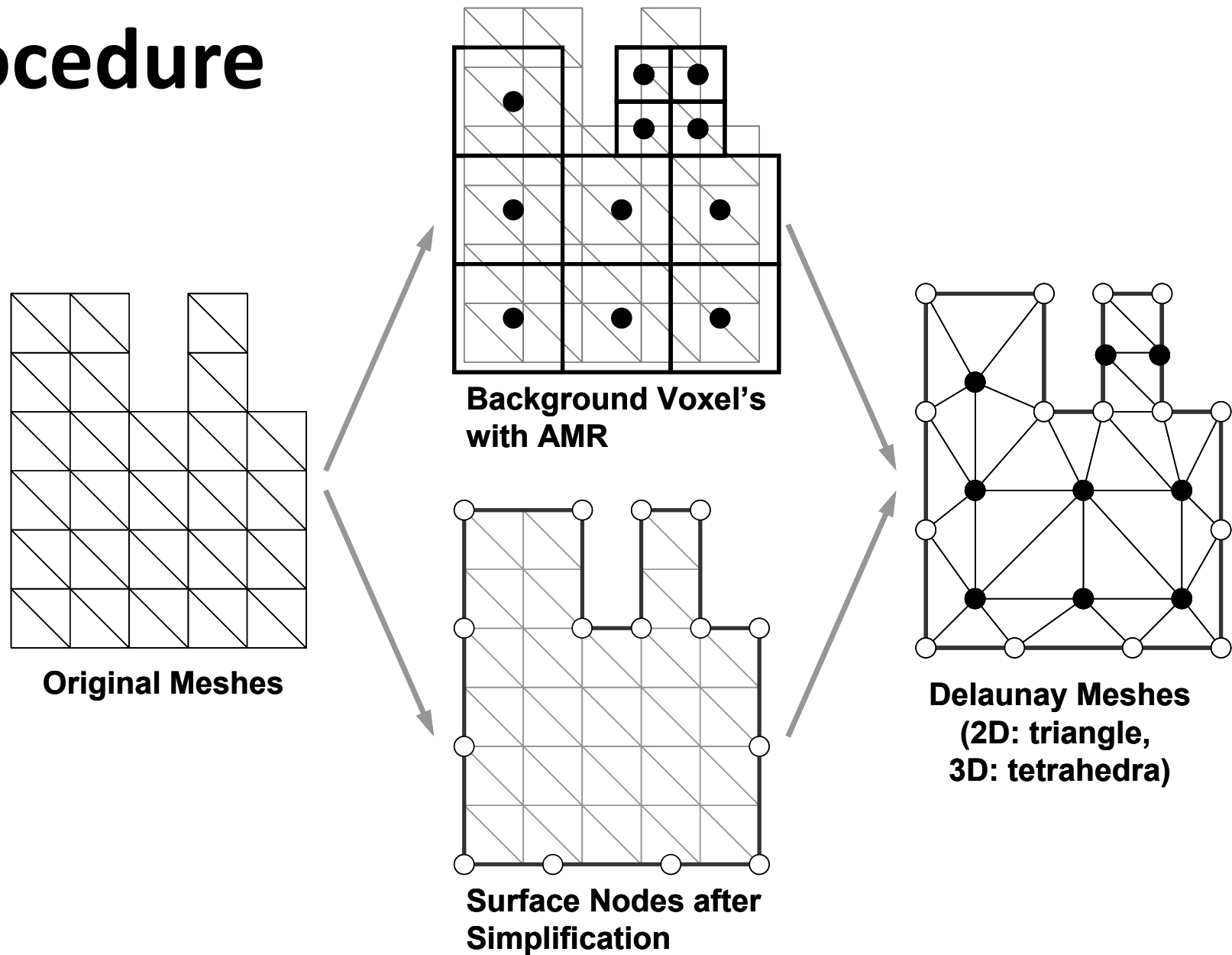
Maximum number of refinement levels

Simplified Parallel Visualization using Background Voxels

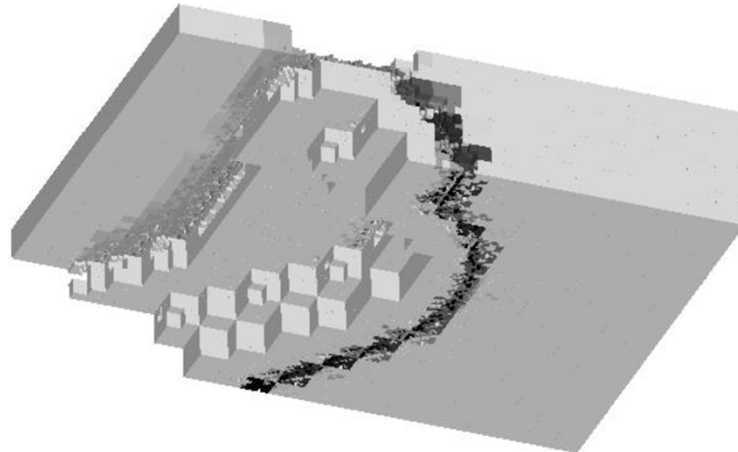
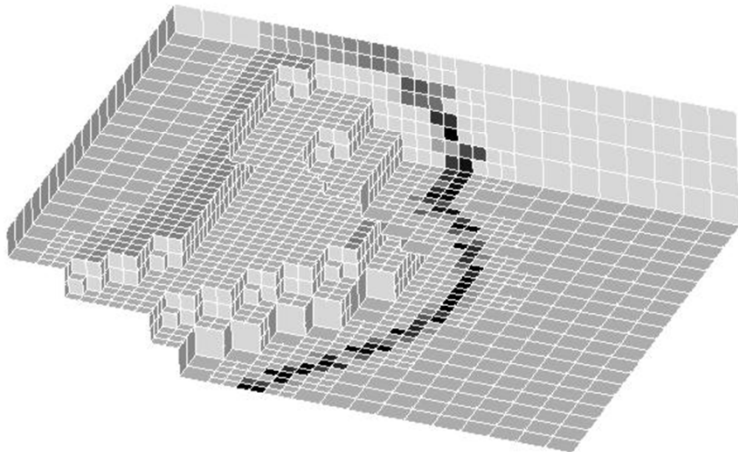
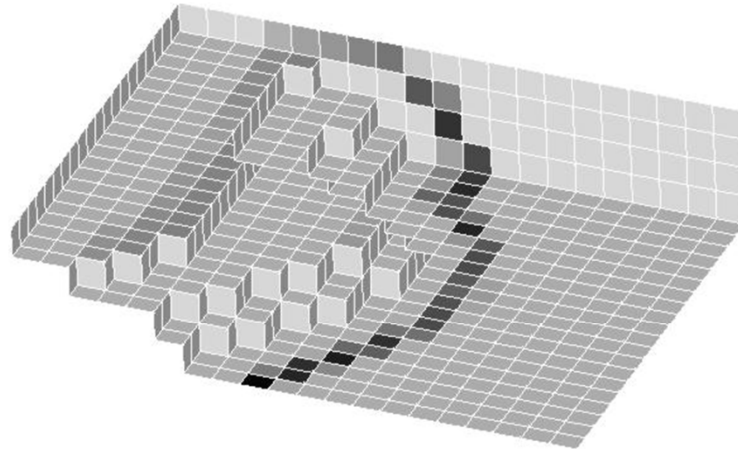
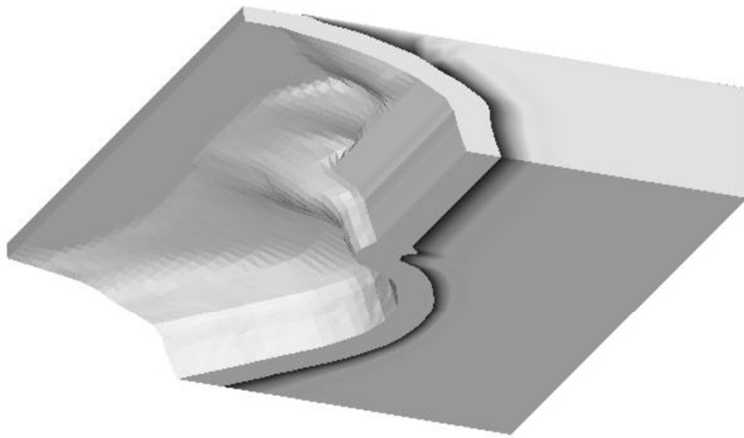
- Octree-based AMR
- AMR applied to the region where gradient of field values are large
 - stress concentration, shock wave, separation etc.
- If the number of voxels are controlled, a single file with 10^5 meshes is possible, even though entire problem size is 10^9 with distributed data sets.



