東京大学情報基盤センター
第45回 FX10 スーパーコンピュータシステム
お試しアカウント付き並列プログラミング講習会

「OpenFOAM初級入門」

今野　雅
(株式会社OCAEL・東京大学客員研究員)

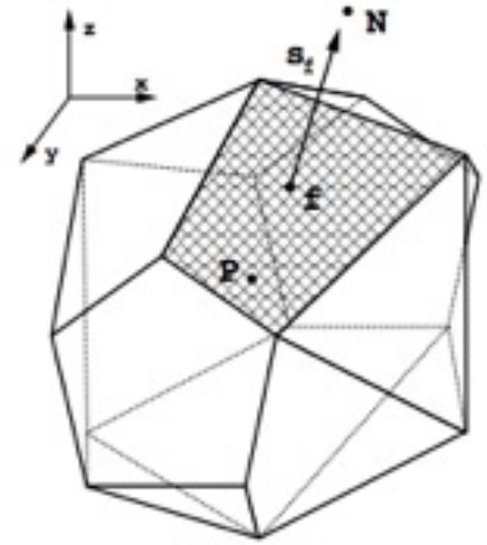
講習会プログラム

1. OpenFOAM概要
2. キャビティ流れ演習
3. ダムブレイク流れ，並列計算演習
4. チャネル流れ演習，実行性能・並列化効率評価

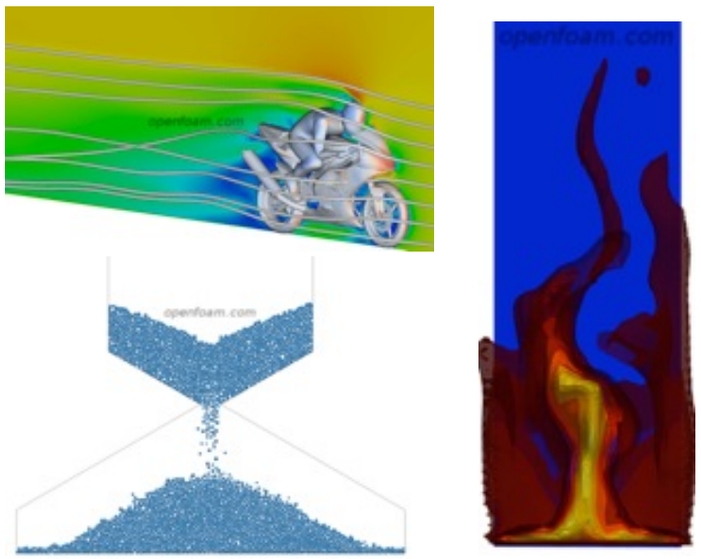
OpenFOAM概要

OpenFOAMの概要

図出典：The OpenFOAM Foundation



FVM
ポリヘドラル

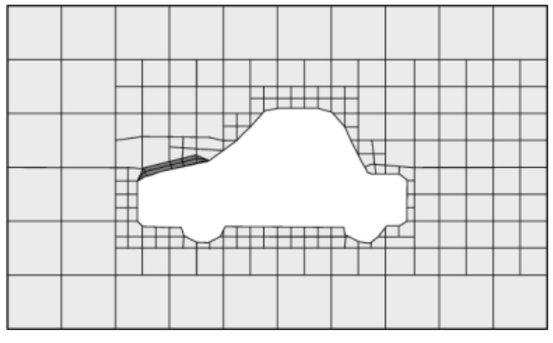


マルチフィジックス

$$\frac{\partial T}{\partial t} + \nabla \bullet \phi T - \nabla \bullet (\alpha T) = S_T$$

```
solve(fvm::ddt(T)
    + fvm::div(phi,T)
    - fvm::laplacian(DT,T)
   == fvOptions(T));
```

C++



境界適合Hex
メッシャー

乱流モデル:
RAS, LES, DES, …
線型ソルバー :
AMG, PCG, PBiCG, …
離散化スキーム: …

多数のモデル実装済

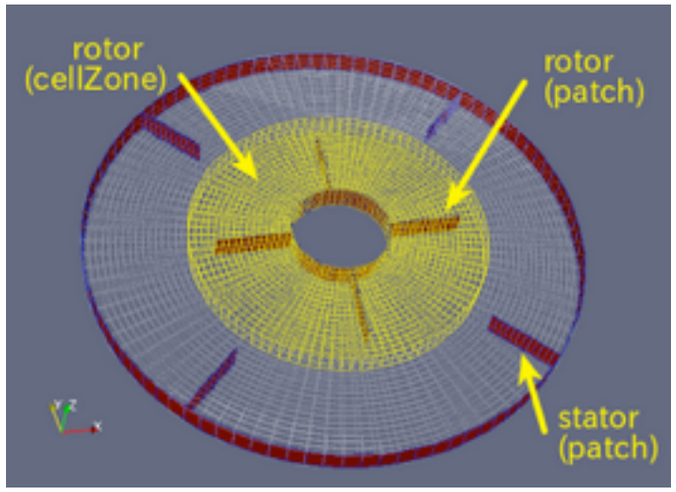
GPL
Open Source

カスタマイズ可能
低コストな超並列計算

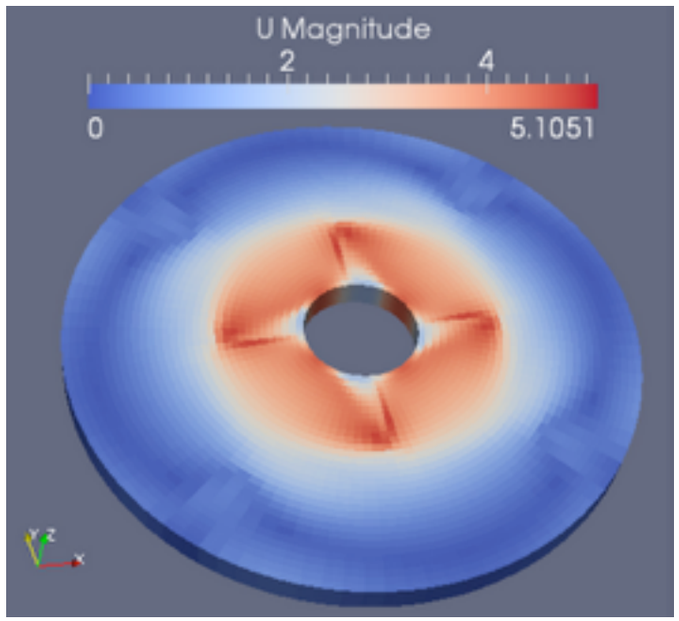
OpenFOAMの歴史

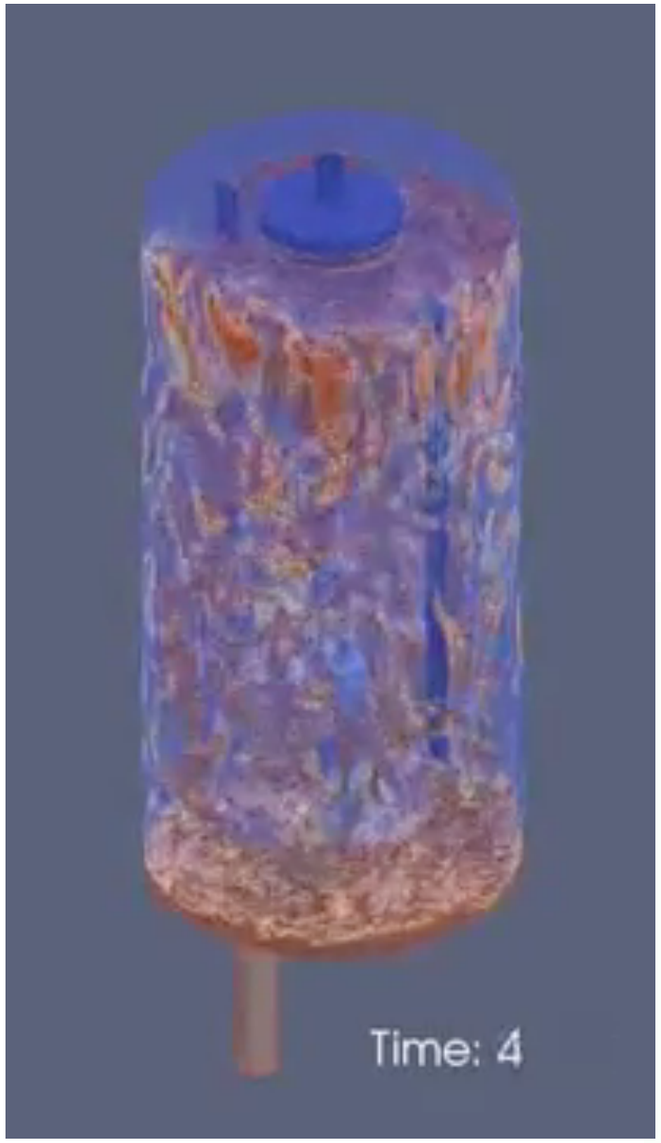
- 1989年ー2000年：**研究室のFORTRANコード時代**
  (GeCo+GUISE)、開発元：英Imperial CollegeのGosman研(Star-CDの開発元)の Henry Weller, Charlie Hill(GUI担当、→Apple→IBM)
- 1993年夏：**事故により全コード消失。**C++で書き直し(FOAM)
- 1999年ー2004年：**商用コード期 (FOAM)** Field Operation And Manipulationの略、開発元：∇Nabla(Henry, Hrvoje Jasak, Mattijs Janssensら)、代理店：CAEソリューションズ(フルイドテクノロジー)
- 2004年12月：**オープンソース化 (現在のOpenFOAMに名称変更)**、開発元：OpenCFD(Henry, Mattijs, Chris Greenshields)
- 2011年8月15日：**SGIによる買収**、GPL下のソースの管理や配布は、同時に設立されたThe OpenFOAM® Foundationが運用
- 2012年9月12日：**ESIによる買収**、Foundationによる運用は継続

標準ソルバの解析例

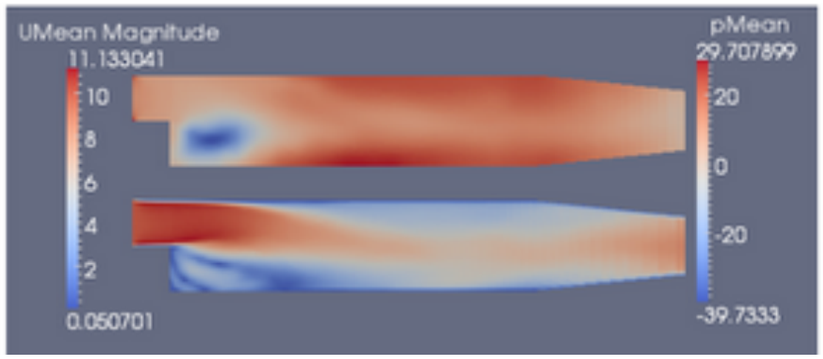


MRFSimpleFoam
回転攪拌槽の流れ



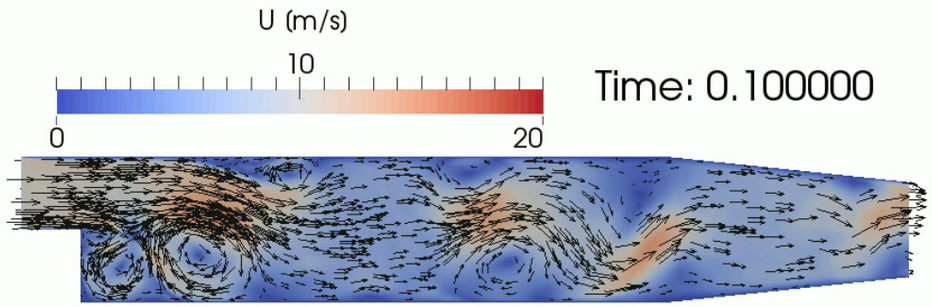


interDyMFoam
撹拌槽内の流れ



時間平均
圧力(p)

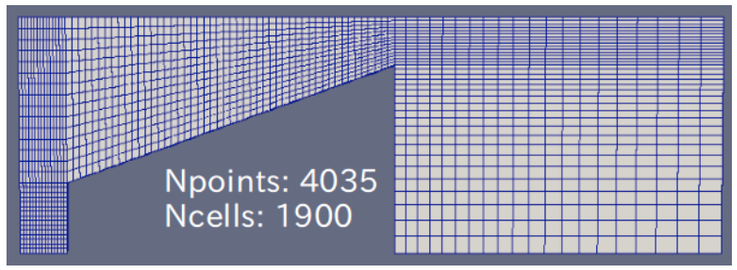
時間平均
速度(U)

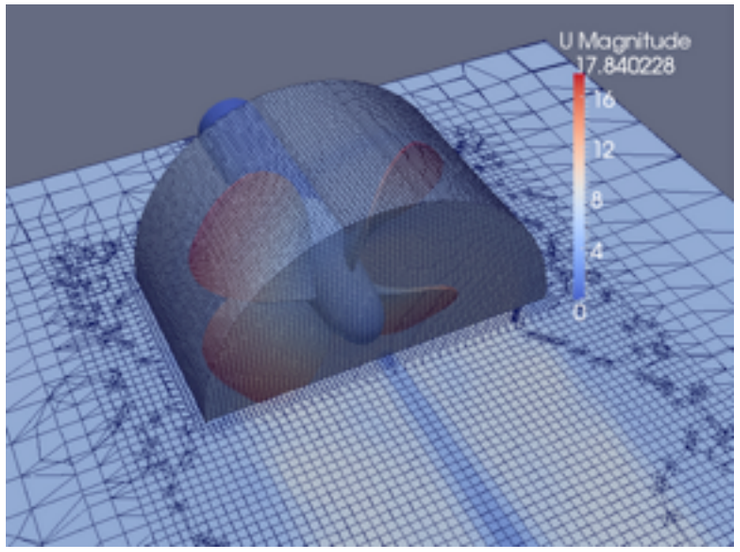


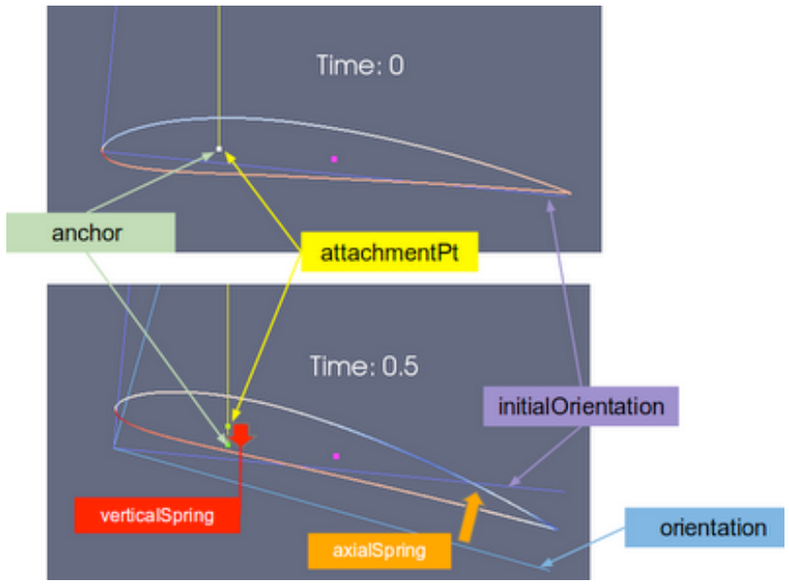
pisoFoam
LESによるバックステップ流れ

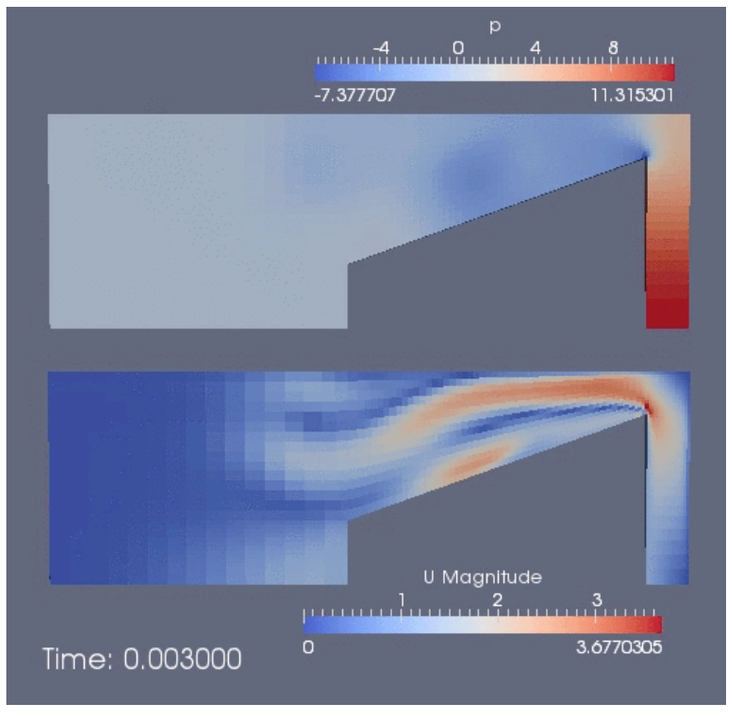
図出典：OpenFOAMチュートリアルドキュメント作成プロジェクト

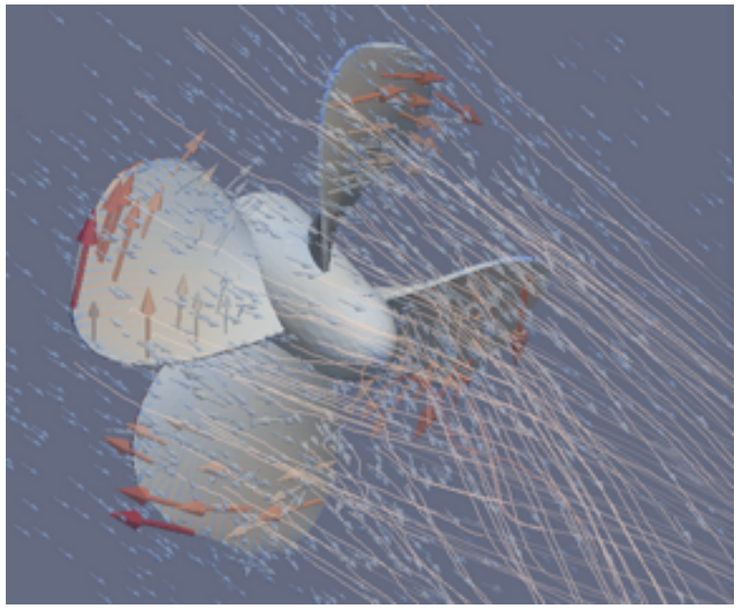
移動格子ソルバの解析例

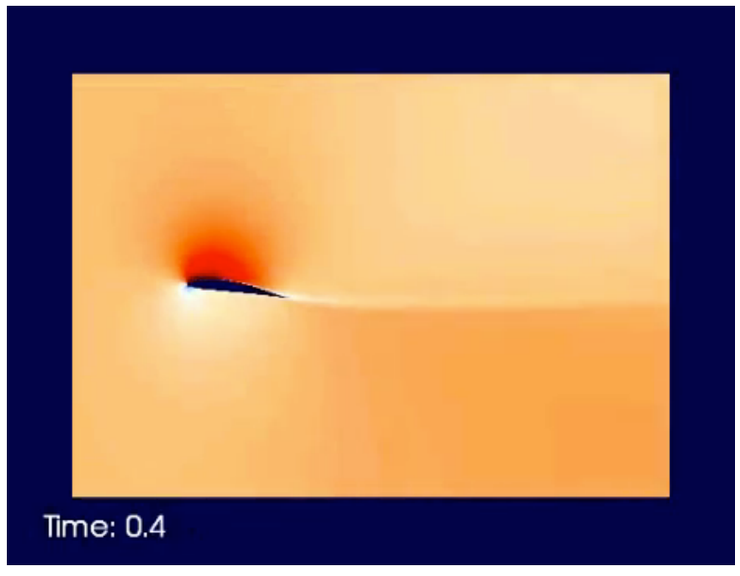








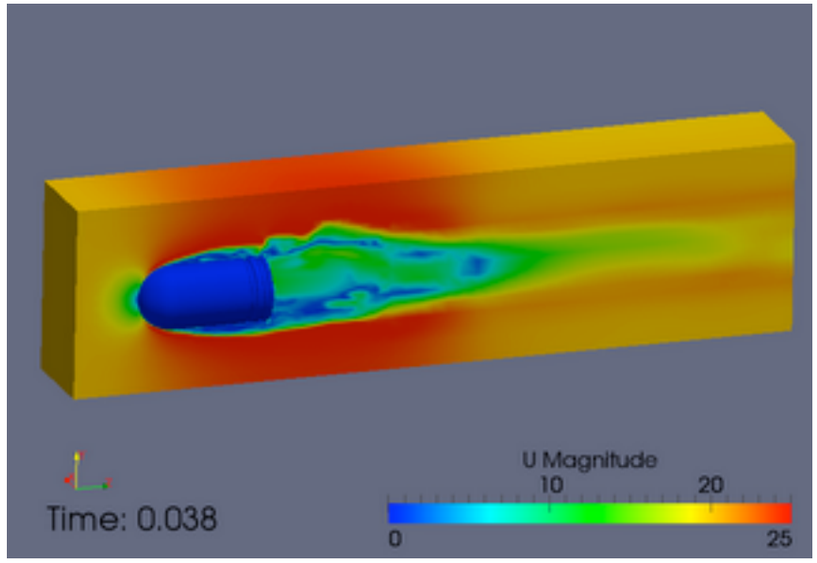




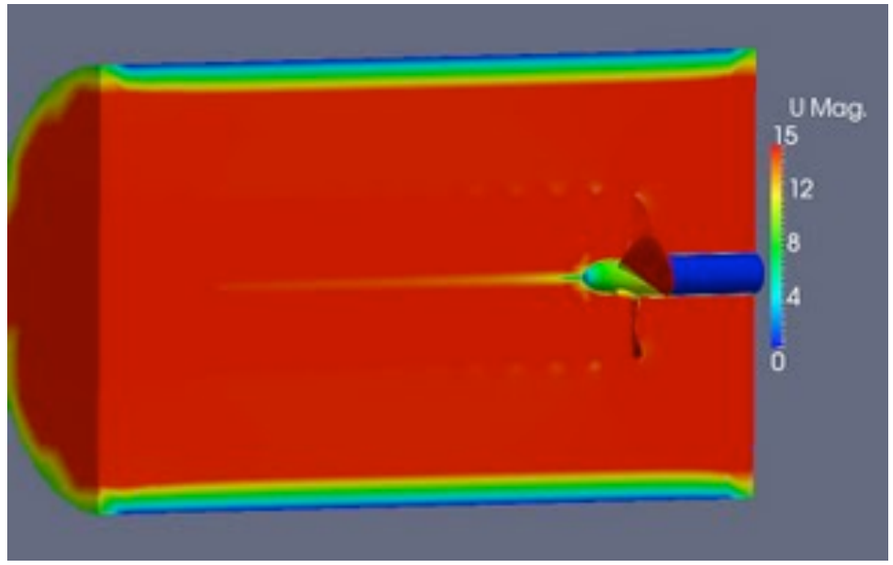
ピストン押し込み流れ　　スクリューの回転流れ場　　翼型の6自由度剛体運動

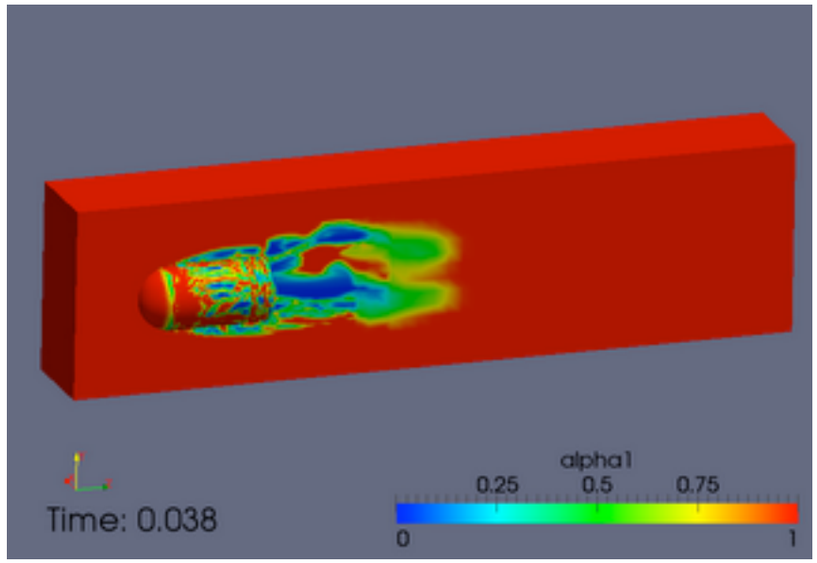
図出典：OpenFOAMチュートリアルドキュメント作成プロジェクト

相変化ソルバの解析例

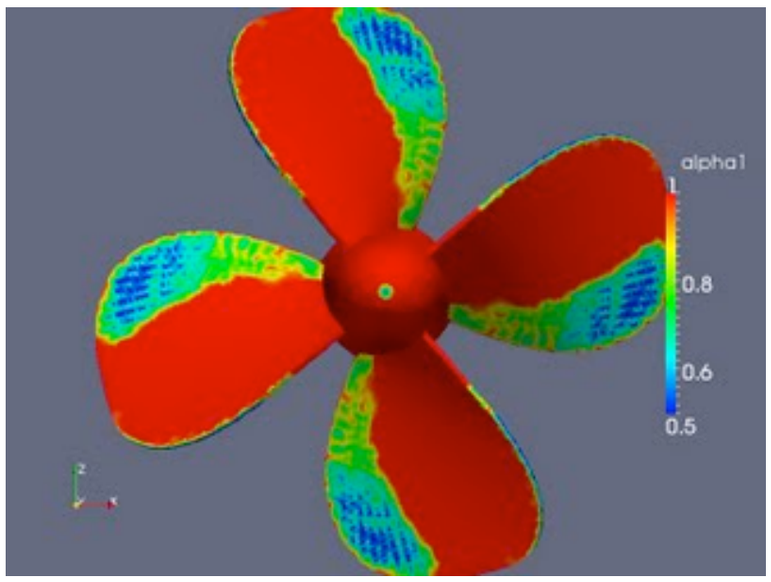


速度





相率



interPhaseChangeFoam
弾丸周りのキャビテーション

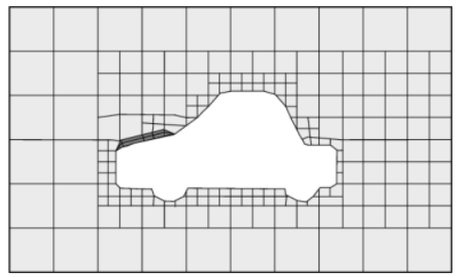
interPhaseChangeDyMFoam
プロペラ周りのキャビテーション

図出典：OpenFOAMチュートリアルドキュメント作成プロジェクト

OpenFOAMの最近の開発状況

図出典：The OpenFOAM Foundation

- ✓2008年07月28日 v.1.5 (snappyHexMesh)
- ✓2009年07月28日 v.1.6
- ✓2010年06月25日 v.1.7.0
- ✓2011年06月16日 v.2.0.0