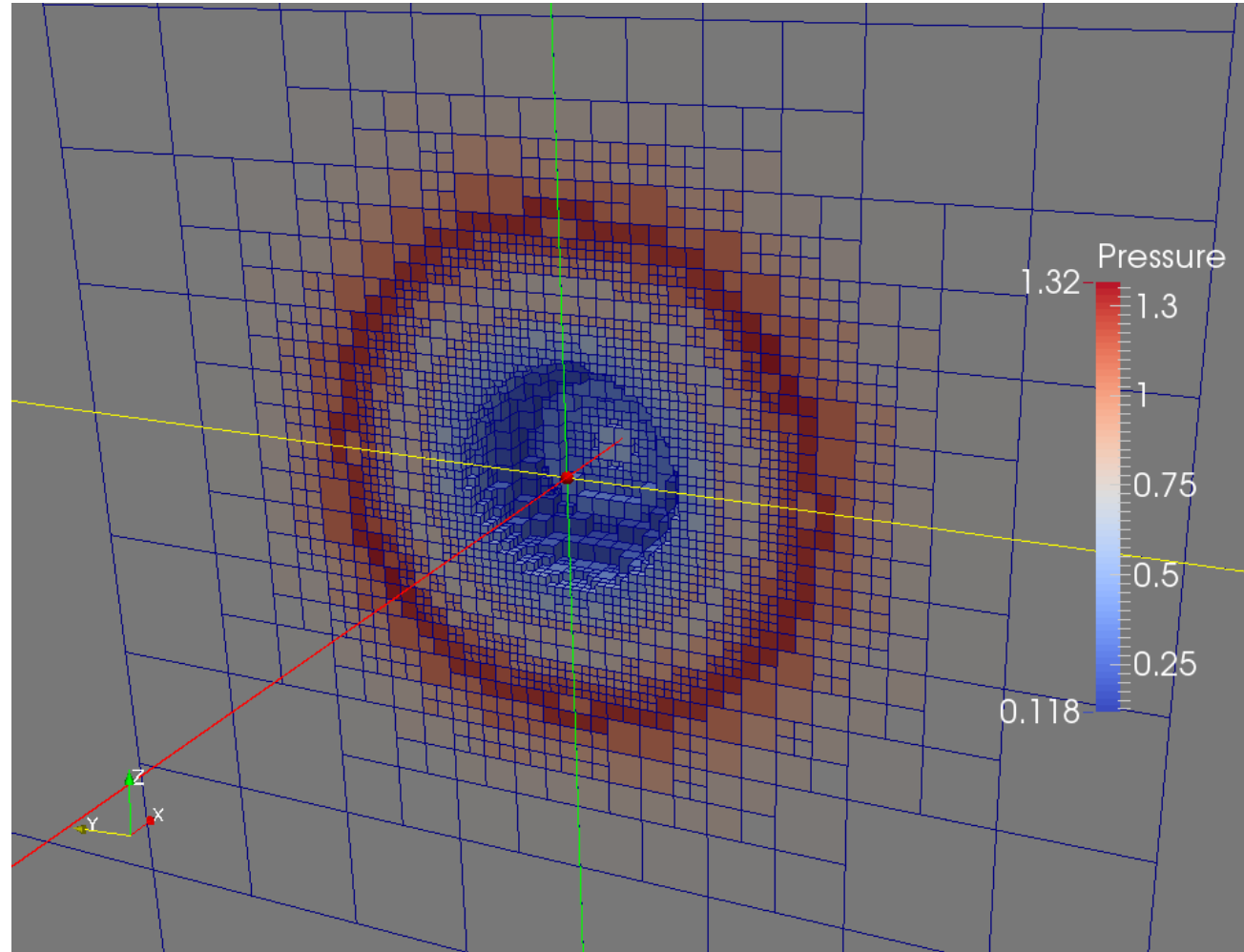
Procedure



Voxel Mesh (adapted)



Flow around a sphere



pFEM3D + ppOpen-MATH/VIS

```
>$ cd <$0-TOP>/pfem3d/srcV  
>$ make  
>$ cd ../run  
>$ ls solv  
solv
```

Makefile (Fortran)

```
include Makefile.in
```

```
FFLAGSL    = -I/home/z30088/class_eps/include  
FLDFLAGSL  = -L/home/z30088/class_eps/lib  
LIBSL      = -lfppohvispfem3d -lppohvispfem3d
```

```
.SUFFIXES:
```

```
.SUFFIXES: .o .f90 .f
```

```
.f.o:
```

```
$(FC) -c $(FFLAGS) $(FFLAGSL) $< -o $@
```

```
.f90.o:
```

```
$(F90) -c $(F90FLAGS) $(FFLAGSL) $< -o $@
```

```
TARGET = ../run/solv
```

```
OBJS = ¥
```

```
test1.o ...
```

```
all: $(TARGET)
```

```
$(TARGET): $(OBJS)
```

```
$(F90) -o $(TARGET) $(F90FLAGS) $(FFLAGSL) $(OBJS)
```

```
$(LDFLAGSL) $(LIBS) $(LIBSL) $(FLDFLAGSL)
```