1年毎
リリース

- ✓2011年08月15日 SGIによるOpenCFD買収
- ✓2011年12月11日 v.2.1.0
- ✓2012年06月16日 v.2.1.1

半年毎
マイナーリリース

- ✓2012年09月12日 ESIによるOpenCFD買収
- ✓2013年03月06日 v.2.2.0
- ✓2013年07月11日 v.2.2.1
- ✓2013年10月14日 v.2.2.2
- ✓2014年02月17日 v.2.3.0

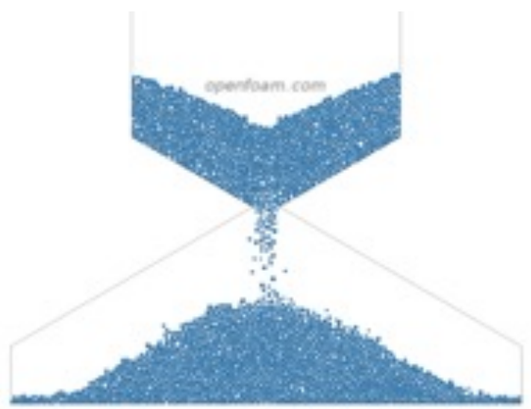
3〜4ヶ月年毎
マイナーリリース
1年毎
メジャーリリース

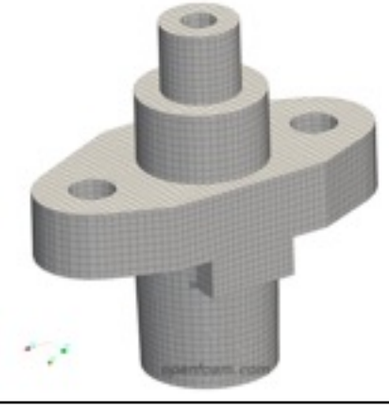
- ✓2014年12月10日 v.2.3.1 (最新版.　Henry, ChrisがFoundationに移行)

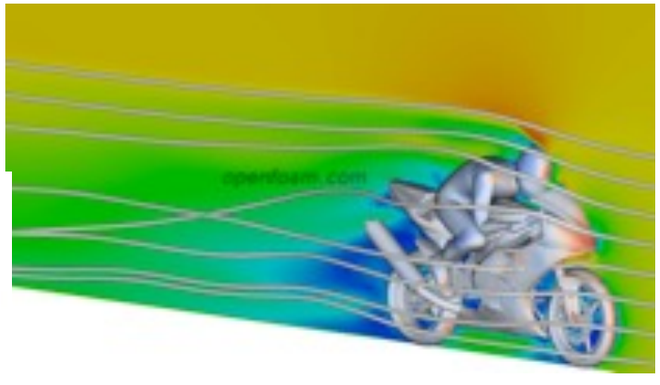
OpenFOAM 2.0.0 (2011/6)の新機能

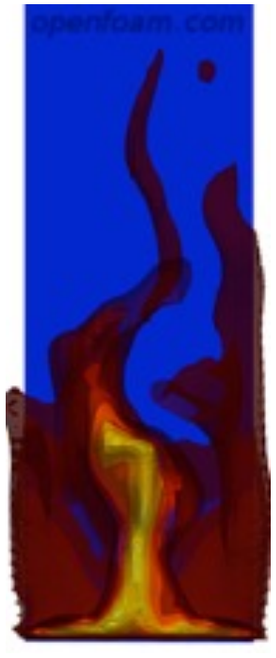
図出典：The OpenFOAM Foundation

- 表面フィルムモデル
- 定常VOF(ローカル・タイム・ステッピング)
- ラグラジアンモデル (離散要素モデル、粒子追跡)
- 熱物理モデル(形態係数放射、薄板・ポーラスの熱解析)
- 化学機能(火災解析用の熱分解モデル、化学反応ソルバ)
- 乱流モデル(ダイナミック・ラグラジアンLES)
- ポスト処理(流線等のポスト処理関数)
- 実行時制御 (実行時のC++コード実行)
- メッシュ生成(特徴辺の再現)







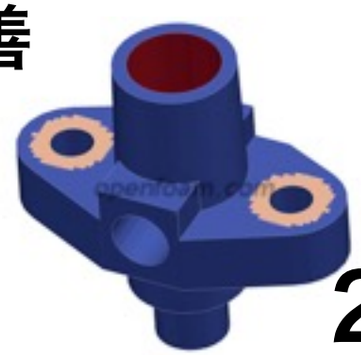


OpenFOAM 2.2.0 (2013/3) の新機能

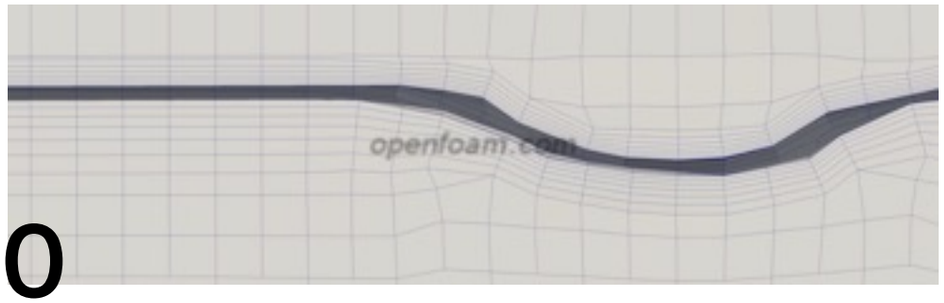
図出典：The OpenFOAM Foundation

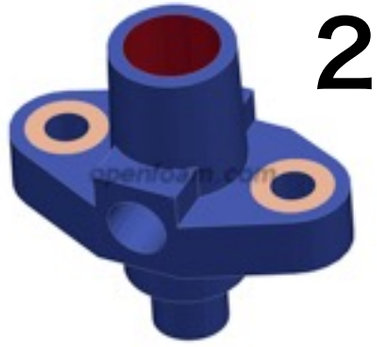
- 境界適合六面体メッシャーの改善

(パッチの特徴辺、レイヤー改善)

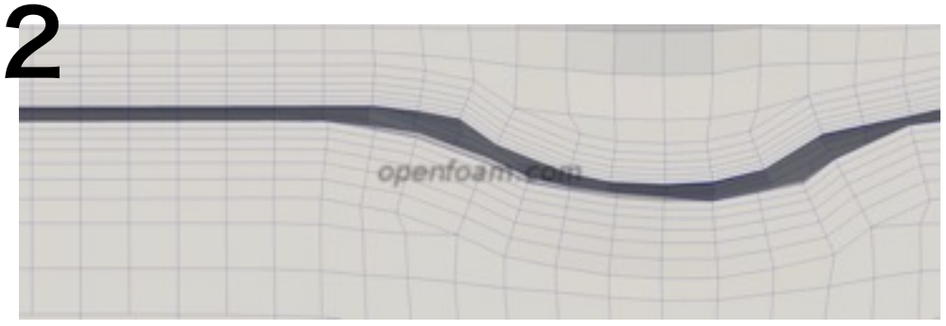


2.0

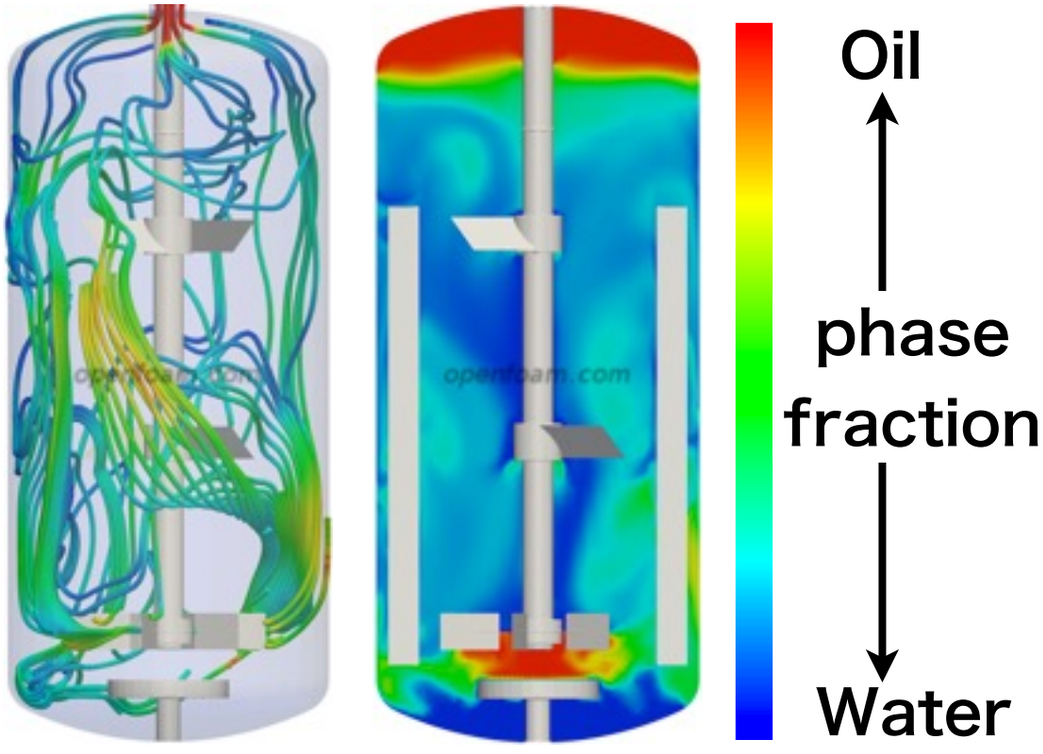




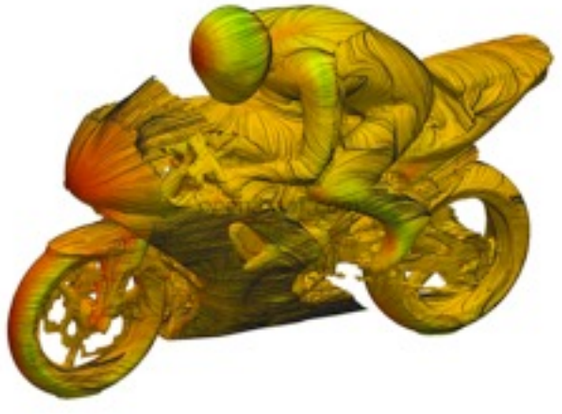
2.2

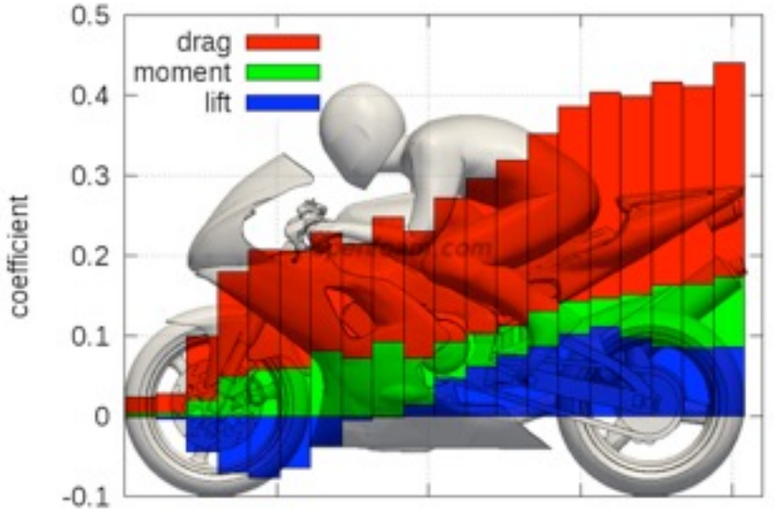


- Vector-Coupled Solver
- 多相流での熱力学モデル改良



- 実行時ポスト処理の機能向上

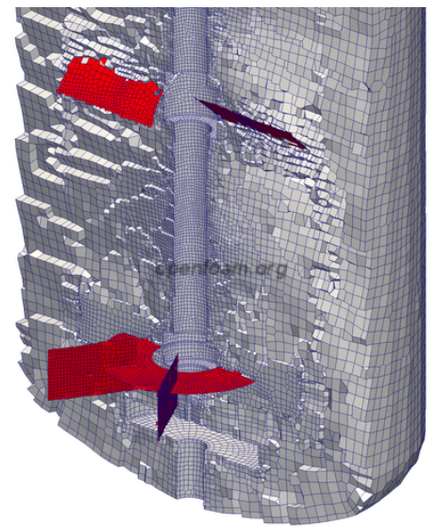




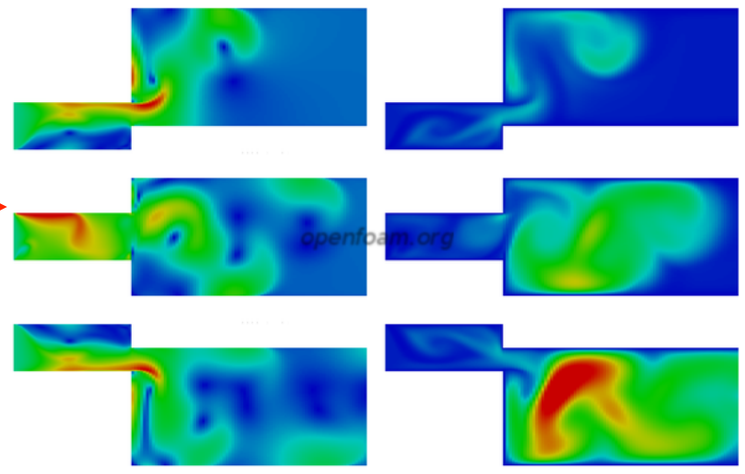
OpenFOAM 2.3.0 (2014/2) の新機能

図出典：The OpenFOAM Foundation

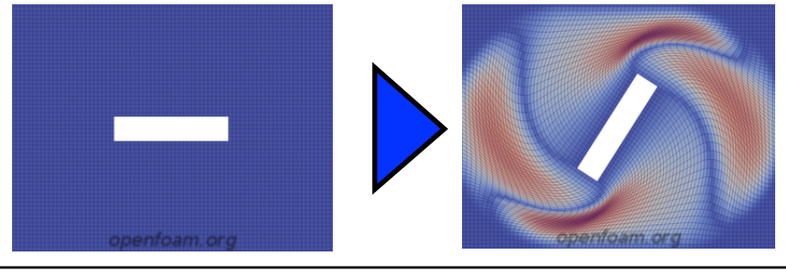
- 境界適合六面体メッシャーsnappyHexMeshの改善
- 新自動六面体メッシャーfoamyHexMesh

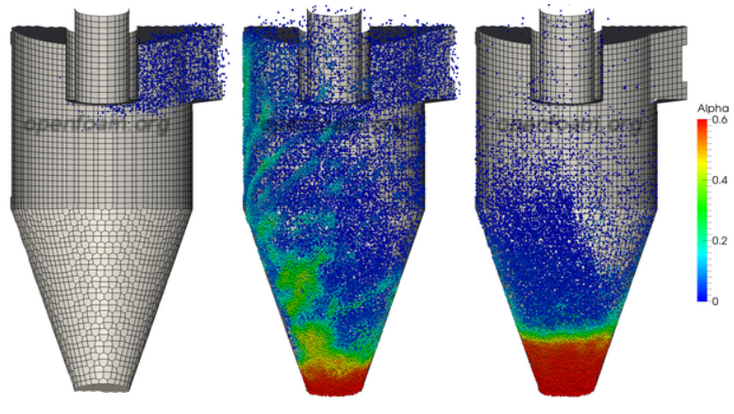


- 並列計算時の線型ソルバGAMGの高速化
- VOF法多相流ソルバにおけるMulti-dimensional limiter for explicit solution(MULES)の半陰解版



- Arbitary Mesh Interface(AMI)の改善・拡張
- 密粒子流れ用のDiscrete Particle Modelling, MultiPhase Particle-in-Cell法
- 移動格子での球面線型補間を用いたモーフィング





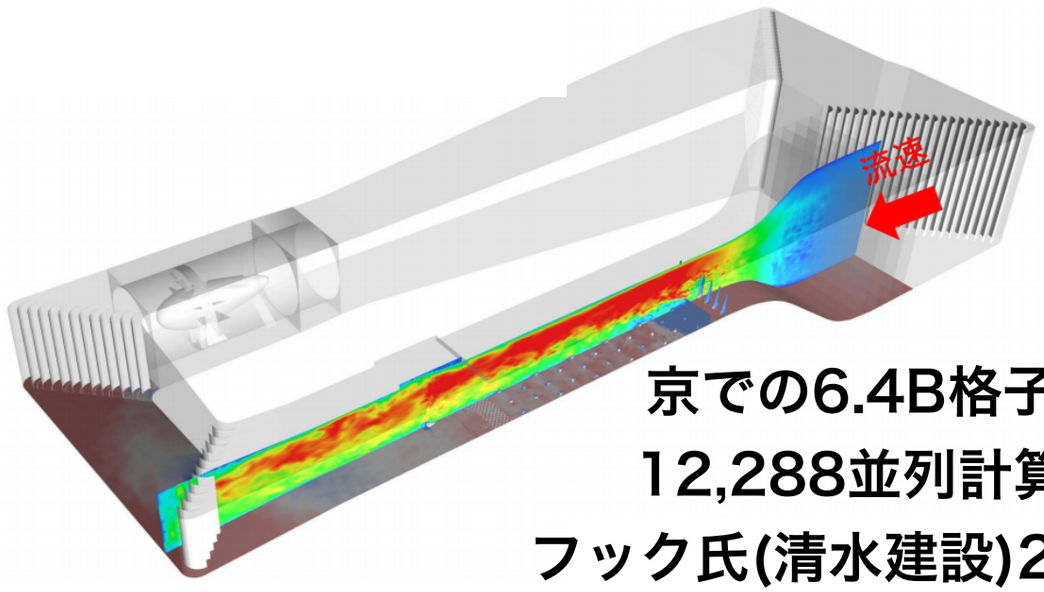
OpenFOAMの稼働環境

Linux, Mac, Windows機で動作

Amazon AWS
IBM SoftLayer
Microsoft Azure
FOCUS，Etc.