Makefile(C)

```
include Makefile.in
```

```
CFLAGSL = -I/home/z30088/class_eps/include
```

```
LDFLAGSL = -L/home/z30088/class_eps/lib
```

```
LIBSL    = -lppohvispfem3d
```

```
.SUFFIXES:
```

```
.SUFFIXES: .o .c
```

```
.c.o:
```

```
$(CC) -c $(CFLAGS) $(CFLAGSL) $< -o $@
```

```
TARGET = ../run/solv
```

```
OBJS = ¥
```

```
test1.o ...
```

```
all: $(TARGET)
```

```
$(TARGET): $(OBJS)
```

```
$(CC) -o $(TARGET) $(CFLAGS) $(CFLAGSL) $(OBJS)
```

```
$(LDFLAGSL) $(LIBS) $(LIBSL)
```

```
rm -f *.o *.mod
```

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

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] =

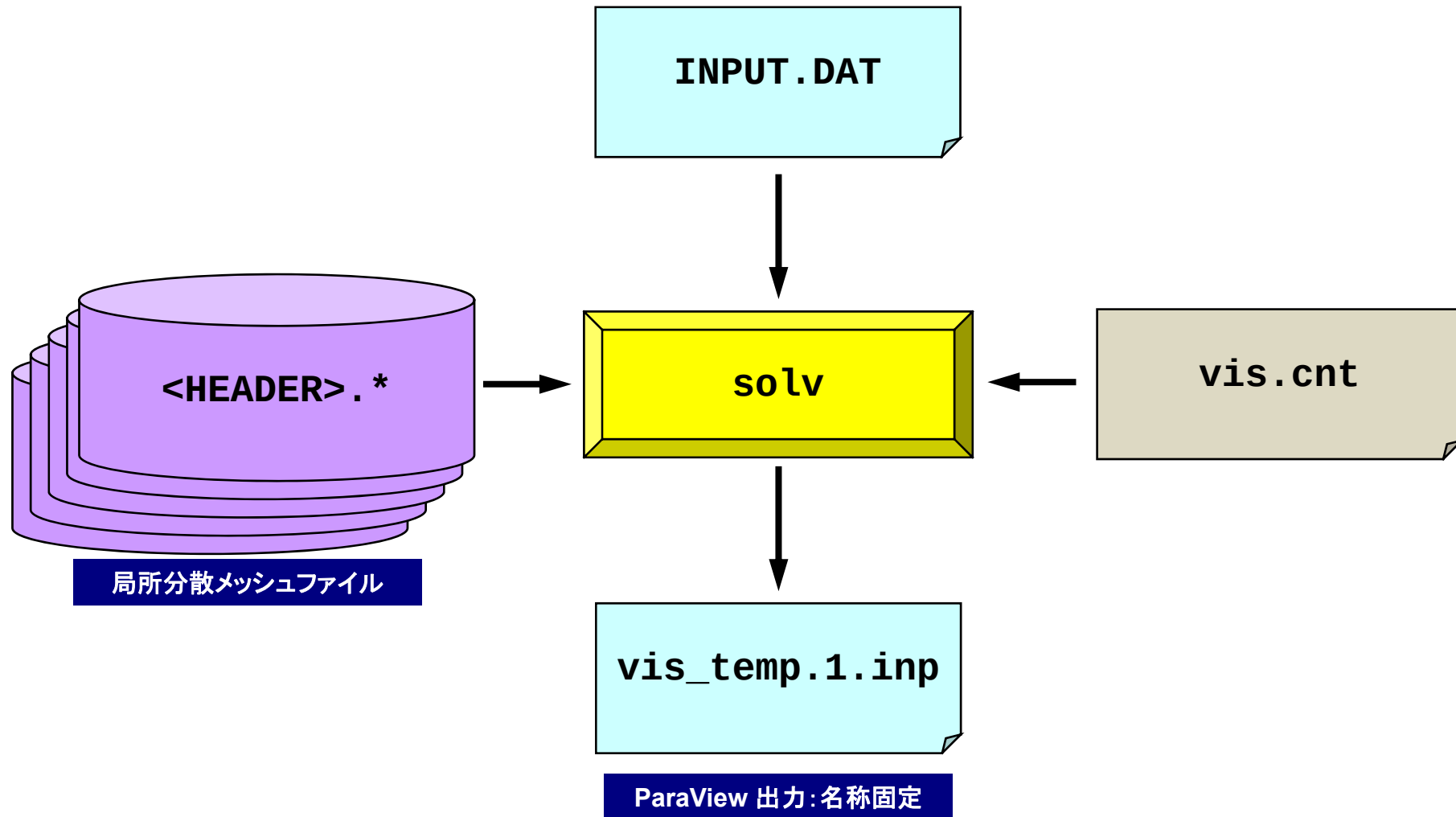
ppohVIS_PFEM3D_ConvResultNodeItemPart(NP,1,0,"temp",X);

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();
}
```

pFEM3D + ppOpen-MATH/VIS



Preparing Distributed Mesh Files

```
>$ cd <$0-TOP>/pfem3d/pmesh  
(mesh.inp, mg.sh)
```

```
>$ pjsub mg.sh
```

mesh.inp

```
256 256 256  
  4   4   4  
pcube
```

256^3 nodes into $4 \times 4 \times 4 = 64$ partitions

Each MPI process has 64^3 nodes

mg.sh

```
#!/bin/sh  
#PJM -L "node=4"  
#PJM -L "elapsed=00:05:00"  
#PJM -L "rscgrp=lecture3"  
#PJM -g "gt13"  
#PJM -j  
#PJM -o "mg.lst"  
#PJM --mpi "proc=64"  
  
mpiexec ./pmesh  
rm wk.*
```

Computation + Visualization

```
>$ cd <$0-TOP>/pfem3d/run  
(INPUT.DAT, gv.sh)
```

```
>$ pjsub gv.sh
```

INPUT.DAT

```
../pmesh/pcube  
2000  
1.0 1.0  
1.0e-08
```

gv.sh

```
#!/bin/sh  
#PJM -L "node=4"  
#PJM -L "elapsed=00:10:00"  
#PJM -L "rscgrp=lecture3"  
#PJM -g "gt13"  
#PJM -j  
#PJM -o "aa.lst"  
#PJM --mpi "proc=64"  
  
mpiexec ./solv
```

vis.cnt

[Refine]

AvailableMemory = 2.0

MaxVoxelCount = 1000

MaxRefineLevel = 20

[Simple]

ReductionRate = 0.0

[Output]

FileFormat = 2

Control Info. for Refinement

Available Memory (GB) not in use

Max Voxel #

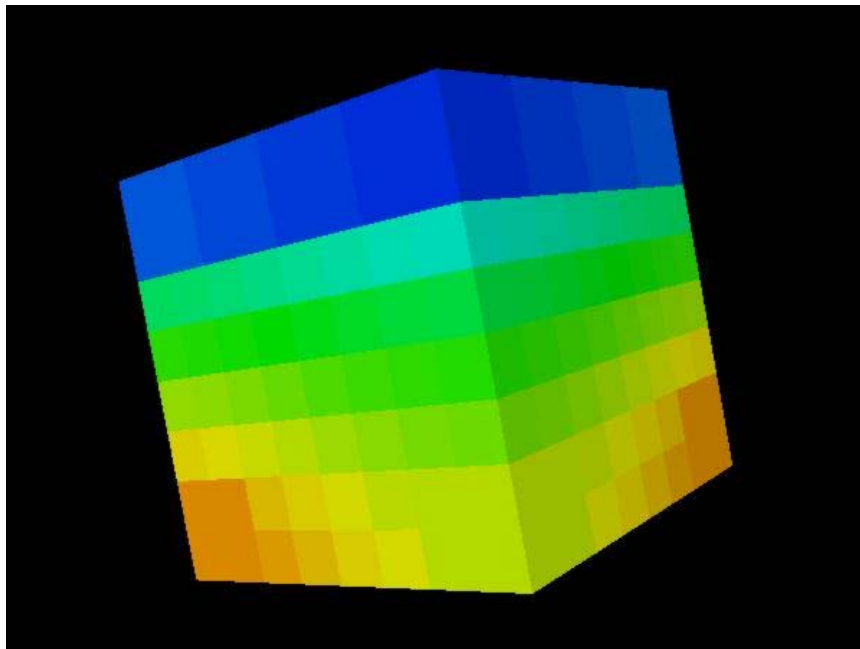
Max Voxel Refinement Level

Control Info. for Simplification

Reduction Rate of Surface Patches

Output Format

=1:MicroAVS, =2:ParaView



Values at Cell Ctr.

16,777,216 nodes

16,581,375 elem's, 64 MPI proc's



vis_temp.1.inp

1,436 nodes

1,002 elements