東大FX-10ではインストール済み
京ではRISTが京ユーザ向けに最適化を支援

Laptop　PC　クラスタ　クラウド　スーパーコンピュータ　京
～1M格子　～10M格子　～100M格子　～10B格子



京での6.4B格子
12,288並列計算
フック氏(清水建設)2013

K computer

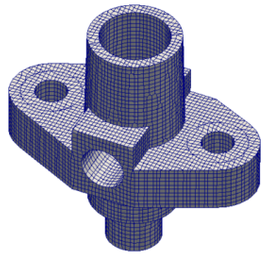
図出典：ファンバンフック「京」コンピュータにおけるOpenFOAMの大規模数値流体計算による建築物風圧予測，OpenFOAMワークショップ,2013

キャビティ流れ演習

OpenFOAMの代表的な解析手順

前処理(格子生成など)

格子生成
[blockMesh,
snappyHexMeshなど]



または

格子生成
[cfMesh,Salome,gmsh
商用メッシャー等]

必要あれば格子変換
[gmshToFoam等]

解析

初期設定
[setFields等]

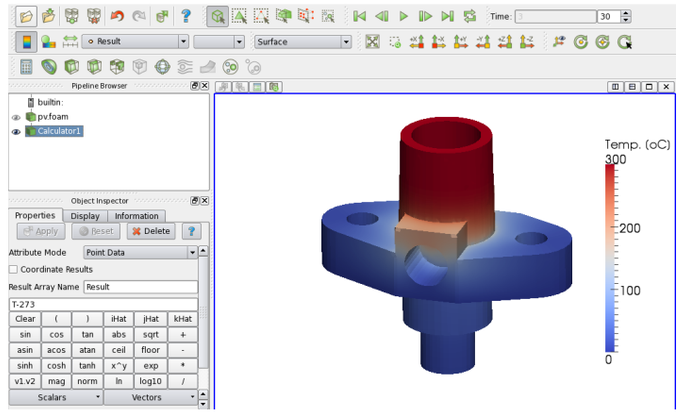
領域分割(並列計算時)
[decomposePar]

解析ソルバ
[icoFoam等]

領域統合(並列計算時)
[reconstructPar]

後処理(可視化など)

可視化
[ParaView, Visit,
商用可視化ツール等]



様々な結果後処理
[sample等]

FX10でのOpenFOAMの代表的な解析手順

ユーザーマシン

4. 格子可視化 [ParaView]

7.可視化 [ParaView]

1. ログイン [ssh]

3. 格子転送 [scpまたは rsync]

6. 解析結果転送 [scpまたは rsync]

ログインノード
PRIMERGY RX300

2.格子生成 [pjsub, blockMesh]

5.流体解析 [pjsub, ソルバ]

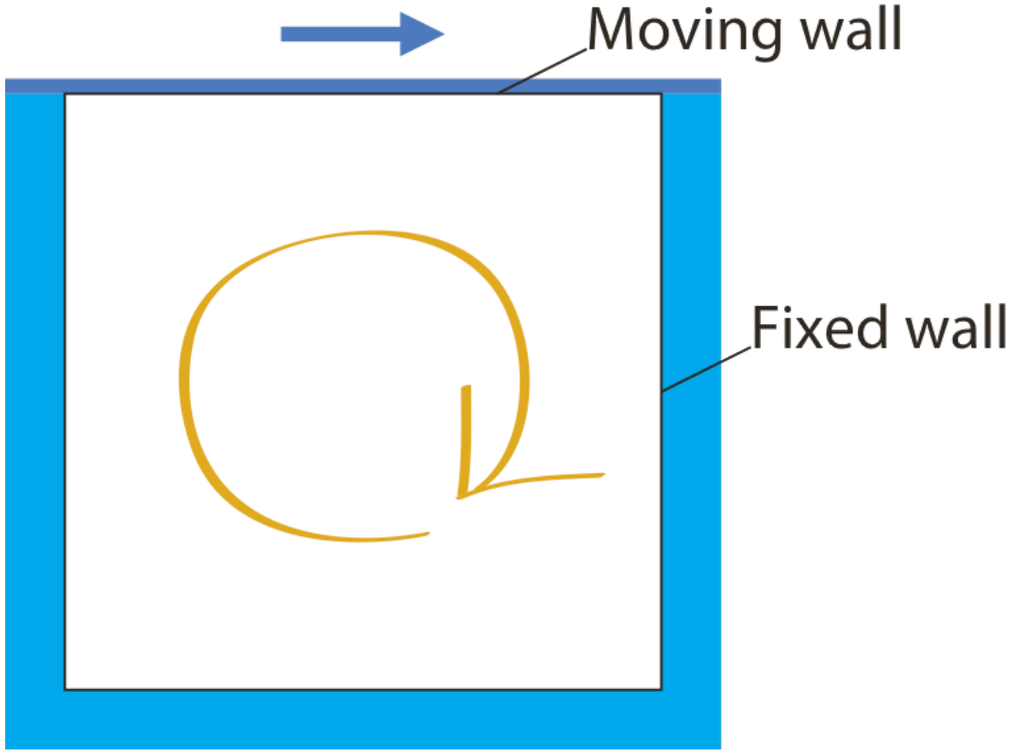
計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

キャビティ流れとは

- キャビティ(空洞)の上壁が動き、その摩擦で空洞内の流体が動く流れ場
- CFDソフトウェアの基礎的な検証例(ベンチマークテスト)として良く用いられる
- Ghiaら(1982)の論文が有名

C.T. Shin U. Ghia, K.N. Ghia. High-Re solution for incompressible flow using the Navier-Stokes equations and the multigrid method. J. Comput. Phys., 48:387–411, 1982.



図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」
第一回OpenFOAM講習会

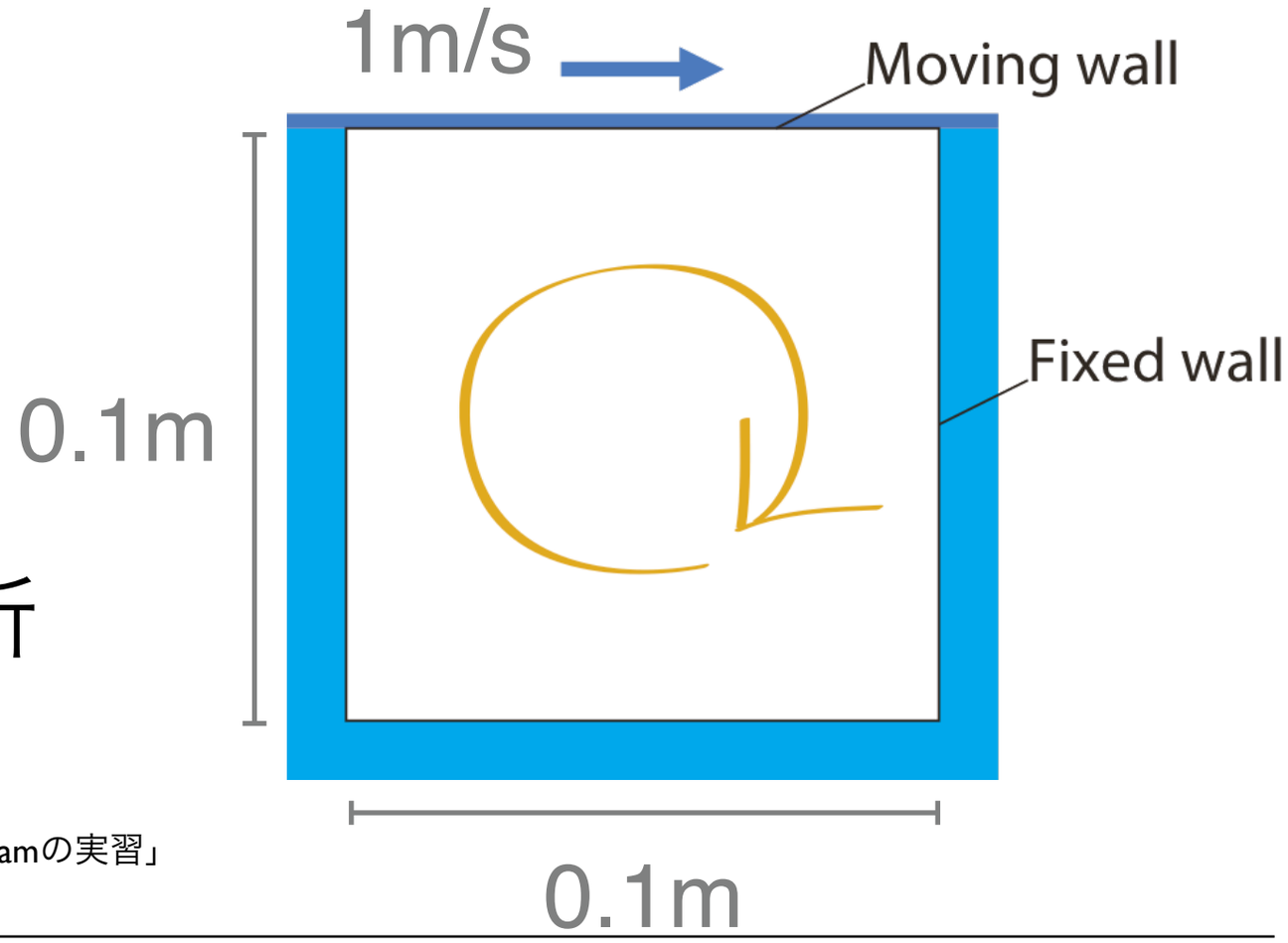
キャビティ流れのチュートリアル

- 正方形のキャビティで辺長： $d = 0.1m$
- 上壁の移動速度： $U = 1m/s$
- 流体の動粘性係数 (=粘性係数/密度)：$\nu = 0.01m^2/s$
- レイノルズ数：

$$Re = \frac{dU}{\nu} = 10$$

- 非圧縮性の層流解析

→ `icoFoam`ソルバーで解析



図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」
第一回OpenFOAM講習会

キャビティケースのディレクトリ

講習会用ファイルの展開およびcavityのケースディレクトリへ移動

```
Oakleaf-FXのホームディレクトリに講習用ファイルlecture.tar.gzをコピー
tar zxf lecture.tar.gz
cd ~/lecture/cavity
```

ケースディレクトリのファイル・ディレクトリ構成を表示

```
tree
```

```
.
├── 0
│   ├── U
│   └── p
├── blockMesh.PJM
├── constant
│   ├── polyMesh
│   │   ├── blockMeshDict
│   │   └── boundary
│   └── transportProperties
├── icoFoam.PJM
└── system
    ├── controlDict
    ├── fvSchemes
    └── fvSolution
```

キャビティケースのディレクトリ構成

ジョブファイル(*.PJM)は後述

0/ — 初期条件・境界条件ディレクトリ

U	速度ベクトル場
p	圧力場

constant/ — 不変な格子・定数・条件を格納するディレクトリ

transportProperties	流体物性(物性モデル，動粘性係数，密度など)

constant/ polyMesh/ — 格子データのディレクトリ

blockMeshDict	構造格子設定ファイル
boundary	境界パッチ設定ファイル

system/ — 解析条件を設定するディレクトリ

controlDict	実行制御の設定
fvSchemes	離散化スキームの設定
fvSolution	時間解法やマトリックスソルバの設定

キャビティケースの解析手順

1. 格子生成

`blockMesh`

constant/polyMesh/

- blockMeshDict — 構造格子設定ファイル
- boundary — パッチ設定(上書される)
- faces, neighbour, owner, points — 格子データ

0/

- U — 速度ベクトル場
- p — 圧力場

constant/

- transportProperties — 流体物性

system/

- controlDict — 実行制御の設定
- fvSchemes — 離散化スキームの設定
- fvSolution — ソルバの設定

解析結果の時刻ディレトリ/

- U, p, phi — 解析結果

2. ソルバ実行

`icoFoam`

- 読み込み
- 読み書き
- 新規作成

fileName **既存ファイル**

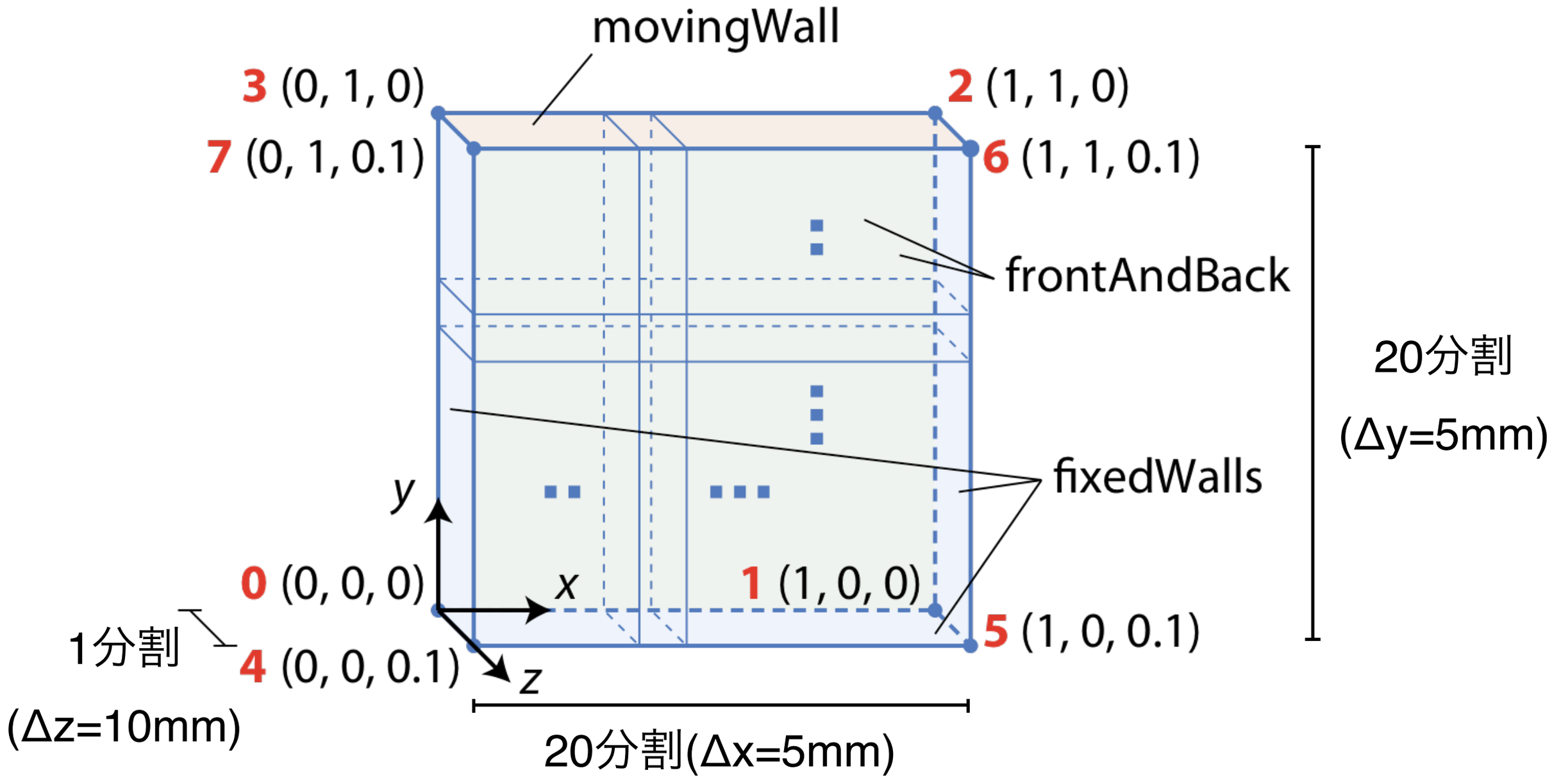
fileName **修正ファイル**

fileName **作成ファイル**

この他にも可視化等のポスト処理が通常含まれる

キャビティ流れの格子生成

キャビティケースの格子分割

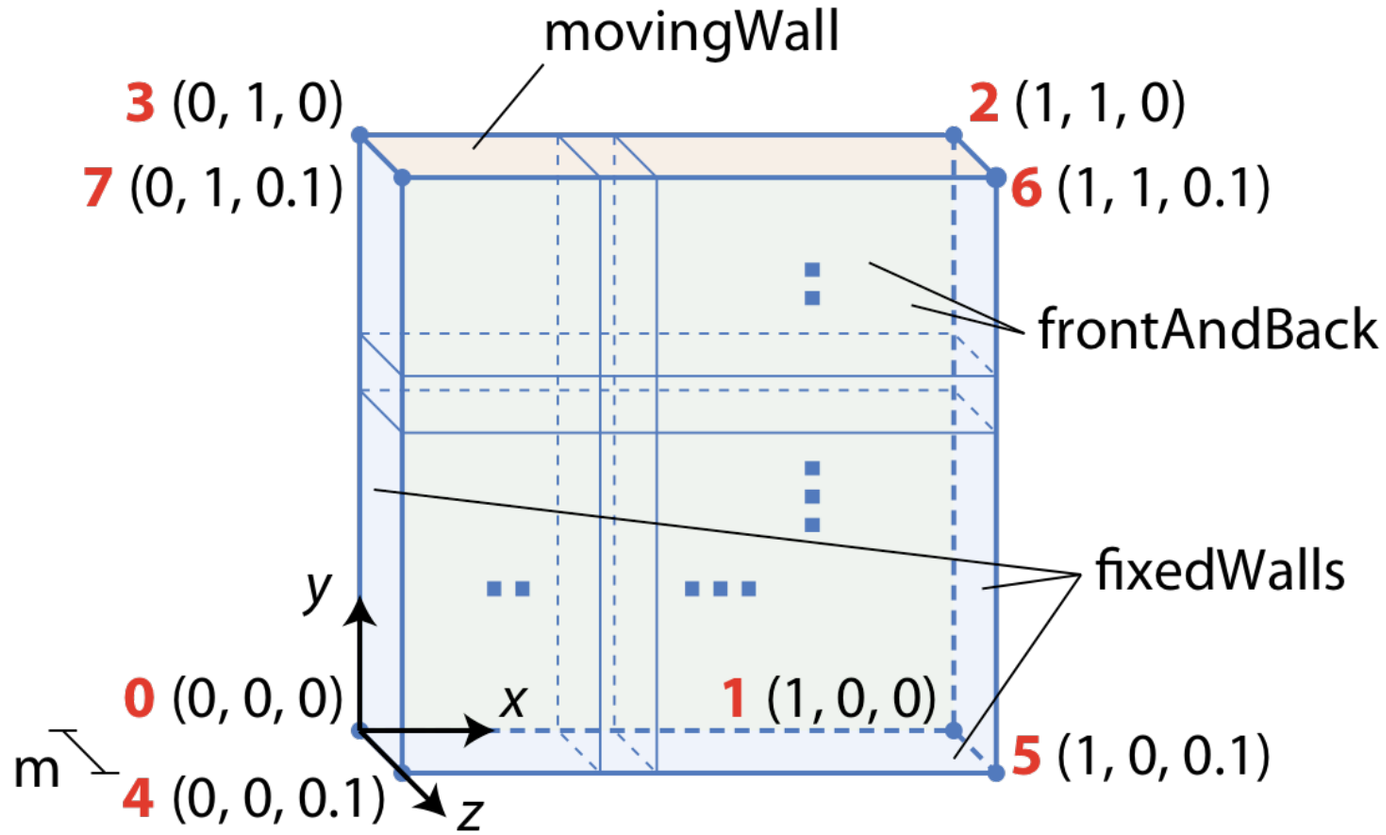


注)2次元なので、z方向は1分割(幅は任意)

図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」 第一回OpenFOAM講習会

設定ファイルblockMeshDict

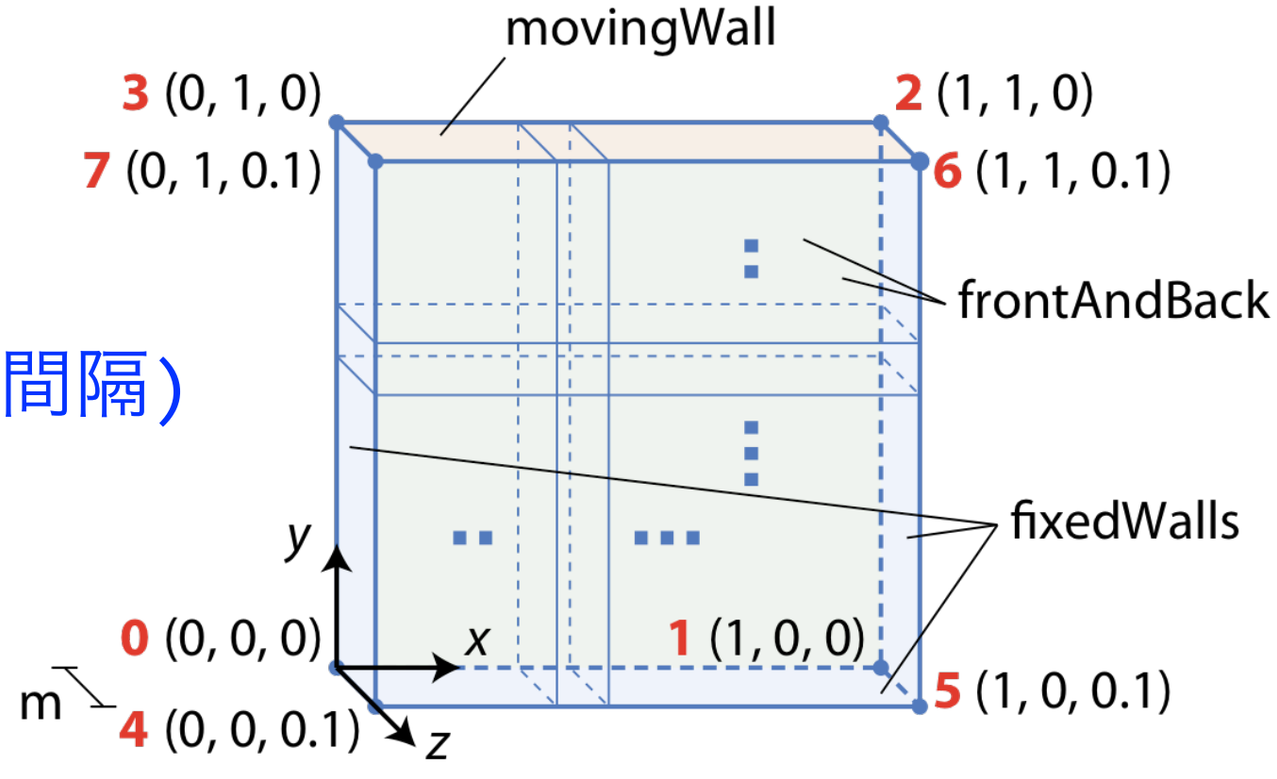
```
convertToMeters 0.1; メートル単位への変換係数

vertices 頂点の座標リスト
(
    (0 0 0)    頂点0
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)  頂点7
);
```



図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」 第一回OpenFOAM講習会

設定ファイルblockMeshDict

```
blocks   格子ブロックの定義
(
   hex   (0 1 2 3 4 5 6 7)      (20 20 1)
   形状       頂点番号リスト                  各方向の分割数

   simpleGrading   (1 1 1)
   格子幅は等比級数
                  各方向の格子幅比
  (最初と最後の格子の比。1:等間隔)
);
edges
(  辺が曲線の場合に指定
);
```

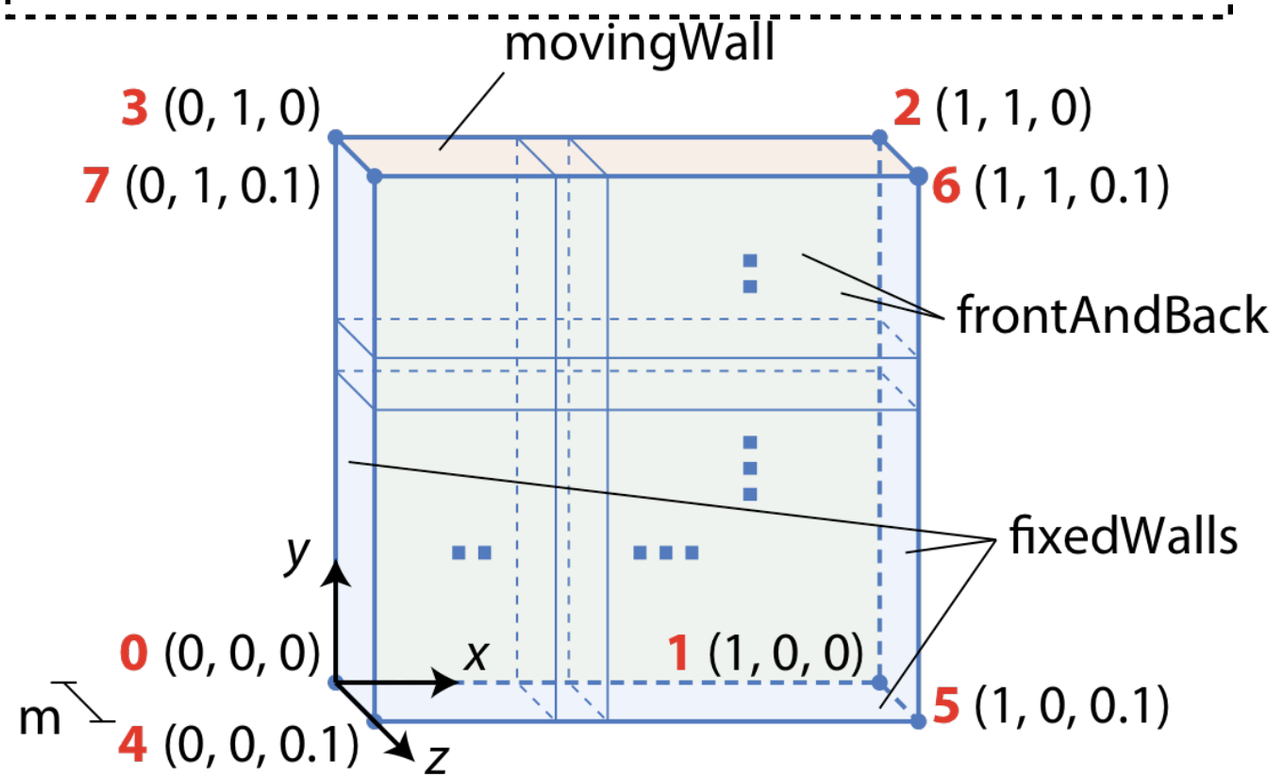


図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」 第一回OpenFOAM講習会

設定ファイルblockMeshDict

```
boundary 境界の設定
(
    movingWall 境界の名前
    {
        type wall; 種別：壁面
        faces 界面リスト
        (
            (3 7 6 2) 頂点リスト
        ); 内部から見て時計回り
    }       ただし，スタートの頂点は任意
    fixedWalls 境界の名前 (以下同様)
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty; 空の境界(2次元問題)
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

右上に続く



図出典: 大嶋 拓也 (新潟大学)「キャビティ流れの解析、paraFoamの実習」 第一回OpenFOAM講習会

blockMeshDictの確認

```
more constant/polyMesh/blockMeshDict
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
--続ける --(55%)
```

moreコマンドは続ける(英語版ではmore)と表示してキー入力待ちとなる.

主な操作キー　SPC：前、b：後、h：ヘルプ、q：終了

FX10でのOpenFOAMの代表的な解析手順

ユーザーマシン

4. 格子可視化
[ParaView]

7.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子転送
[scpまたは
rsync]

6. 解析結果転送
[scpまたは
rsync]

ログインノード
PRIMERGY RX300

2.格子生成
[pjsub,
blockMesh]

5.流体解析
[pjsub,
ソルバ]

計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

blockMesh実行用ジョブスクリプト

blockMesh.PJM

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture" #リソースグループ指定(演習中はtutorial)
#PJM -g gt00 #グループ名指定(適宜変更)
#PJM -L "node=1" #ノード数指定
#PJM -j #ジョブの標準エラー出力を標準出力へ出力
#------- Program execution -------#
module load OpenFOAM/2.3.0 #OpenFOAM/2.3.0のmoduleの設定
source $WM_PROJECT_DIR/etc/bashrc #OpenFOAMの環境設定

blockMesh > log.blockMesh #blockMeshを起動
```

blockMeshのジョブ投入

ジョブの投入

```
pjsub blockMesh.PJM
```

```
[INFO] PJM 0000 pjsub Job JOB_ID submitted. JOB_IDは各自異なる
```

ジョブ状態確認 (以降の演習ではジョブ状態の確認を適宜行ってください)

```
pjstat
```

→ ジョブ実行前の場合

```
JOB_ID      JOB_NAME    STATUS   PROJECT   RSCGROUP 以降略
JOB_ID      blockMesh.  QUEUED   gt00      lecture  以降略
```

→ ジョブ実行中の場合

```
JOB_ID      JOB_NAME    STATUS   PROJECT   RSCGROUP 以降略
JOB_ID      blockMesh.  RUNNING  gt00      lecture  以降略
```

→ ジョブ終了後の場合

```
No unfinished job found.
```

生成されたファイルの確認

ファイルの確認

```
tree
```

```
├── 0
│   ├── U
│   └── p
├── blockMesh.PJM
├── blockMesh.PJM.o1870352   ジョブの標準出力(JOB_IDは環境によって異なる)
├── constant
│   ├── polyMesh
│   │   ├── blockMeshDict
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points            ↕ 生成された格子ファイル
│   └── transportProperties
├── icoFoam.PJM
├── log.blockMesh    blockMeshのログ
└── system
    ├── controlDict
    ├── fvSchemes
    └── fvSolution
```

ジョブの標準出力を確認(エラーが出ていないことを確認)

```
more blockMesh.PJM.o*   *(ワイルドカード)は本来JOB_IDを書く.
```

blockMeshのログ確認

blockMeshのログ確認

```
more log.blockMesh
```

```
Mesh Information
----------------
  boundingBox: (0 0 0) (0.1 0.1 0.01) 解析領域範囲
  nPoints: 882 節点数
  nCells: 400 格子数
  nFaces: 1640 界面(フェース)数
  nInternalFaces: 760 内部界面(フェース)数
----------------
Patches パッチ(境界面)情報
----------------
  patch 0 (start: 760 size: 20) name: movingWall
  patch 1 (start: 780 size: 60) name: fixedWalls
  patch 2 (start: 840 size: 800) name: frontAndBack

End
```

格子データの転送と
ParaViewによる格子可視化

FX10でのOpenFOAMの代表的な解析手順

ユーザーマシン

4. 格子可視化
[ParaView]

7.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子転送
[scpまたは
rsync]

6. 解析結果転送
[scpまたは
rsync]

ログインノード
PRIMERGY RX300

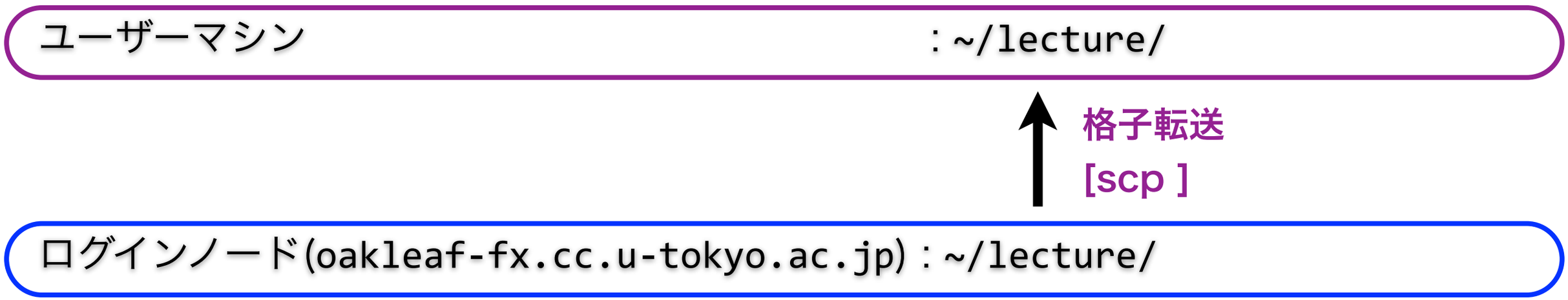
2.格子生成
[pjsub,
blockMesh]

5.流体解析
[pjsub,
ソルバ]

計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

講習用ファイル一式と格子データの転送



講習用ファイル一式と作成した格子データの転送(ユーザマシン側で実行)

ユーザー名@ユーザーマシン ~

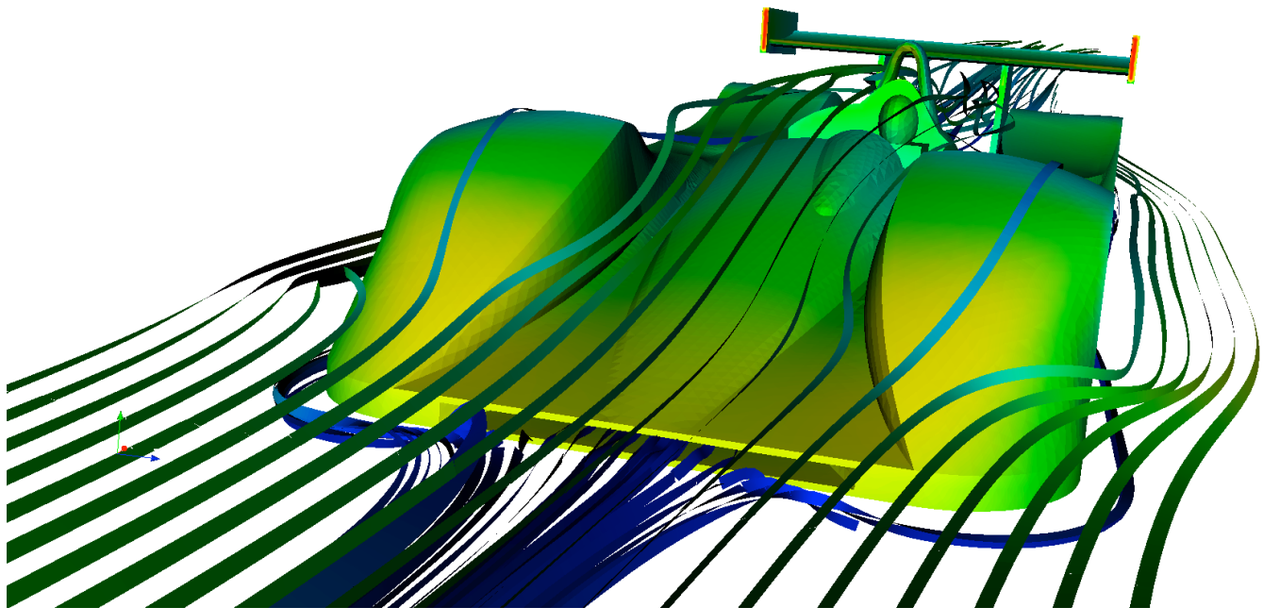
```
$ cd もしホームディレクトリ~上にいない場合はcdでホームディレクトリに移動
$ scp -r tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/lecture ./
        tYYxxx=利用者番号
```

以降，scpやrsyncでpassphraseが聞かれた場合には登録したものを入力する

```
$ cd lecture/cavity
```

ParaViewとは?

- オープンソース、スケーラブル、かつマルチプラットフォームな可視化アプリケーション
- 大容量データセットを処理するための分散型計算手法のサポート
- オープン、柔軟かつ直感的なユーザインターフェイス
- オープンな規格に基づいた拡張性の高いモジュール化構造
- 柔軟な3条項BSDライセンス
- 有償の保守およびサポート



ル・マンのレースカー周りの気流

(ブラジル リオ・デ・ジャネイロ NACAD/COPPE/UFRJ Renato N. Elias)

図出典：Kenneth Moreland et.al : Large Scale Visualization with ParaView, Supercomputing 2014 Tutorial, November 16, 2014

ParaViewによるOpenFOAMデータ可視化

✓OpenFOAMの格子や解析結果はParaViewで可視化やデータ解析が可能

✓OpenFOAMデータ読み込み方法

・OpenFOAM附属のparaFoamコマンド使用

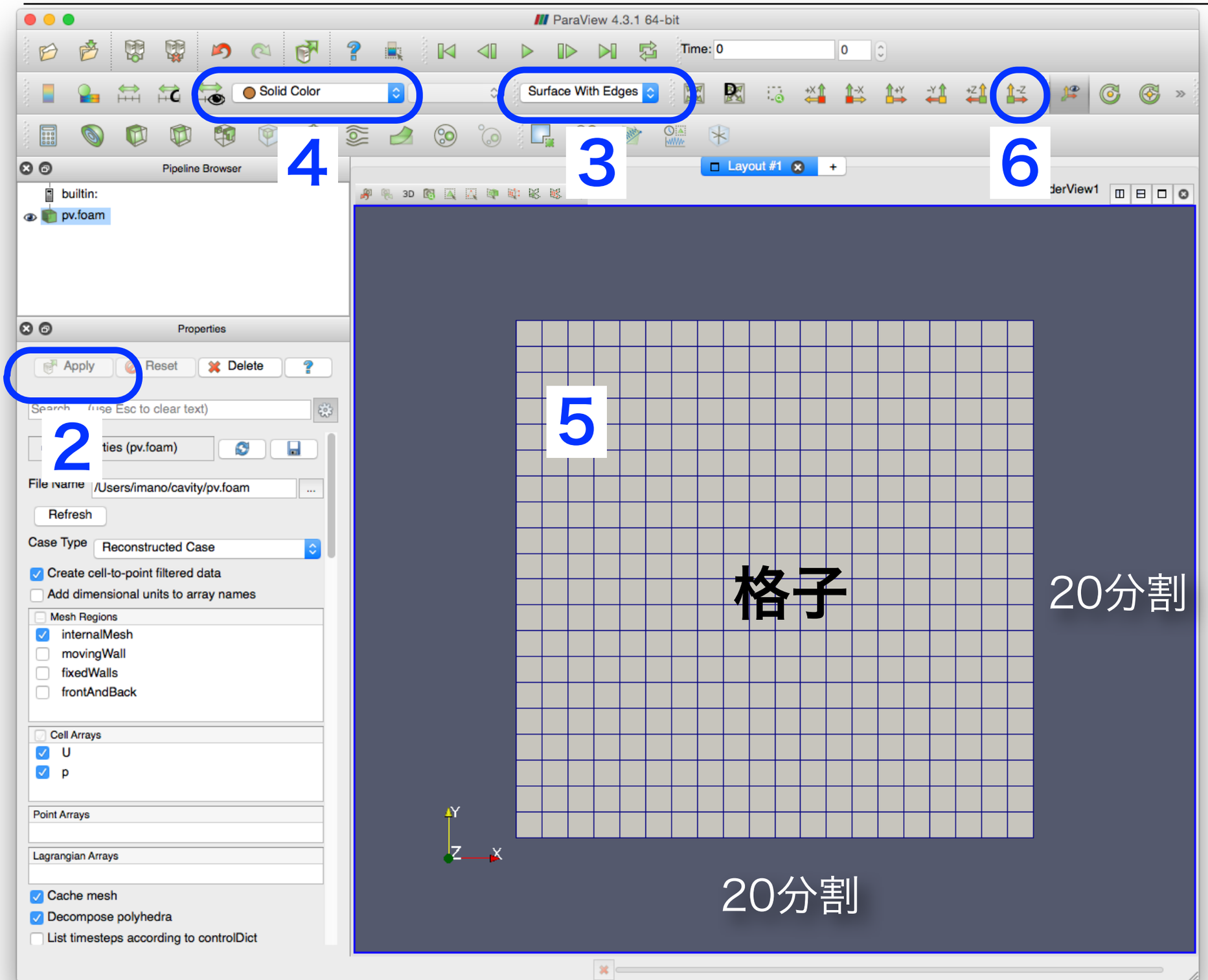
- ParaViewを起動するクライアント側にOpenFOAMの環境が必要
- ログインノード上で可視化する場合はこの方法
- ログインノードでのParaView起動は負荷がかかるため推奨しない

・通常のParaView(Ver.3.8以降)を使用　(今回はこの方法)

- ParaViewを起動するクライアント側にOpenFOAMの環境は不要
- FX10の格子・解析結果をPCに転送して，PC上のParaViewで可視化
- OpenFOAMのデータを読むには拡張子が.foamのファイルが必要

```
ユーザー名@ユーザーマシン ~/lecture/cavity
$ touch pv.foam touchで空ファイルが作成される．名前は任意
```

ParaViewによる格子の可視化



1. Fileメニュー/Open/cavityのディレクトリのpv.foamを開く
2. Apply
3. Representation/Surface With Edges 選択
4. Coloring/Solid Color 選択
5. マウスで適当に動す
6. 法線を-Z方向にして，リセット

左はMac版の画面.
その他のOS版では画面が多少異なる

キャビティ流れ解析

初期条件・境界条件設定

- 初期条件・境界条件ファイル：0/U, 0/p

なお、初期条件を置くディレクトリは０で無くても良く，後述の system/controlDict で指定

速度のファイルを見るために、端末で赤字のように打ちましょう

```
more 0/U
```

初期条件・境界条件設定

- 速度の初期条件・境界条件ファイル：0/U

```
dimensions   [ 0    1   -1   0    0    0    0 ];
```

	質量	長さ	時間	温度	物質量	電流	光度
単位の次元							
SI単位での例：	[kg]	[m]	[s]	[K]	[kgmol]	[A]	[cd]

(単位系は任意)

(長さ[m])・(時間[s])$^{-1}$ → 速度[m/s]

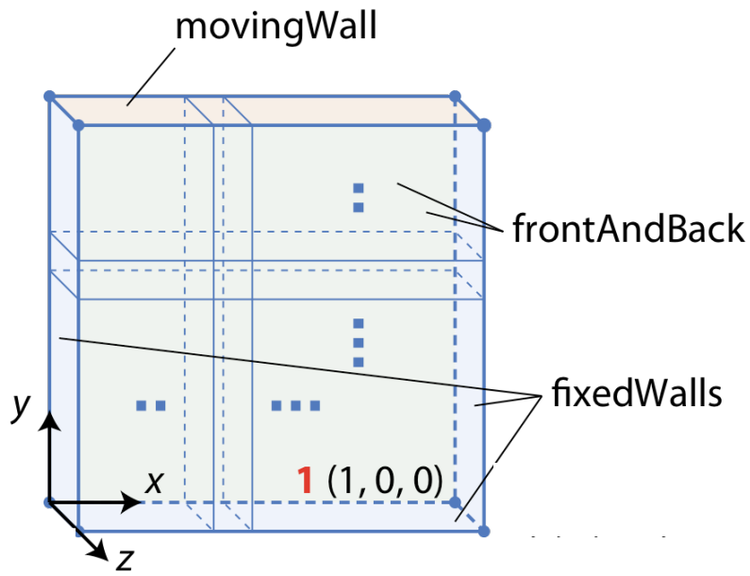
```
internalField    uniform   (0 0 0);
```

内部の場　　一様(分布)　　0ベクトル (速度0)

各演算では左辺・右辺での単位一致の検査がなされる

初期条件・境界条件設定

```
boundaryField      境界条件
{
  movingWall    動く上壁
  {
    type  fixedValue;     値固定
    value uniform (1 0 0);     (1,0,0)で一様
  }
  fixedWalls    固定壁
  {
    type  fixedValue;     値固定
    value uniform (0 0 0);     (0,0,0)で一様(壁)
  }
  frontAndBack
  {
    type  empty;     2次元なので空
  }
}
```



初期条件・境界条件設定

圧力のファイルを見るために、端末で赤字のように打ちましょう

```
more 0/p
```

- 速度の初期条件・境界条件ファイル：0/p

```
dimensions      [ 0   2   -2   0   0   0   0 ];
```

単位の次元　　質量　長さ　時間　温度　物質量　電流　光度

OpenFOAMの非圧縮性ソルバでは、圧力は密度で割っている

(質量)・(長さ)・(時間)$^{-2}$ /((質量)・(長さ)$^{-3}$)= (長さ)^{2} ・(時間)$^{-2}$

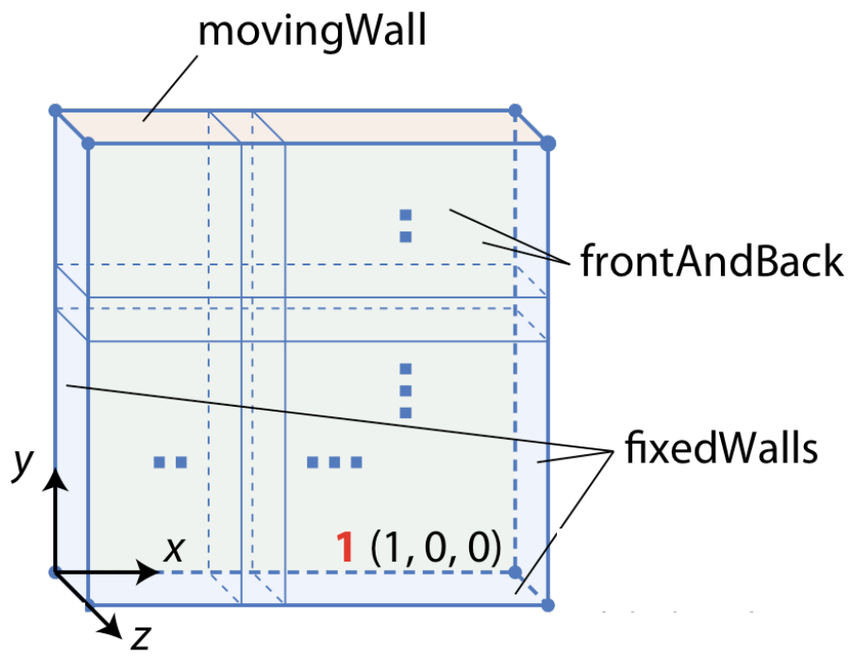
圧力の次元　　密度の次元　　pの次元

```
internalField   uniform   0;
```

内部の場　　一様(分布)　　圧力0

初期条件・境界条件設定

```
boundaryField
{
  movingWall 動く上壁
  {
    type zeroGradient; 境界の法線方向勾配0
  }

  fixedWalls 固定壁
  {
    type zeroGradient;
  }

  frontAndBack
  {
    type empty; 2次元なので空
  }
}
```



実行条件等の設定

cavity　　ケース・ディレクトリ

0

constant

system　…… 解析条件を設定するディレクトリ

controlDict　…… 実行条件の設定

fvSchemes　…… 離散化スキームの設定 (詳細は略)

fvSolution　…… 時間解法やマトリックスソルバの設定 (詳細は略)

実行条件等の設定

system/controlDict

```
application         icoFoam;        ソルバー名
startFrom           startTime;      解析開始の設定法(他にlatestTime等)
startTime           0;              解析の開始時刻
stopAt              endTime;        解析終了の設定法(他にnextWrite等)
endTime             0.5;            解析の終了時刻
deltaT              0.005;          時間刻み(単位[s])
writeControl        timeStep;       解析結果書き出しの決定法
writeInterval       20;             書き出す間隔(20time step=0.1s毎)
writeFormat         ascii;          データファイルのフォーマット(ascii, binary)
```

binaryのほうが入出力が速い．ただし，FX10はBig endian機なのでbinaryのデータはintel等のLittle endian機で読めない．asciiで計算するか，binaryで計算後，writeFormatをasciiにしてfoamFormatConvertを実行し，ascii形式に変換してからLittle endian機に転送する．

```
writePrecision      6;              データファイルの有効桁(上記がasciiの場合)
writeCompression    off;            データファイルの圧縮(off, on)
timeFormat          general;        時刻ディレクトリのフォーマット
timePrecision       6;              時刻ディレクトリのフォーマット有効桁
runTimeModifiable   true;           各時間ステップで設定ファイルを再読み込みするか
```

FX10でのOpenFOAMの代表的な解析手順

ユーザーマシン

4. 格子可視化
[ParaView]

7.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子転送
[scpまたは
rsync]

6. 解析結果転送
[scpまたは
rsync]

ログインノード
PRIMERGY RX300

2.格子生成
[pjsub,
blockMesh]

5.流体解析
[pjsub,
ソルバ]

計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

icoFoam用ジョブスクリプトとジョブ投入

icoFoam.PJM（逐次ジョブ用スクリプト）

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture" #リソースグループ指定(演習中はtutorial)
#PJM -g gt00 #グループ名指定(適宜変更)
#PJM -L "node=1" #ノード数指定
#PJM -j #ジョブの標準エラー出力を標準出力へ出力
#------- Program execution -------#
module load OpenFOAM/2.3.0 #OpenFOAM/2.3.0のmoduleの設定
source $WM_PROJECT_DIR/etc/bashrc #OpenFOAMの環境設定

icoFoam> log #ソルバーicoFoamを実行して，ログをlogに出力
```

ジョブの投入

```
pjsub icoFoam.PJM
```

生成ファイルの確認

ファイルの確認

```
tree
```

新規に生成されたもの

```
├── 0.1～0.5 解析結果の時刻ディレクトリ
│   ├── U 速度ベクトル
│   ├── p 圧力
│   ├── phi 流束
│   └── uniform: 時刻ディレクトリの情報ファイルを収めたディレクトリ
│       └── time 時刻や時間刻み等の情報
├── icoFoam.PJM.o186970 ジョブの標準出力ファイル(JOB_IDは異なる)
└── log ソルバのログ
```

流体解析のログ

ログの確認

```
more log
```

```
Build  : 2.3.0-f5222ca19ce6 ビルドバージョン(実行環境によって異なる)
Exec   : icoFoam 実行コマンド
Date   : Jan 1 2015 開始日時(実行環境によって異なる)
Time   : 00:00:00 開始時刻(実行環境によって異なる)
Host   : "a01t70081" ホスト名(実行環境によって異なる)
PID    : 22439 プロセスID(“kill プロセスID” でプロセスを強制終了できる)
Case   : /home/******/lecture/cavity ケースディレクトリ(ユーザ毎に異なる)
nProcs : 1 使用プロセッサ数(今回シングル計算なので1)
以下は実行時環境設定(実行環境によって異なる)
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using
timeStampMaster
allowSystemOperations : Disallowing user-supplied system call operations
```

流体解析のログ

```
Starting time loop 時間(反復)ループの開始

Time = 0.005 時間(反復)ループの開始

Courant Number mean: 0 max: 0
smoothSolver:  Solving for Ux, Initial residual = 1, Final residual =
8.90511e-06, No Iterations 19
Ux(速度のx方向成分) の離散方程式についての線形ソルバのログ
smoothSolver: 線型ソルバ(GaussSeidel法)
Initial residual: 初期残差, Final residual: 最終残差
No Iterations 1 反復回数
smoothSolver:  Solving for Uy, Initial residual = 0, Final residual = 0, No
Iterations 0
Uy (速度のy方向成分)の離散方程式についての線形ソルバのログ
ICPCG:  Solving for p, Initial residual = 1, Final residual = 7.55423e-07,
No Iterations 35
p(圧力)の離散方程式についての線形ソルバのログ
DICPCG: 線型ソルバ, 前処理：DIC, 線型ソルバ：PCG(前処理付き共役勾配法)
```

流体解析のログ

```
time step continuity errors : sum local = 5.03808e-09, global =
-7.94093e-21, cumulative = -7.94093e-21
```

連続の式の誤差

sum local : 誤差絶対値の格子体積重み付け平均

global : 誤差(符号あり)の格子体積重み付け平均

cumulative : globalの累積

```
ExecutionTime = 0.05 s  ClockTime = 0 s
```

ExecutionTime: 計算のみに要した時間, ClockTime :ファイルI/Oなども含めた実際の経過時間

```
Time = 0.01

Courant Number mean: 0.0976803 max: 0.585723
```

Time = 2 時間(反復)ループ。以下同様

略

End

解析結果の転送と
ParaViewによる解析可視化

FX10でのOpenFOAMの代表的な解析手順

ユーザーマシン

4. 格子可視化
[ParaView]

7.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子転送
[scpまたは
rsync]

6. 解析結果転送
[scpまたは
rsync]

ログインノード
PRIMERGY RX300

2.格子生成
[pjsub,
blockMesh]

5.流体解析
[pjsub,
ソルバ]

計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

解析結果の転送

ユーザーマシン　　：~/lecture/cavity/

解析結果転送 [rsync]

ログインノード(oakleaf-fx.cc.u-tokyo.ac.jp)：~/lecture/cavity/

解析結果の転送

ユーザー名@ユーザーマシン ~/lecture/cavity

```
$ rsync -auv tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/lecture/cavity/ ./
```

転送元と転送先どちらもに/(スラッシュ)を付ける ↑　↑

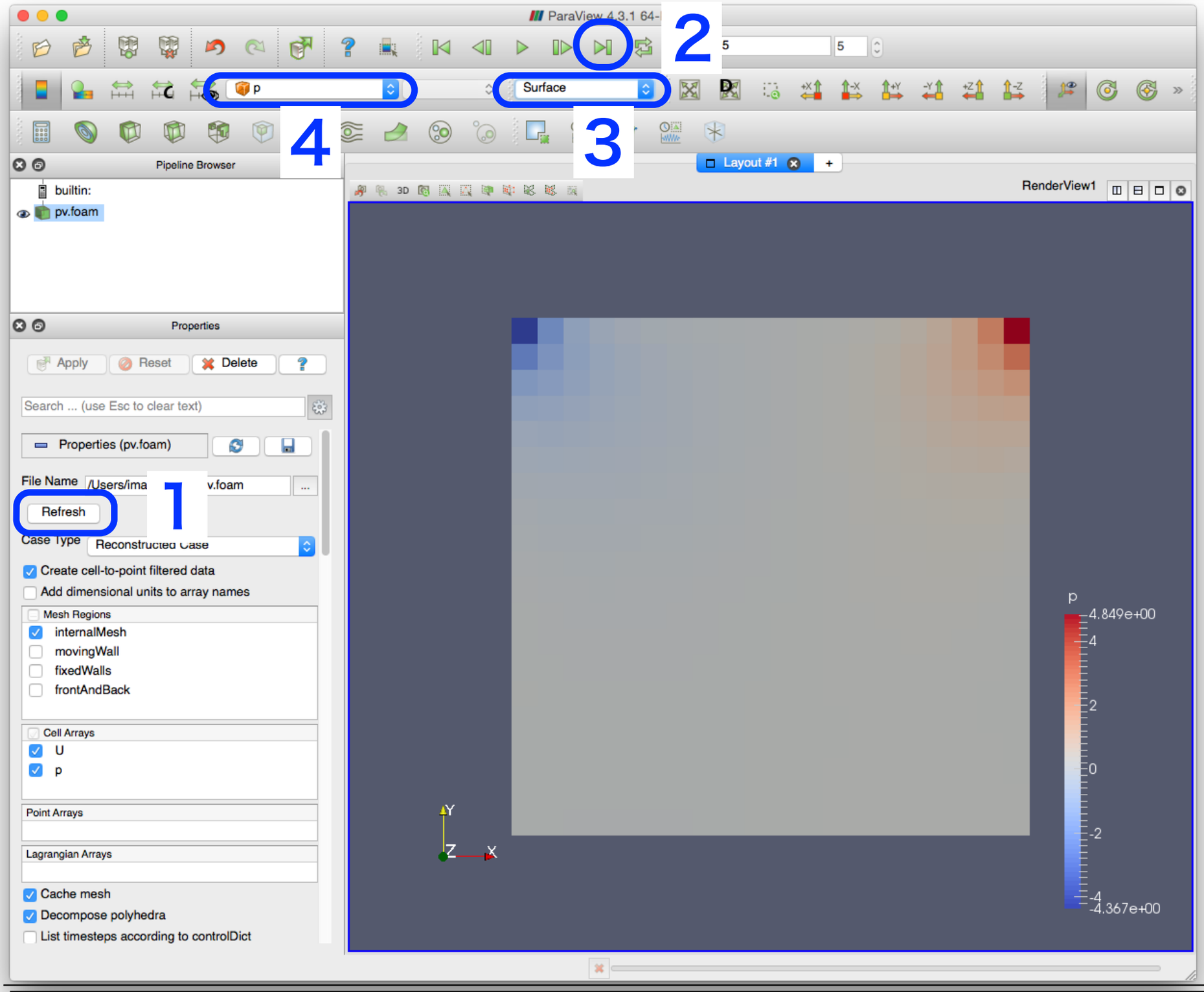
a=archive(ディレクトリを再帰的かつ，ファイル情報を保持したまま転送),

u=update(新規・更新されたもののみ転送)，v=verbose(転送情報を表示)

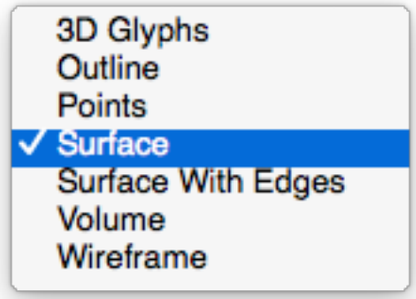
```
receiving file list ... done
./
log
略
sent 592 bytes  received 289598 bytes  193460.00 bytes/sec
total size is 356595  speedup is 1.23
```

新規作成・更新されたicoFoamの解析結果ファイルのみ転送される(rsyncを使う所以)

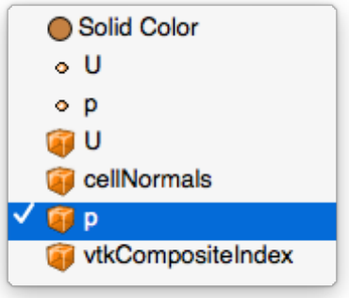
ParaViewによる圧力場の可視化



1. Refresh(更新)
2. Last Frame
3. Representation /Surface

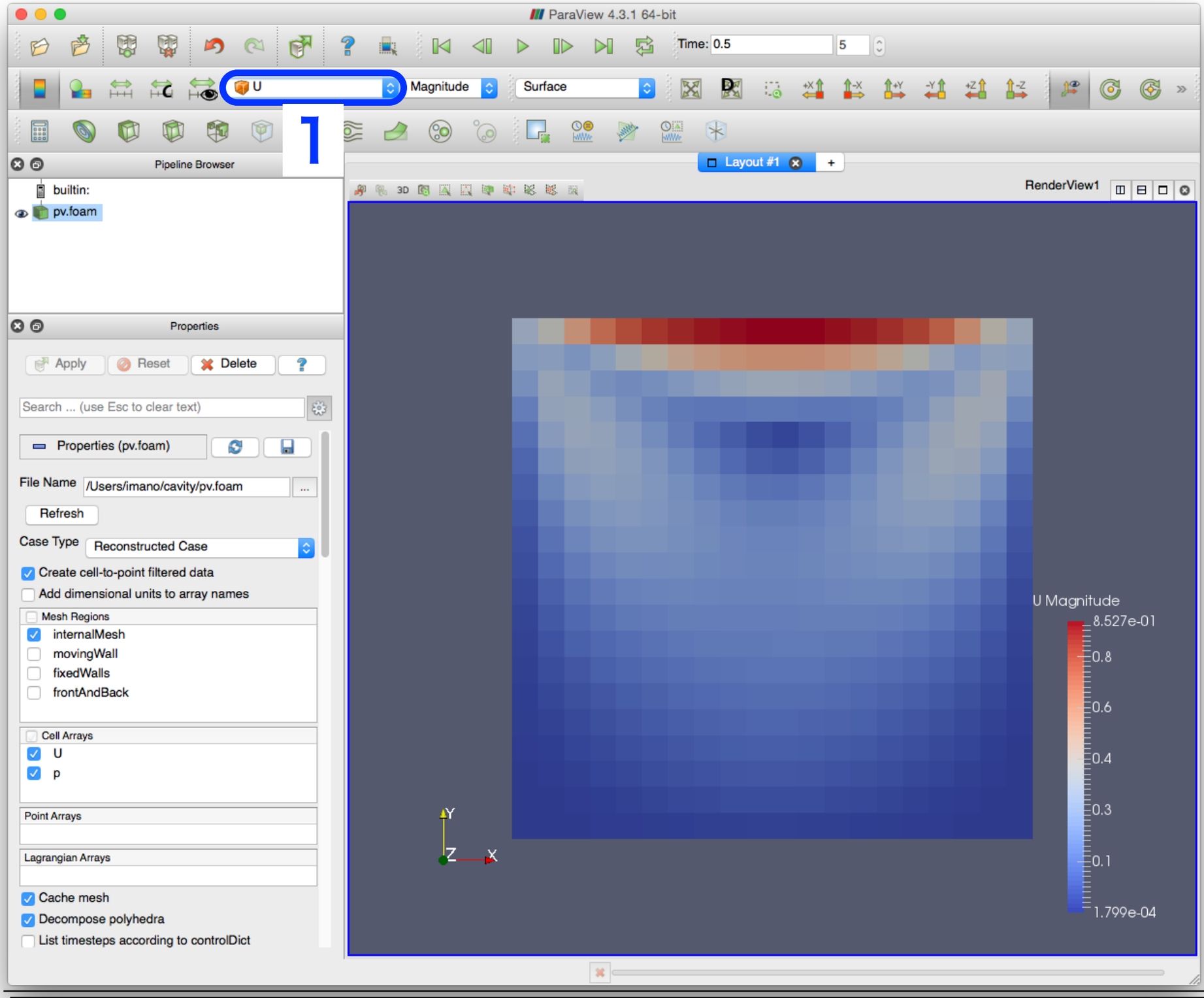


4. Coloring/□p

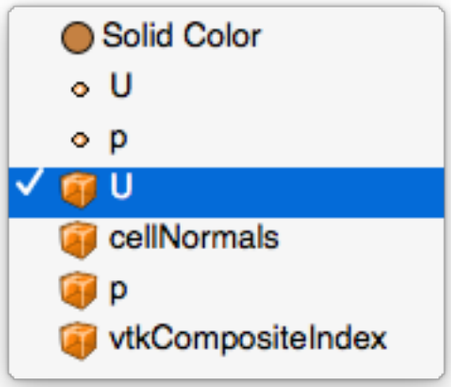


(□は格子の値そのままで補間無し、○は補間有り=スムージング)

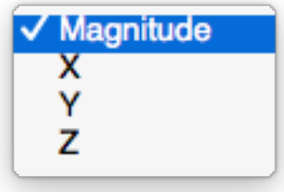
ParaViewによる風速の可視化



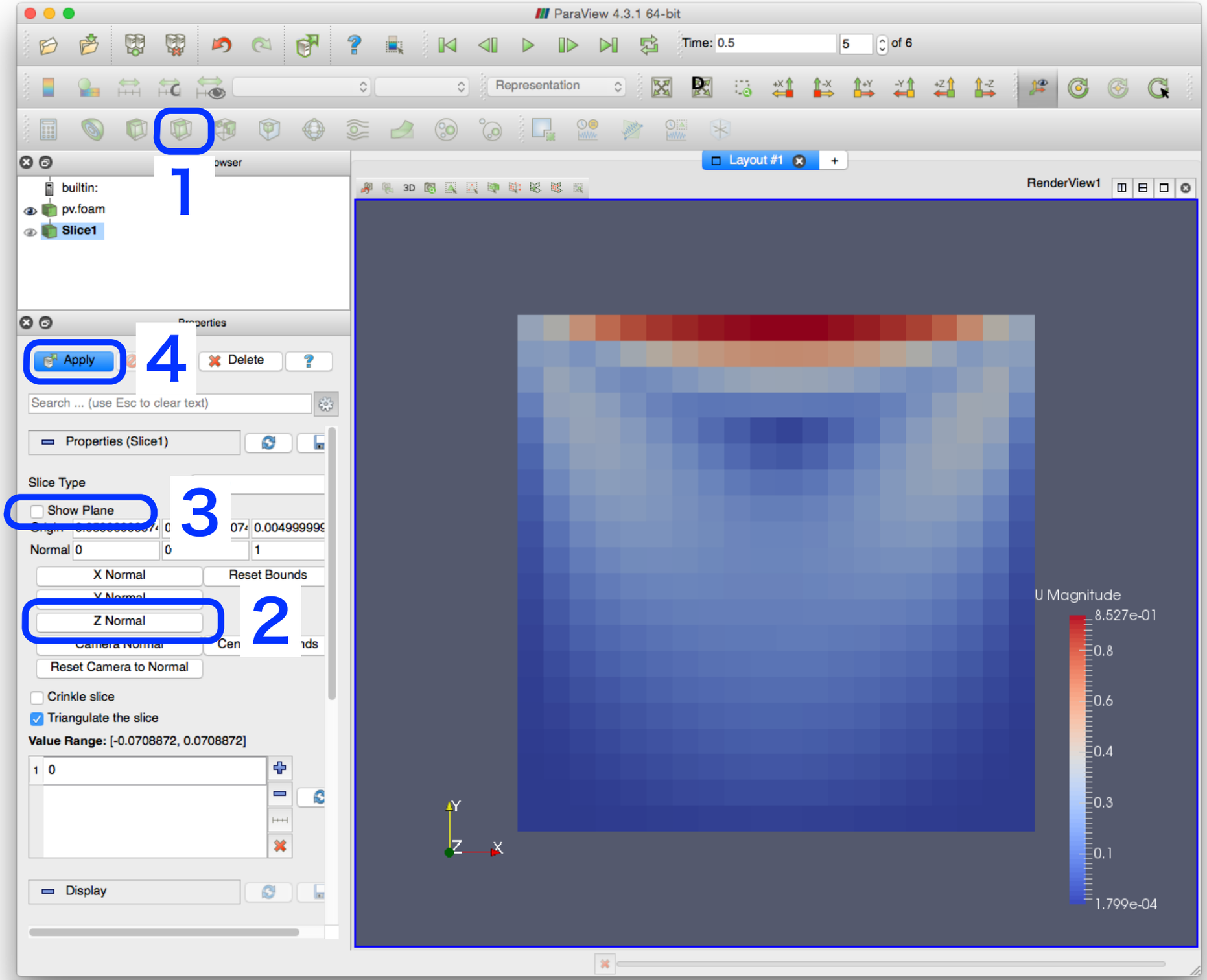
1. Coloring/□U



2. Magnitudeを X，Y，Zに変更することで，各風速成分が可視化できる．

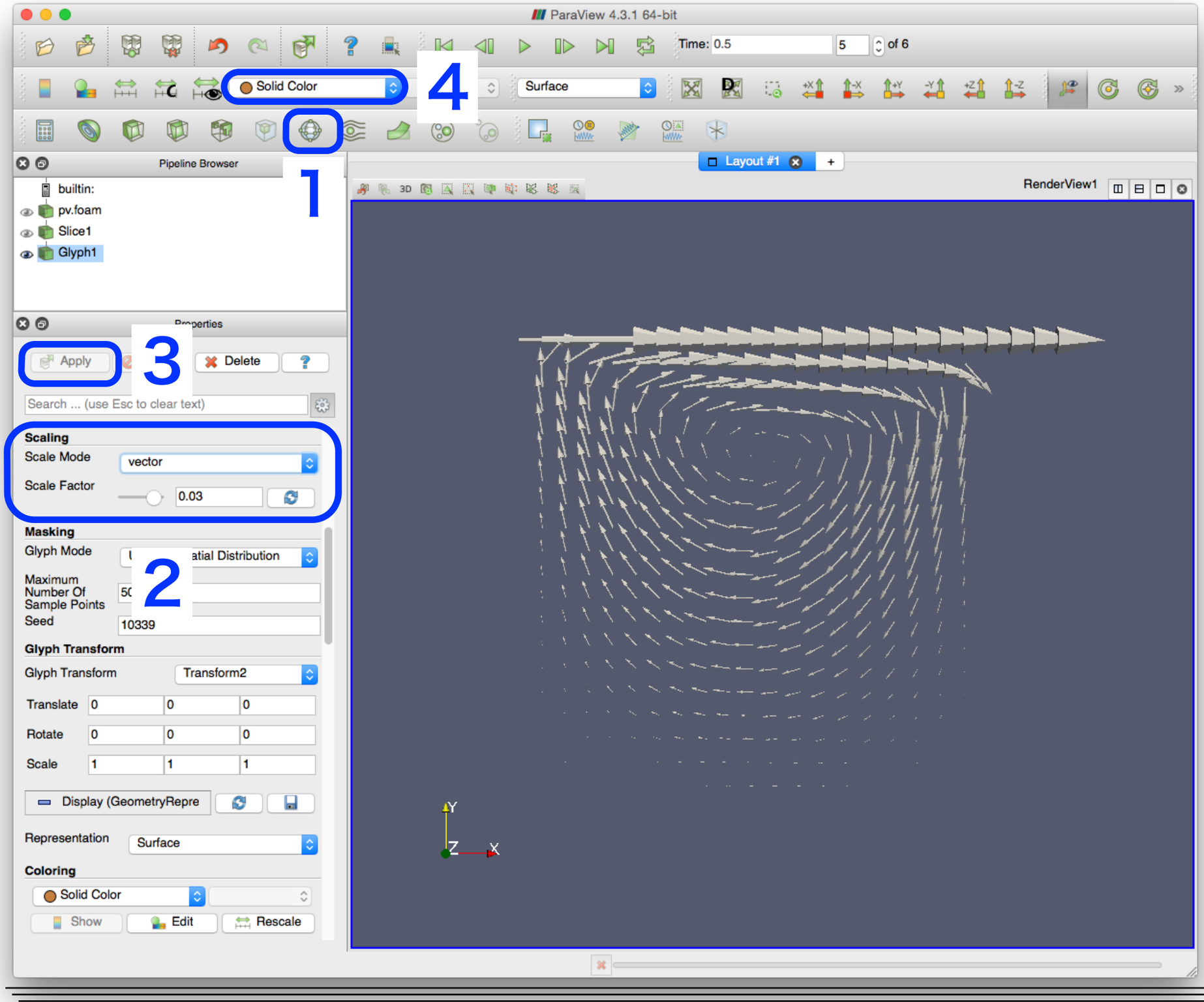


ParaViewによる風速ベクトルの可視化



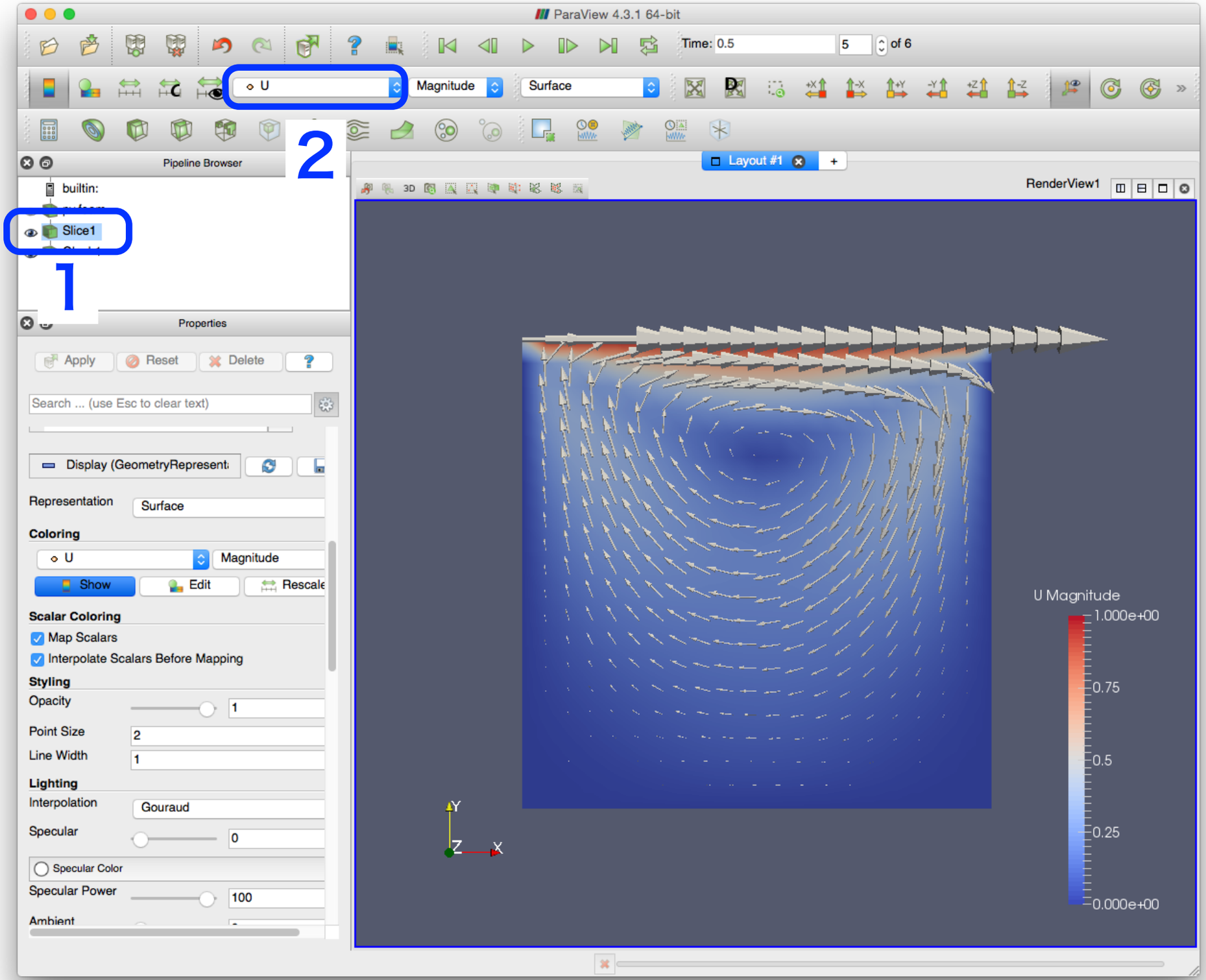
1. Sliceフィルタ
2. Z Normal
3. Show Planeを非選択
4. Apply

ParaViewによる風速ベクトルの可視化



1. Glyphフィルタ
2. Scaling/Scale Mode/vectorを選択．Scale Factor: 0.03
3. Apply
4. Coloring/Solid Color を選択

ParaViewによる風速ベクトルの可視化



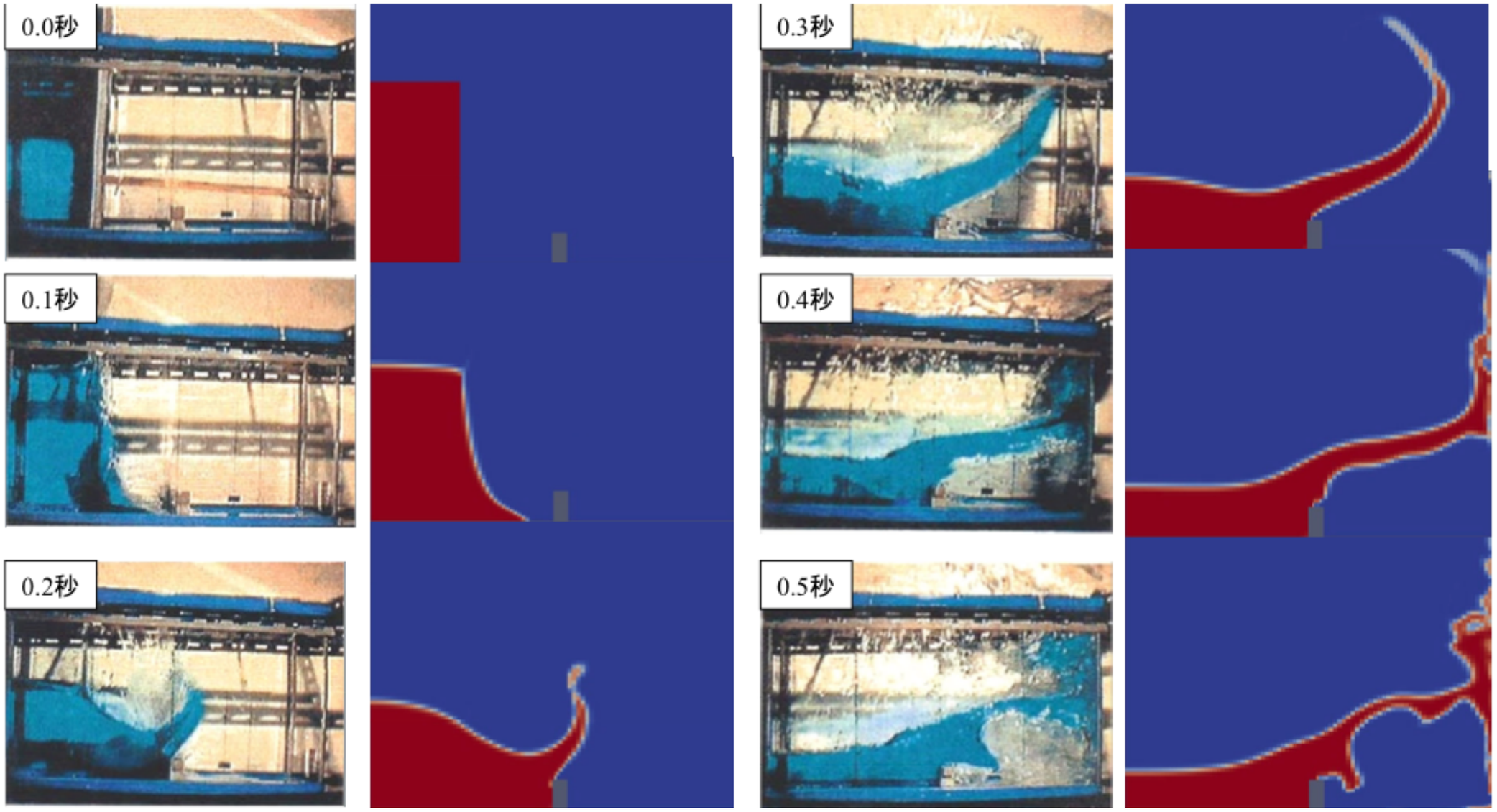
1. Sliceフィルタを表示(目のマーク をチェック)
2. Coloring/・Uを選択
3. Quitメニュー/Quit ParaViewでParaView終了

ダムブレイク流れ

damBreakFineケース

Koshizukaら[1]によるダム崩壊(dam break)の実験をVOF(Volume of Fluid)法の二相流ソルバinterFoamを用いて，二次元層流モデルで解析したもの

解析図出典：SM「ただで始める流体解析」第11回オープンCAE勉強会@岐阜



[1] Koshizuka, S., H. Tamako and Y. Oka :“A Particle Method for Incompressible Viscous Flow with Fluid Fragmentation”, CFD Journal ,Vol.4, No.1, pp.29-46, 1995

OpenFOAMのVOF法

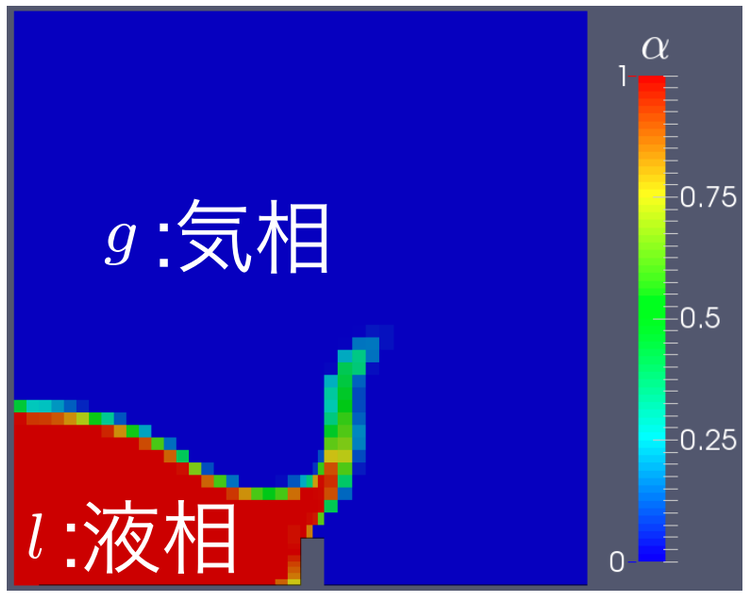
質量保存式　： $\nabla \cdot \boldsymbol{U} = 0$

運動量保存式： $\dfrac{\partial \rho \boldsymbol{U}}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) - \nabla \cdot \tau = -\nabla p + \rho \boldsymbol{g} + \boldsymbol{F_\sigma}$

相率輸送式　： $\dfrac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \boldsymbol{U}) + \nabla \cdot [(1-\alpha)\alpha \boldsymbol{U}_r] = 0$

界面への圧縮項の導入により保存・収束・有界性が向上[2,3]

$\boldsymbol{U}$	：速度	α	：相率(体積分率)
ρ	：密度	σ	：表面張力
τ	：応力テンソル	κ	：界面の曲率
p	：圧力	$\boldsymbol{U}_r$	：相対速度(圧縮速度) $(= \boldsymbol{U}_l - \boldsymbol{U}_g)$
$\boldsymbol{g}$	：重力加速度		
$\boldsymbol{F_\sigma}$	：表面張力(CSFモデル)	$(= \sigma\kappa\nabla\alpha)$	

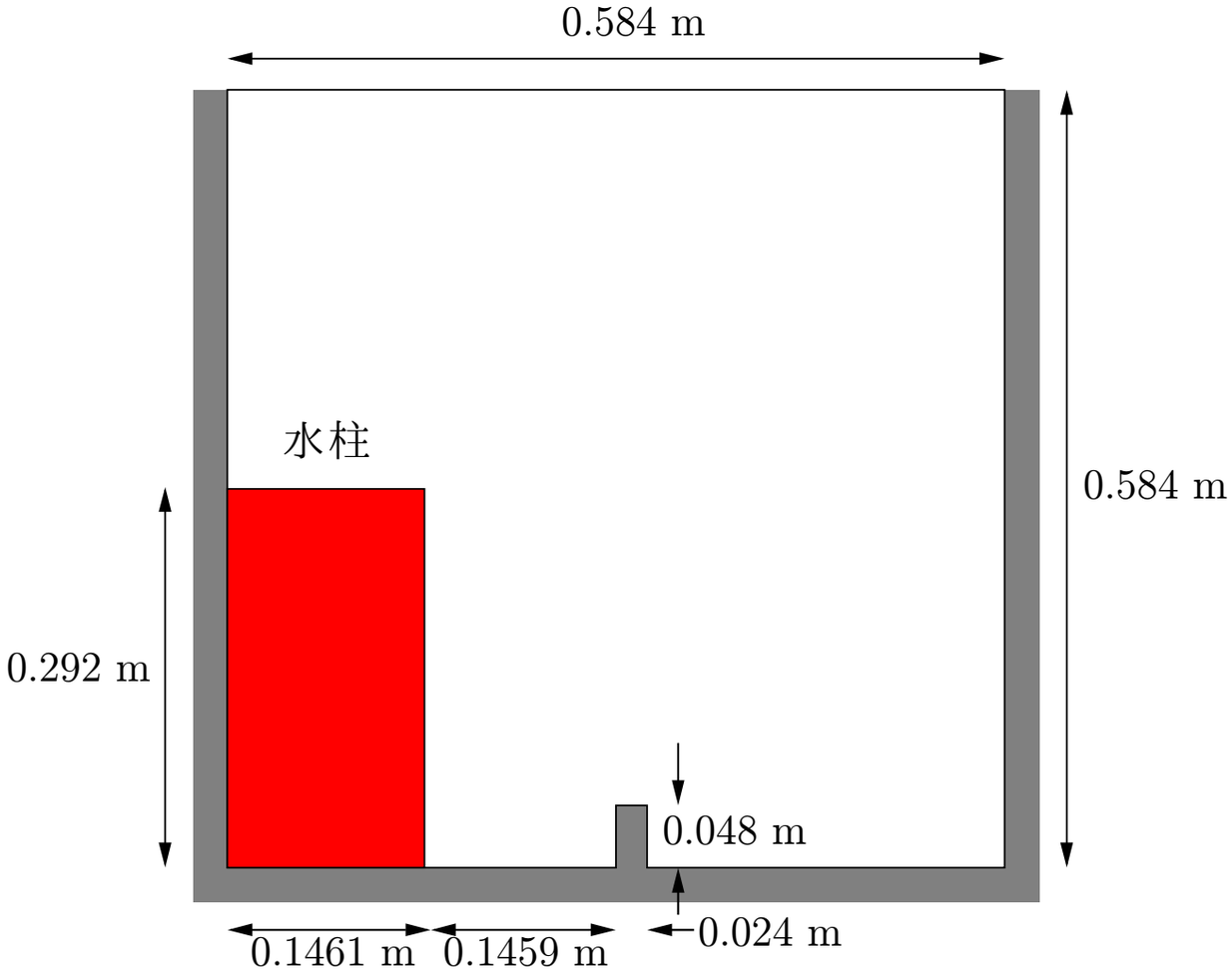


[2] Weller, H.G., 2008. A new approach to VOF-based interface capturing methods for incompressible and compressible flows. Technical Report No. TR/HGW/04

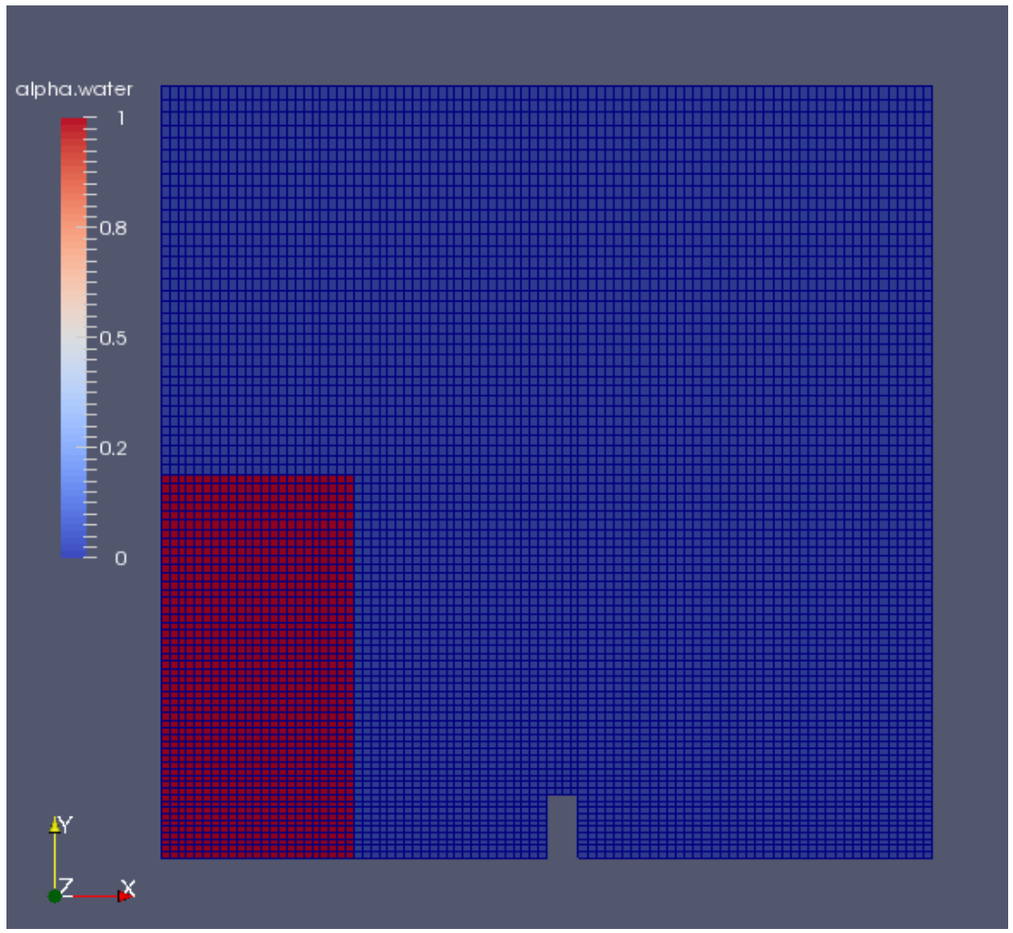
[3] A. Albadawi, D.B. et al., 2013. Influence of surface tension implementation in Volume of Fluid and coupled Volume of Fluid with Level Set methods for bubble growth and detachment, International Journal of Multiphase Flow, Volume 53, pp.11-28

damBreakFineケース計算条件

- 水柱：a(x)×2a(y)，解析領域: 4a(x)× 4a(y) (水柱幅a=0.146[m])
- 格子数：7,700，構造格子
- 乱流モデル：無し(層流モデル．標準k-ε モデルのチュートリアルも有り)
- 圧力速度解法：PISO法，時間刻み：最大クーラン数を1以下に制御



解析対象



格子分割と水相率の初期分布

図引用元: OpenFOAM User Guide Version 2.1.0和訳版

damBreakFineケースディレクトリ

damBreakケースのディレクトリへ移動

```
cd ~/lecture/damBreakFine/
```

ケースディレクトリのファイル・ディレクトリ構成を表示

```
tree
```

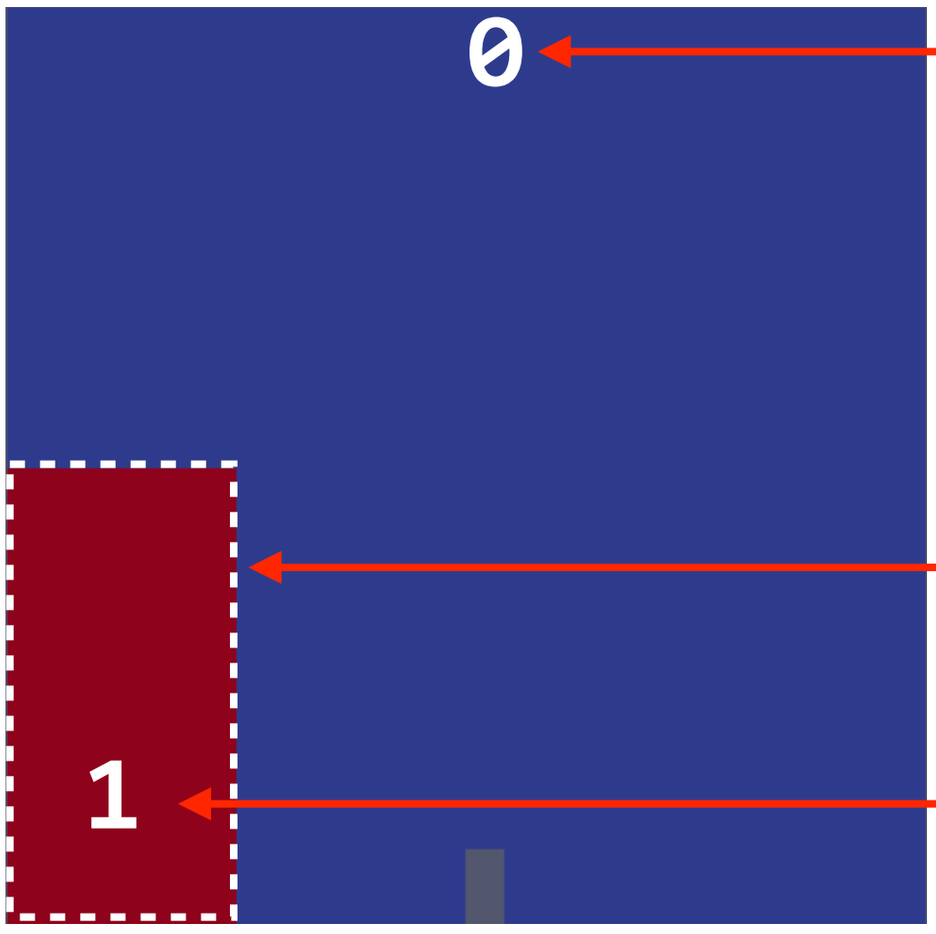
```
.
├── 0
│   ├── U
│   ├── alpha.water.org
│   └── p_rgh
├── Allrun-p.PJM
├── Allrun.PJM
├── constant
│   ├── g
│   ├── polyMesh
│   │   ├── blockMeshDict
│   │   └── boundary
│   ├── transportProperties
│   └── turbulenceProperties
└── system
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    └── setFieldsDict
```

damBreakFineケースの主なファイル構成

0/	**初期条件・境界条件ディレクトリ**
U	速度ベクトル場
alpha.water, alpha.water.org	水相率場 (.orgは水柱の分布が設定されていない元ファイル)
p_rgh	静水圧を引いた圧力場
constant/	**不変な格子・定数・条件を格納するディレクトリ**
dynamicMeshDict	移動格子設定(今回は静止格子 staticFvMesh)
g	重力加速度
transportProperties	流体物性(物性モデル，動粘性係数，密度など)
turbulenceProperties	乱流モデル(今回は層流モデル laminar)
polyMesh/	**格子データのディレクトリ**
blockMeshDict	構造格子設定ファイル
boundary	境界パッチ設定ファイル
system/	**解析条件を設定するディレクトリ**
controlDict	実行制御の設定
decomposeParDict	並列計算用領域分割の設定
fvSchemes	離散化スキームの設定
fvSolution	時間解法やマトリックスソルバの設定
setFieldsDict	場の初期分布設定(alpha.waterに対する水柱分布設定)

水相率の場への水柱分布設定

水相率の場 alpha.water への水柱分布の設定を行うsetFieldsの設定

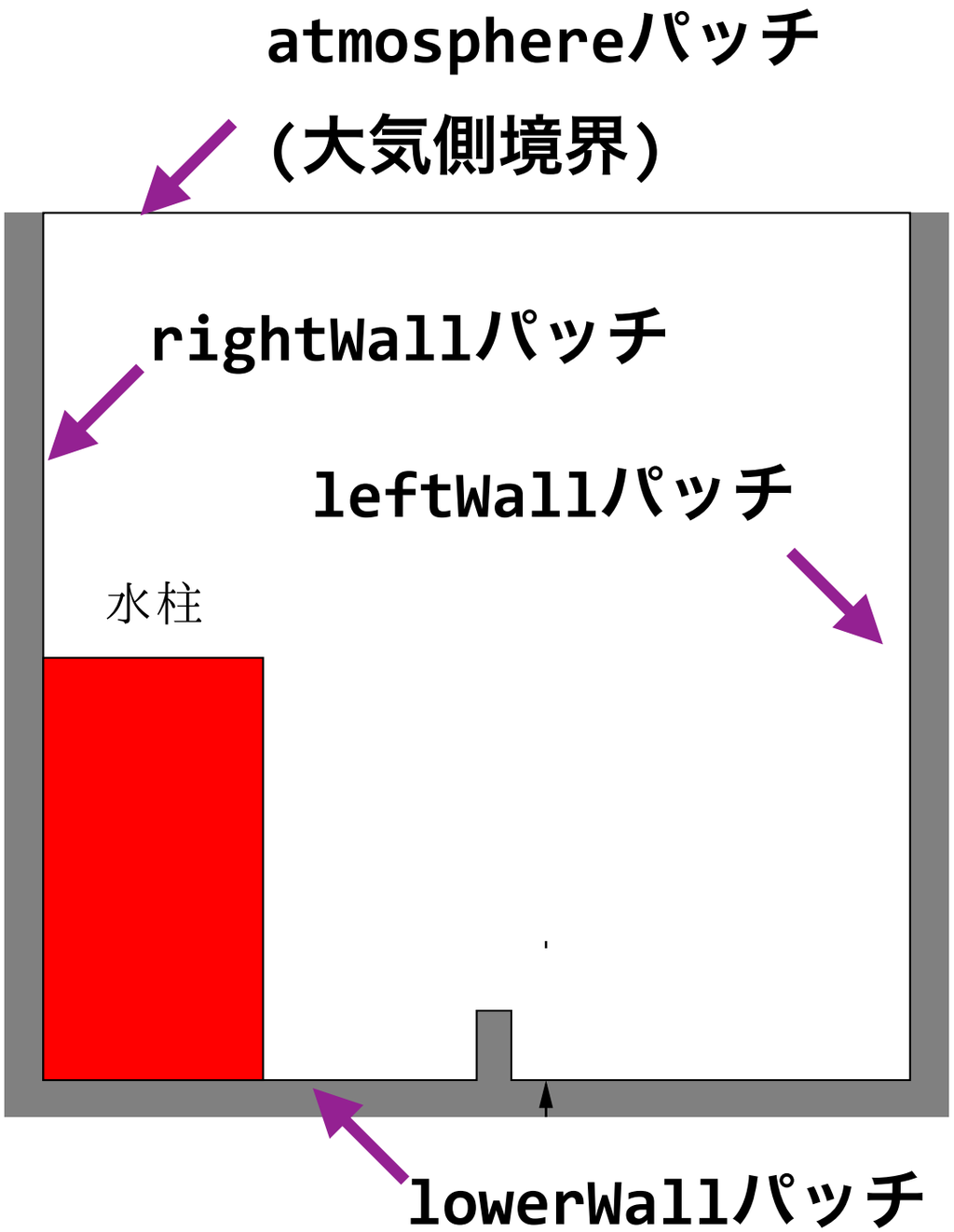


水相率 alpha.water

system/setFieldsDict

```
defaultFieldValues//デフォルト値
(
    volScalarFieldValue alpha.water 0
);

regions //領域指定
(
  boxToCell //矩形領域内のセルの値を設定
  {
    box (0 0 -1) (0.1461 0.292 1); //矩形領域
    fieldValues //領域内の値指定
    (
      volScalarFieldValue alpha.water 1
    );
  }
);
```

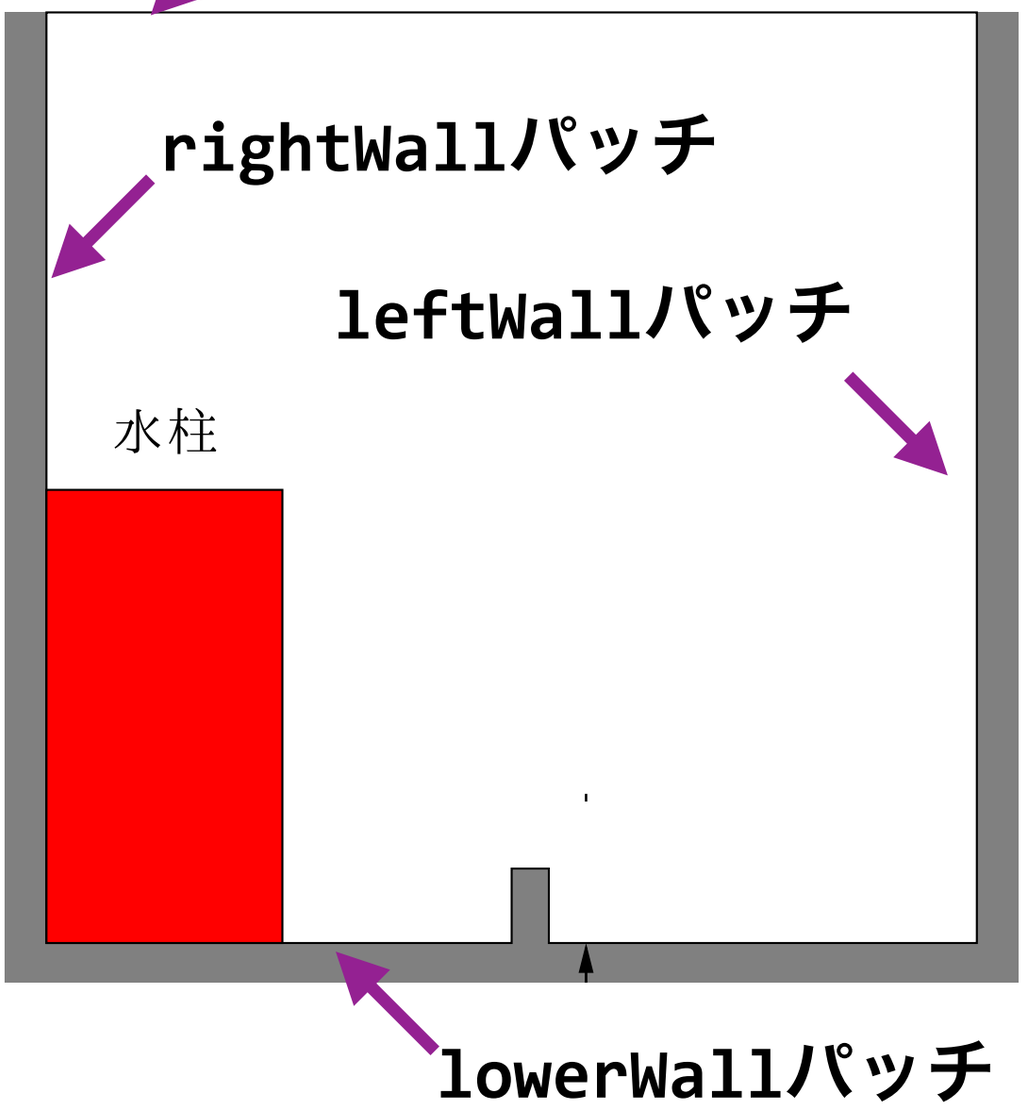
速度の境界条件



0/U

```
boundaryField //境界条件
{
:
    lowerWall //壁面(leftWall,rightWallも同様)
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    atmosphere //大気側
    {
        type      pressureInletOutletVelocity;
        value           uniform (0 0 0);
    }
}
```

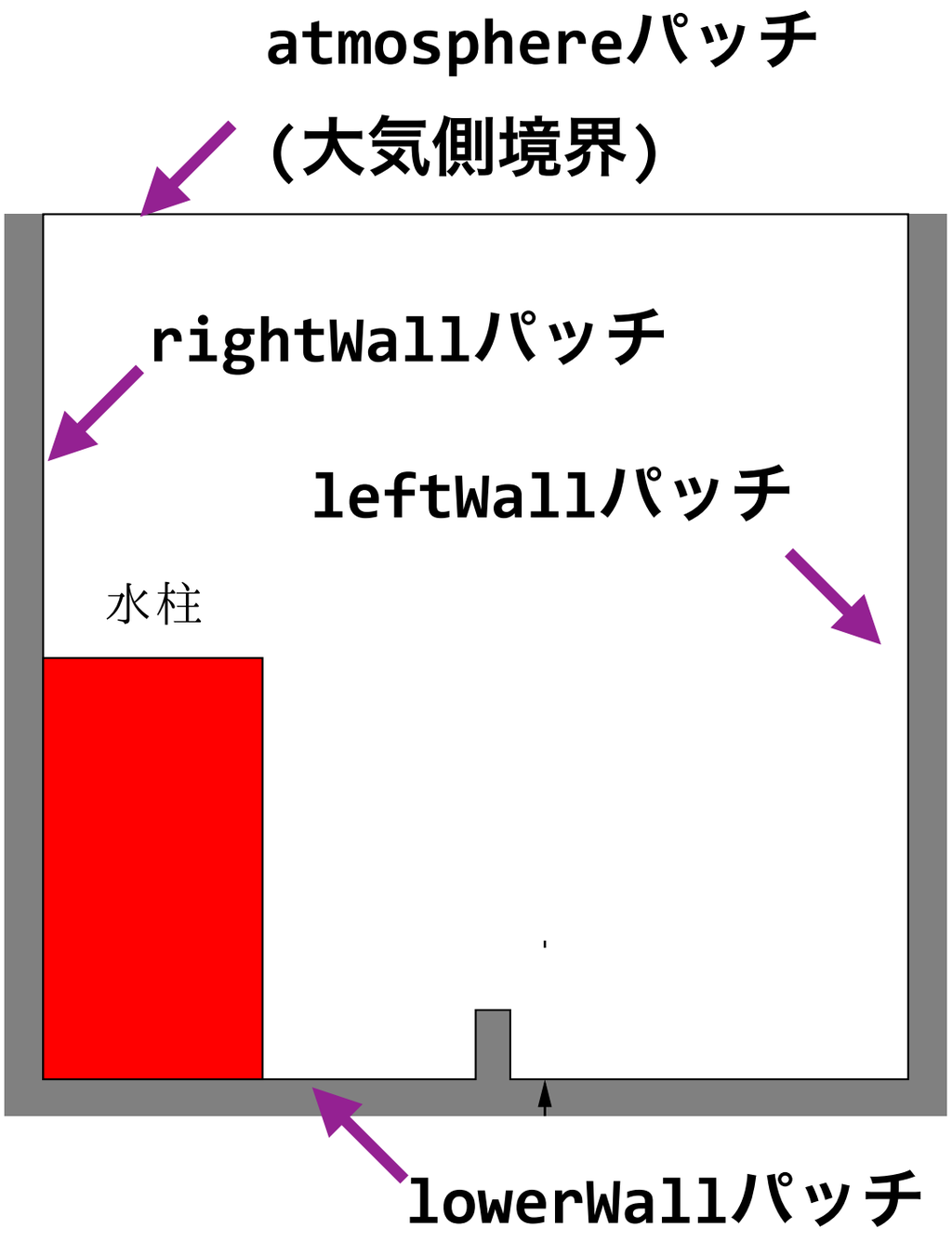
圧力の境界条件



0/p_rgh

```
boundaryField //境界条件
{
:
    lowerWall //壁面(leftWall,rightWallも同様)
    {
        type                fixedFluxPressure;
        value               uniform 0;
    }

    atmosphere //大気側
    {
        type                totalPressure;
        p0                  uniform 0;
        U                   U;
        phi                 phi;
        rho                 rho;
        psi                 none;
        gamma               1;
        value               uniform 0;
    }
}
```

水相率の境界条件



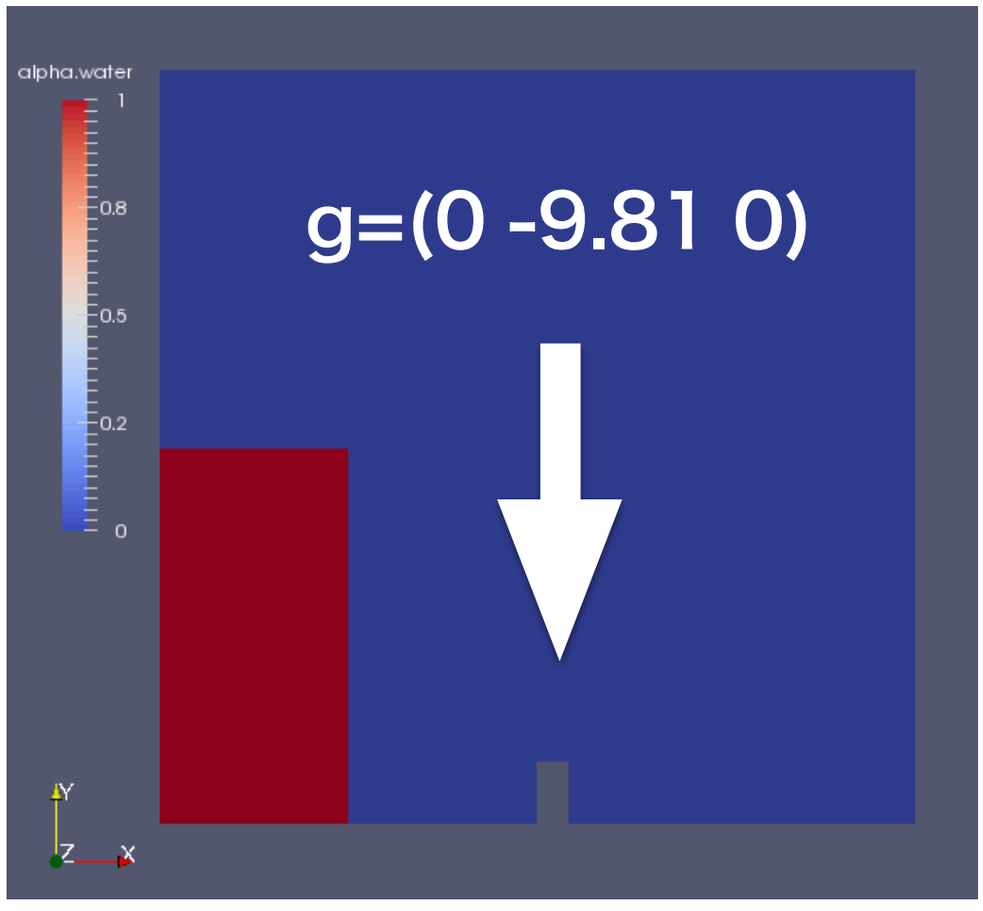
0/alpha.water

```
boundaryField //境界条件
{
:
    lowerWall //壁面(leftWall,rightWallも同様)
    {
        type                zeroGradient;
    }

    atmosphere //大気側
    {
        type                inletOutlet;
        inletValue          uniform 0;
        value               uniform 0;
    }
}
```

重力加速度・乱流モデルの設定

constant/g

```
dimensions      [0 1 -2 0 0 0 0]; //次元 [m/s²]
value           ( 0 -9.81 0 );    //重力加速度ベクトル(y方向)
```



constant/turbulenceProperties

```
simulationType  laminar;          //層流モデル(乱流モデルを使わない)
```

流体物性の設定

constant/transportProperties

```
phases (water air); //二相のラベル名リスト

water //相のラベル名(水)
{
    transportModel  Newtonian; //ニュートン流体
    nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06; //動粘性係数 [m²/s]
    rho             rho [ 1 -3 0 0 0 0 0 ] 1000; //密度[kg/m³]
//transportModelがNewtonianの場合，以下の2つのブロックは無視される
    CrossPowerLawCoeffs {…}
    BirdCarreauCoeffs {…}
}

air //相のラベル名(空気)
{
    transportModel  Newtonian;
    nu              nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;
    rho             rho [ 1 -3 0 0 0 0 0 ] 1;
}

sigma           sigma [ 1 0 -2 0 0 0 0 ] 0.07; //表面張力[N/m](=[kg/s²])
```

実行制御の設定

system/controlDict

```
Application     interFoam; //ソルバ名

startTime       0;         //解析開始時刻 [s]

endTime         1;         //解析終了時刻 [s]

deltaT          0.001;     //時間刻み [s] (時間刻みの自動制御の場合，初期値)

writeInterval   0.05;      //解析結果保存間隔 [s]

adjustTimeStep  yes;       //時間刻みの自動制御

//以下は時間刻みの自動制御用のパラメータ
//以下を満すように時間刻みが各ステップで自動制御される
maxCo           1;         //最大クーラン数
maxAlphaCo      1;         //相界面(0<相率<1)での最大クーラン数

maxDeltaT       1;         //最大時間刻み
```

damBreakFineケースの解析手順

1. 格子生成

`blockMesh`

2. 水相率の場作成

```
cp 0/
alpha.water.org
 0/alpha.water
```

3. 水相率分布設定

`setFields`

この他にも可視化等のポスト処理が通常含まれる

constant/polyMesh/
- blockMeshDict　構造格子設定ファイル
- boundary　パッチ設定(上書される)
- faces, neighbour, owner, points 格子データ

0/
- U　速度ベクトル場
- alpha.water.org　水相率場(分布無し)
- alpha.water　水相率場
- p_rgh　静水圧を引いた圧力場

constant/
- g　重力加速度
- transportProperties　流体物性
- turbulenceProperties　乱流モデル

system/
- setFieldsDict　場の初期分布設定
- controlDict　実行制御の設定
- fvSchemes　離散化スキームの設定
- fvSolution　ソルバの設定

解析結果の時刻ディレトリ/
- U, alpha.water, p_rgh, phi　解析結果

読み込み

読み書き

新規作成
fileName

既存ファイル
fileName

修正ファイル
fileName

作成ファイル

4. ソルバ実行

`interFoam`

ダムブレイク流れの解析手順

ユーザーマシン

4.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子・解析結果転送
[rsync]

ログインノード
PRIMERGY RX300

2.格子生成・水相率分布設定・流体解析
[pjsub,
blockMesh, setFields, interFoam]

同一のジョブ内で連続して実行

計算ノード
PRIMEHPC FX10

バッチジョブシステムによりジョブが実行される

実行用ジョブスクリプトとジョブ投入

Allrun.PJM（逐次ジョブ用スクリプト）

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture" #リソースグループ指定(演習中はtutorial)
#PJM -g gt00 #グループ名指定(適宜変更)
#PJM -L "node=1" #ノード数指定
#PJM -j #ジョブの標準エラー出力を標準出力へ出力
#------- Program execution -----#
module load OpenFOAM/2.3.0 #OpenFOAM/2.3.0のmoduleの設定
source $WM_PROJECT_DIR/etc/bashrc #OpenFOAMの環境設定

blockMesh > log.blockMesh #格子生成
cp 0/alpha.water.org 0/alpha.water #水相率の場作成
setFields > log.setFields #水相率分布設定
interFoam > log #ソルバ実行
```

ジョブの投入

```
pjsub Allrun.PJM
```

格子生成ログ

```
more log.blockMesh
```

```
Mesh Information
----------------
  boundingBox: (0 0 0) (0.584 0.584 0.0146)   解析領域範囲
  nPoints: 15774                              節点数
  nCells: 7700                                格子数
  nFaces: 30986                               界面(フェース)数
  nInternalFaces: 15214                       内部界面(フェース)数
----------------
Patches                                       パッチ(境界面)情報
----------------
  patch 0 (start: 15214 size: 86) name: leftWall
  patch 1 (start: 15300 size: 86) name: rightWall
  patch 2 (start: 15386 size: 110) name: lowerWall
  patch 3 (start: 15496 size: 90) name: atmosphere
  patch 4 (start: 15586 size: 15400) name: defaultFaces

End
```

ソルバのログ確認

```
more log
```

```
Create time

Create mesh for time = 0


PIMPLE: Operating solver in PISO mode

Reading field p_rgh

Reading field U

Reading/calculating face flux field phi

Reading transportProperties

Selecting incompressible transport model Newtonian
Selecting incompressible transport model Newtonian
--続ける--(0%)
```

moreコマンド名の主な操作キー　SPC：前、b：後、h：ヘルプ、q：終了

ソルバのログ確認（続き）

```
Courant Number mean: 0.0814093 max: 0.836649
```
クーラン数統計値

```
Interface Courant Number mean: 0.0083422 max: 0.836649
```
界面でのクーラン数統計値

```
deltaT = 0.00158401
```
時間刻み（クーラン数の最大値が1以下になるように自動調整）

```
Time = 0.15
```
時刻（0.05[s]毎の解析結果の保存時刻に厳密に合うように時間刻みが調整される）

```
smoothSolver:  Solving for alpha.water, Initial residual = 0.0038427, Final residual =
1.13717e
-09, No Iterations 3
```
alpha.waterの離散方程式についての線形ソルバのログ，smoothSolver：線型ソルバ名，Initial residual：初期残差，Final residual：最終残差，No Iterations：反復回数

```
Phase-1 volume fraction = 0.130194  Min(alpha1) = -2.02242e-09  Max(alpha1) = 1
```
水相率統計

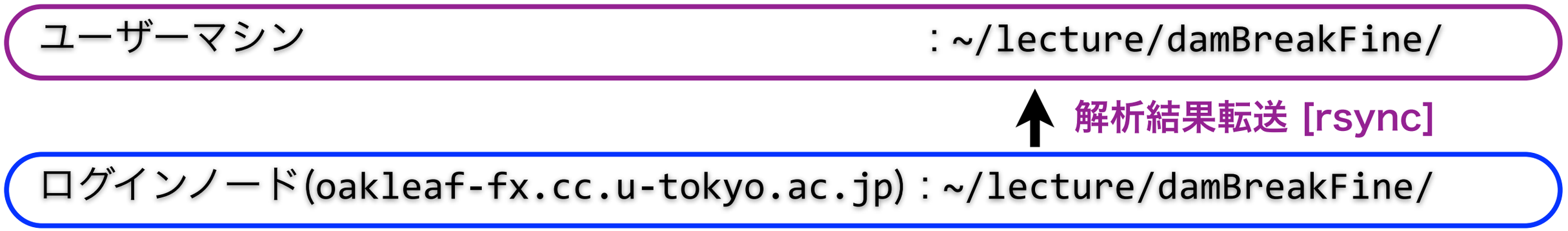
```
:
DICPCG:  Solving for p_rgh, Initial residual = 9.88248e-05, Final residual = 7.85985e-
08, No Itrations 44
```
p_rghの離散方程式(圧力の*Poisson*方程式)についての線形ソルバのログ

```
time step continuity errors : sum local = 4.22458e-08, global = 3.23282e-09,
cumulative = 0.000382964
```
質量保存式の残差，sum local ： 誤差絶対値の格子体積重み付け平均，global ： 誤差(符号あり)の格子体積重み付け平均，cumulative ： globalの累積

確認後qキーで終了

```
ExecutionTime = 0.77 s  ClockTime = 1 s
```
ExecutionTime：計算のみに要した時間，ClockTime ：ファイルI/Oなども含めた実際の経過時間

解析結果の転送



解析結果の転送

ユーザー名@ユーザーマシン ~/lecture/cavity

```
$ cd ../damBreakFine
```

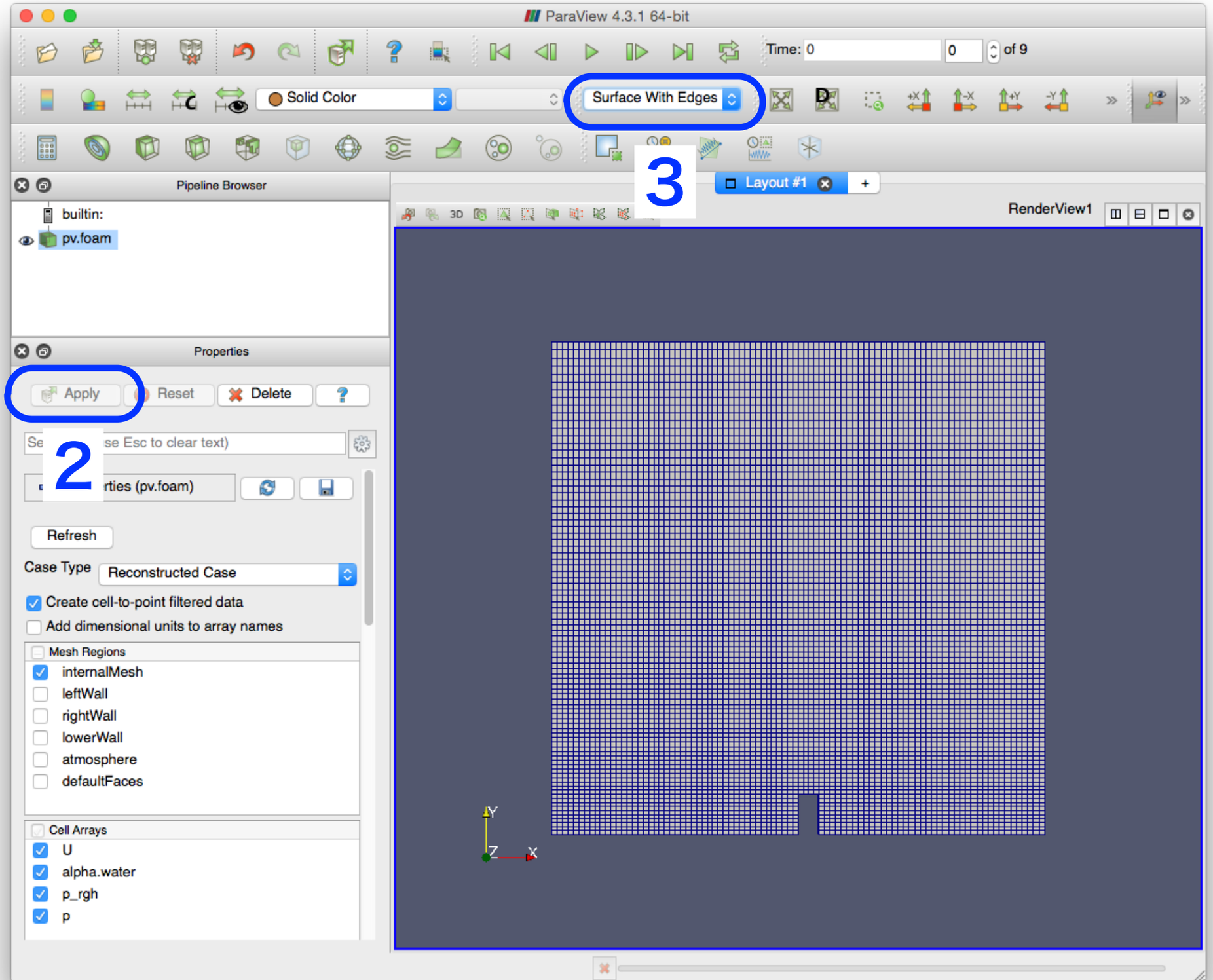
ユーザー名@ユーザーマシン ~/lecture/damBreakFine

```
$ rsync -auv tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/lecture/damBreakFine/ ./
```

↑(カーソル上)を押して前のコマンドを呼び出し(ヒストリ機能)，書き変えるのが楽

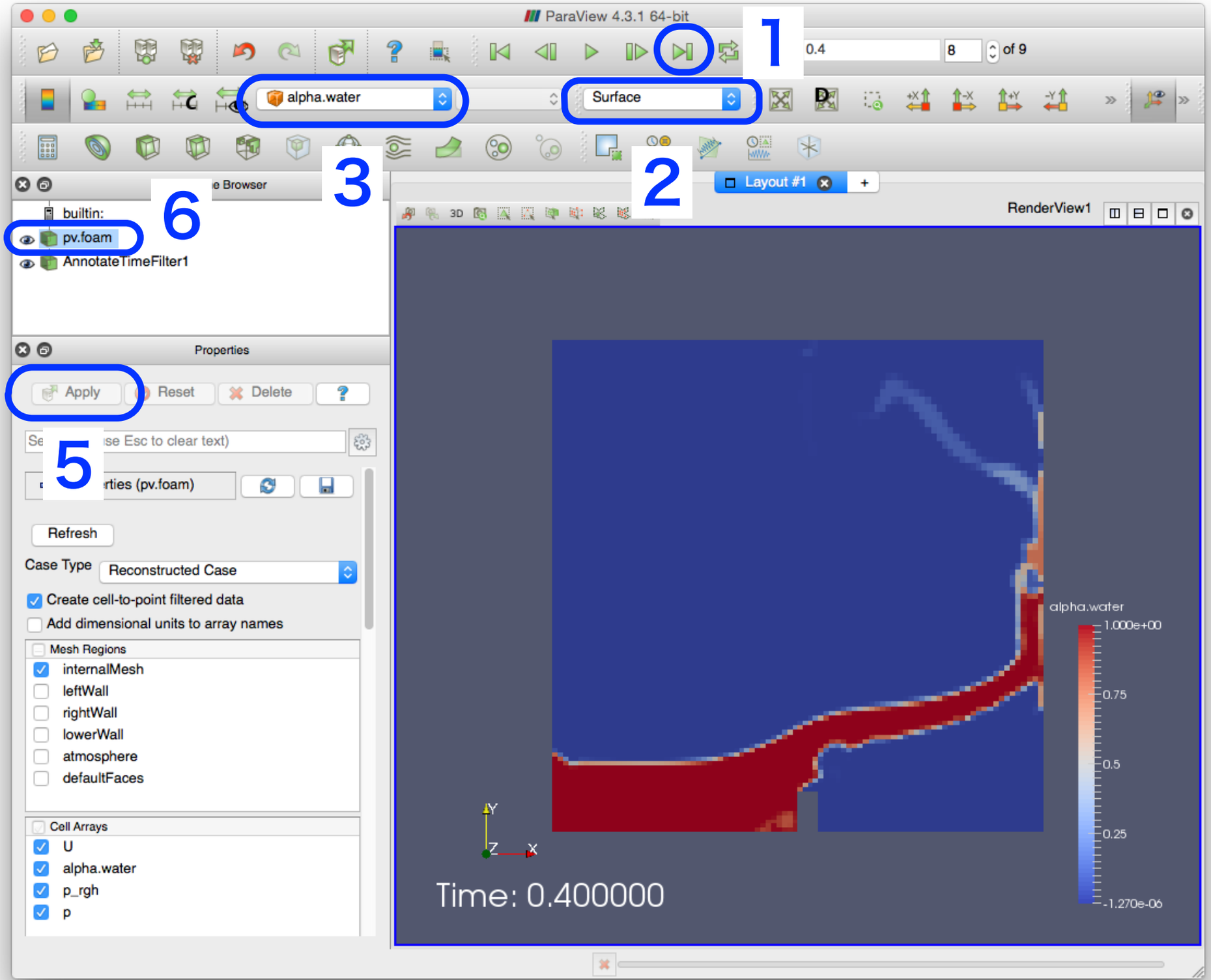
```
$ touch pv.foam ParaViewでの可視化用にpv.foamを作成しておく
```

ParaViewによる格子の可視化



1. Fileメニュー/Open/damBreakFineのディレクトリのpv.foamを開く
2. Apply
3. Representation/Surface With Edges 選択

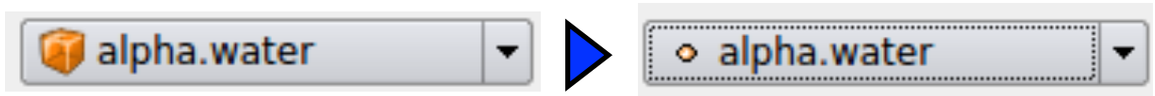
ParaViewによる計算結果の可視化



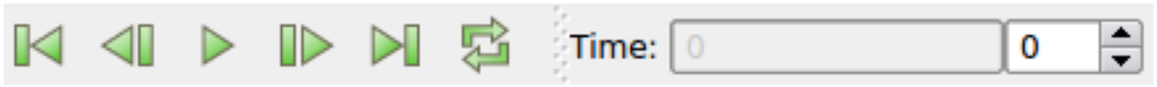
1. Last Frame
2. Representation/ Surface 選択
3. Coloring/ alpha.water 選択
4. Filtersメニュー/ temporal/ Annotate Time Filter選択
5. Apply
6. pv.foam 選択

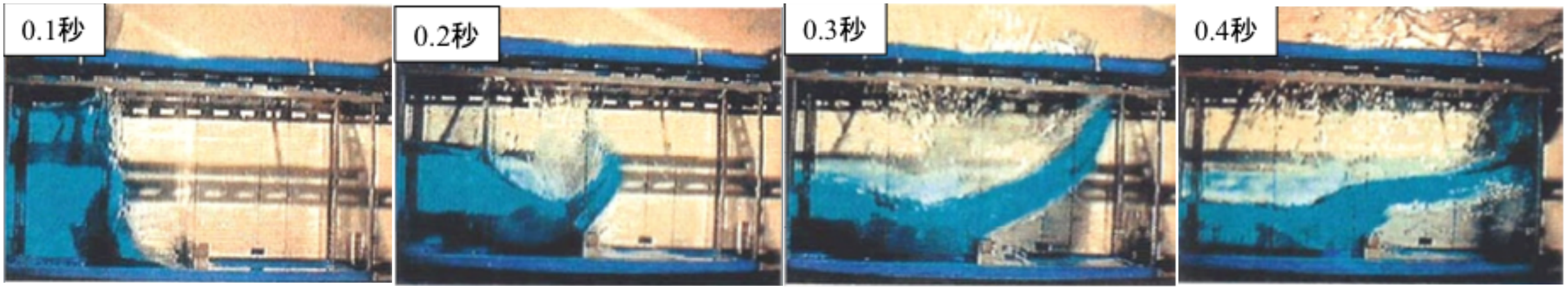
演習課題

課題1: 水相率を補間して表示させる.



課題2: コマ送りして，0.1秒毎の実験可視化画像[1]と比較する.





課題3: 並列計算の結果と比較するため，全時間ステップの可視化画像を保存する.

方法：Fileメニュー/Save Animation/Save Animationボタン/File name: serial,
Files of type: PNG(静止画のほうが比較が容易であるため)/OK.
serial.連番.pngのPNGファイルが作成されるので，画像ビューワ等で表示する.

[1] Koshizuka, S., H. Tamako and Y. Oka :“A Particle Method for Incompressible Viscous Flow with Fluid Fragmentation”, CFD Journal ,Vol.4, No.1, pp.29-46, 1995

並列計算演習

OpenFOAMの並列計算手法

1. 格子生成
2. 領域分割 (decomposePar)
3. MPI並列でソルバを実行

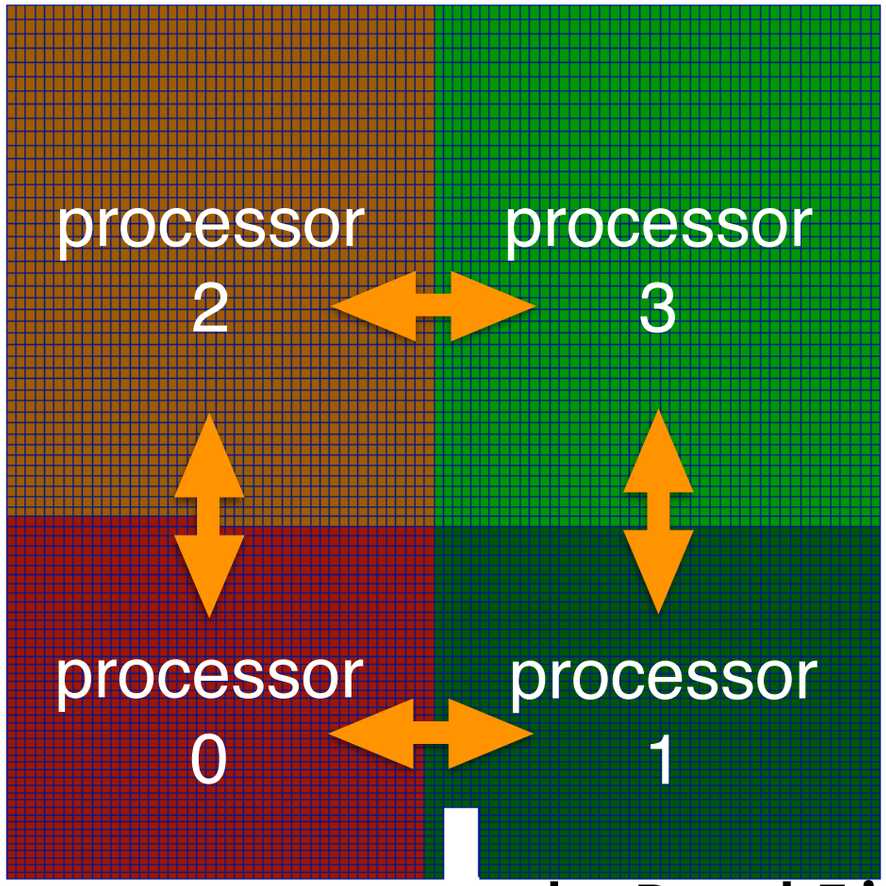
(MPI+OpenMPのハイブリット並列は標準では未実装．ただし，櫻井，片桐らによる線型ソルバのスレッド並列化(1MPI+OpenMP)の研究例有り[1]．現在、MPI+OpenMP実装を開発中)

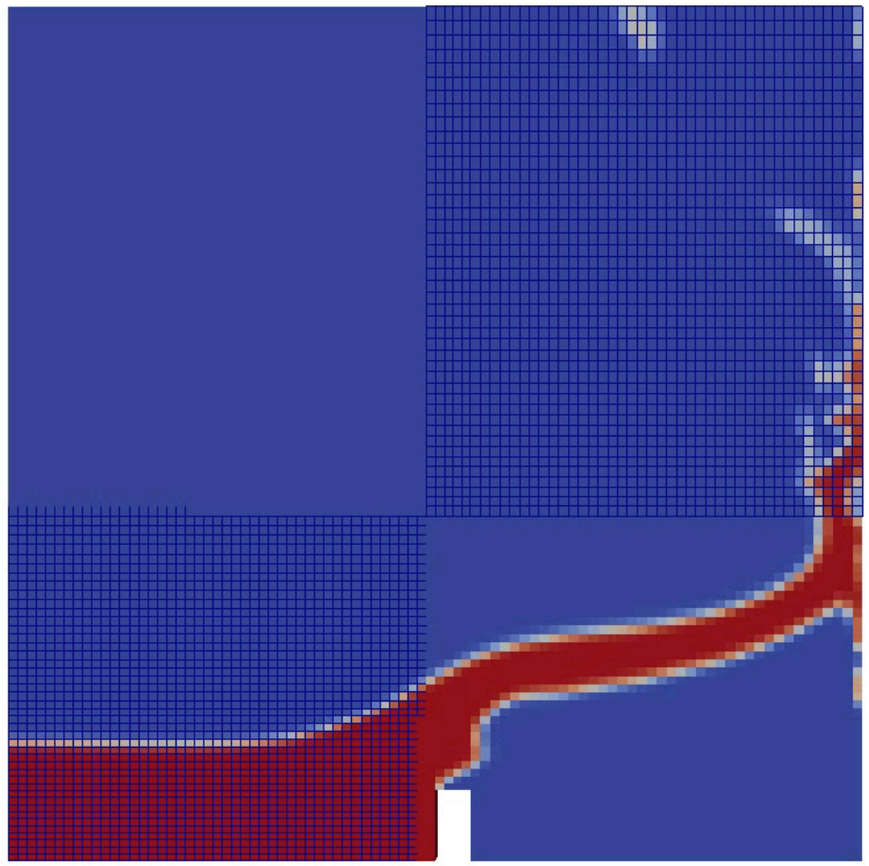
4. 領域毎の解析結果を再構築 (reconstructPar)

[1]櫻井，片桐ら「OpenFOAMへの疎行列計算ライブラリXabclibの適用と評価」オープンCAEシンポジウム2014

プロセッサ間MPI通信(フラットMPI通信)

プロセッサ間通信は通常ボトルネックとなる





damBreakFineチュートリアル

ダムブレイク流れの解析手順

ユーザーマシン

4.可視化
[ParaView]

1. ログイン
[ssh]

3. 格子・解析結果転送
[rsync]

ログインノード
PRIMERGY RX300

2.格子生成・水相率分布設定・**領域分割**・流体解析・**再構築**
[pjsub,blockMesh, setFields,
decomposePar, interFoam, ***recconstructPar***]

同一のジョブ内で連続して実行

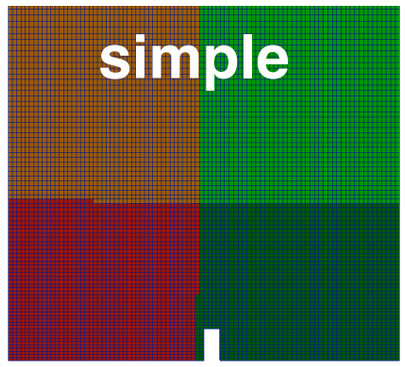
計算ノード
PRIMEHPC FX10

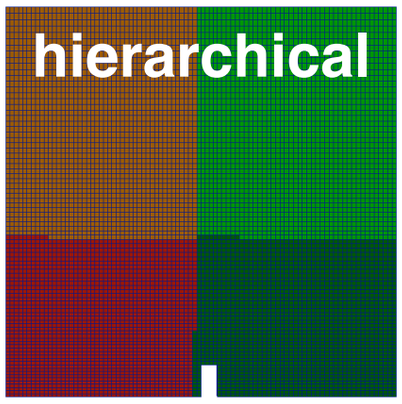
バッチジョブシステムにより指定されたノード数の
計算ノードでジョブが実行される

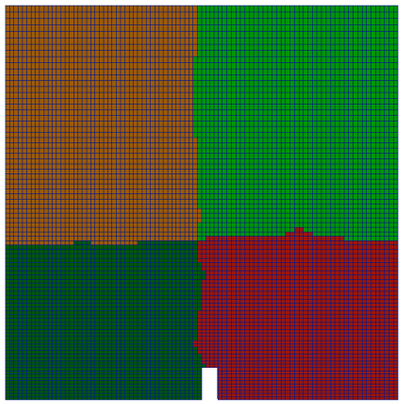
領域分割の設定

system/decomposeParDict

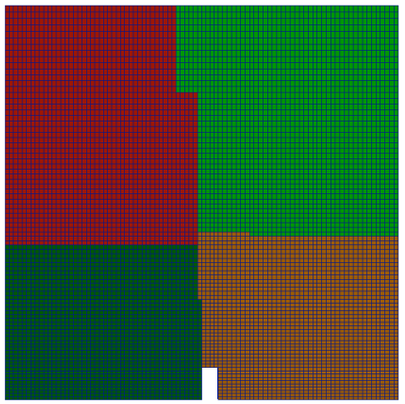
```
numberOfSubdomains 4;          //領域分割数

method              simple;    //領域分割方法

simpleCoeffs//単純に軸方向に分割
{
    n         ( 2 2 1 ); //分割数
    delta     0.001;
}

hierarchicalCoeffs //分割方向の順番を指定
{
    n         ( 2 2 1 );
    order     xyz; //分割方向の順番
    delta     0.001;
}
```







metis



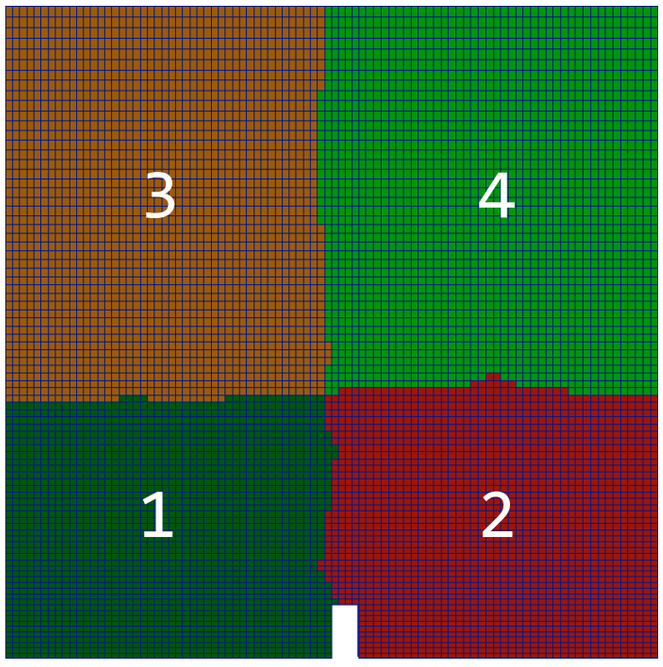
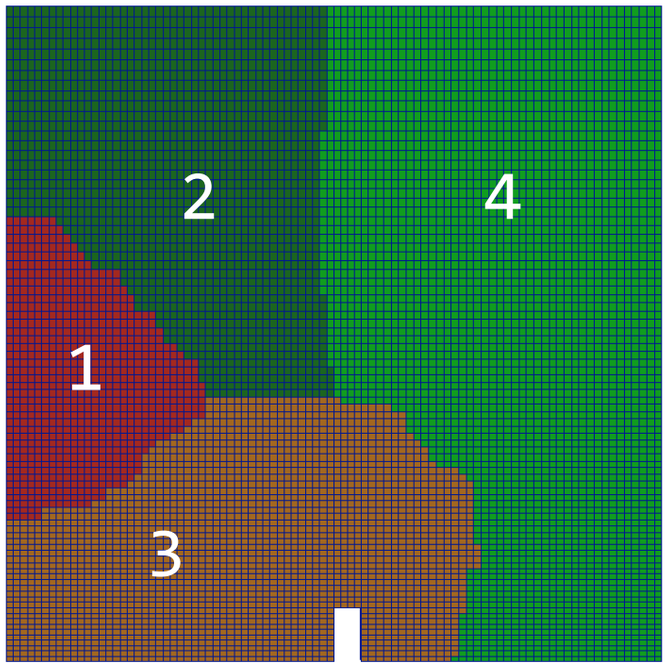
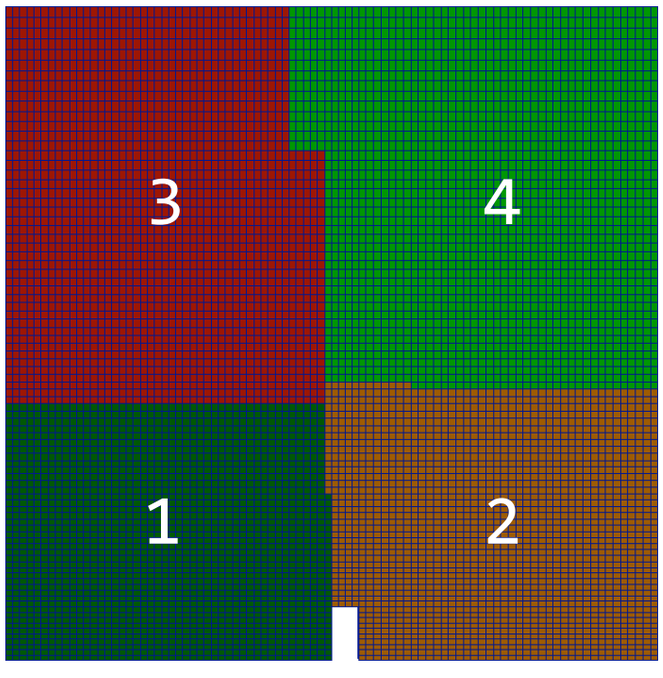
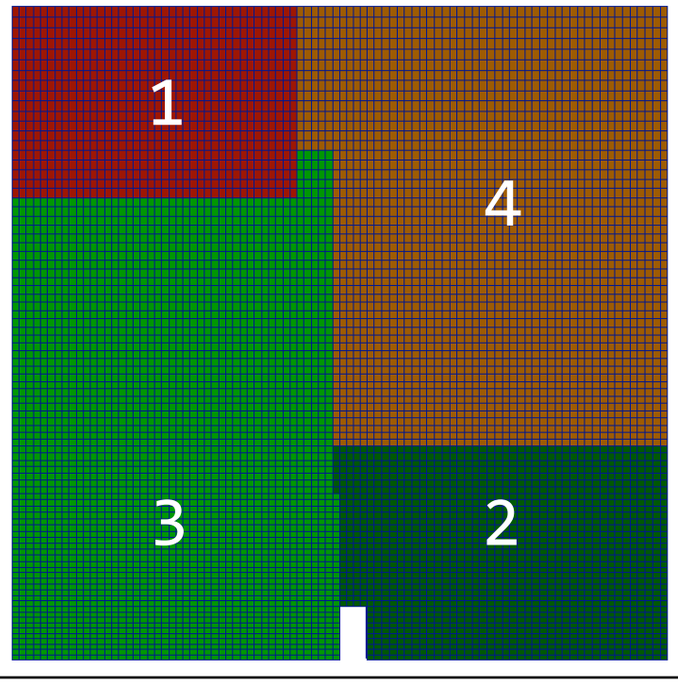
scotch

metis：Metisライブラリを使用。プロセッサ間の通信量に大きく影響する分割領域間の界面数を最小化。ライセンスにより商用利用や再配布が自由ではない

scotch：Scotchライブラリを使用。フリーソフトライセンスでMetisと互換APIを持つ

manual：格子を割り充てるプロセッサを手動で指定

重み付き領域分割例

	重み無し	重み付き
metis		
scotch		

```
scotchCoeffs
または
metisCoeffs
{
   processorWeights
   ( 1 2 3 4 );
   //プロセッサ毎の
格子数の重み係数
}
```

実行用ジョブスクリプトとジョブ投入

Allrun-p.PJM（フラットMPI並列ジョブ用スクリプト，赤字が並列用）

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture" #リソースグループ指定(演習中はtutorial)
#PJM -g gt00 #グループ名指定(適宜変更)
#PJM -L "node=1" #ノード数指定(もちろん複数でも可．本講習会では12以内)
#PJM --mpi "proc=4" #プロセス数指定(MPI並列数を指定する．本講習会では192以内)
#PJM -j #ジョブの標準エラー出力を標準出力へ出力
#------- Program execution -------#
module load OpenFOAM/2.3.0
source $WM_PROJECT_DIR/etc/bashrc

blockMesh > log.blockMesh #格子生成
cp 0/alpha.water.org 0/alpha.water #水相率の場作成
setFields > log.setFields #水相率分布設定
decomposePar > log.decomposePar #領域分割
mpiexec -stdout log-np4 -np 4 interFoam -parallel #MPI並列実行．ログ名は任意
reconstructPar > log.reconstructPar #再構築
```

領域分割のログの確認

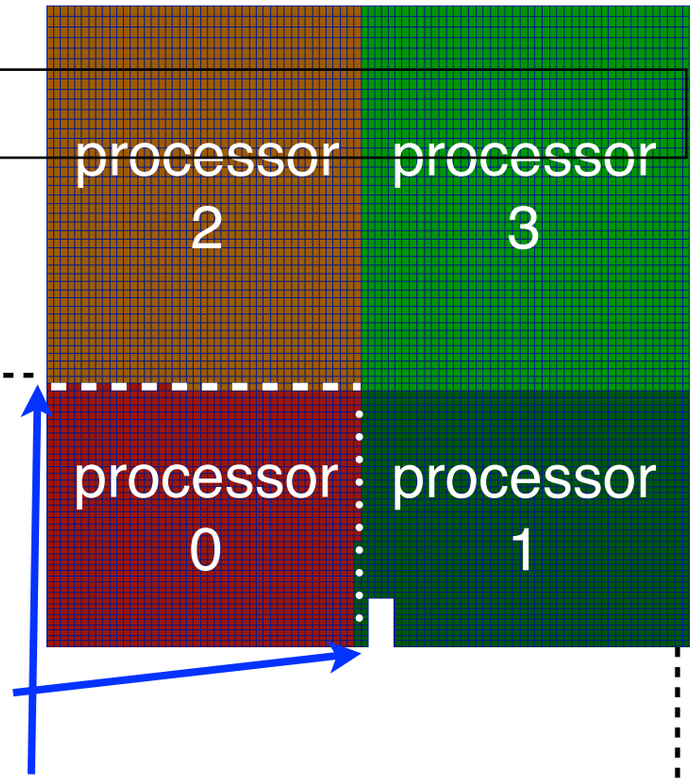
ジョブの投入

```
pjsub Allrun-p.PJM
```

領域分割のログの確認

```
more log.decomposePar
```

```
(略)
Processor 0 プロセッサ0の担当分割領域
    Number of cells = 1935 格子数
     プロセッサ番号1の担当分割領域との共有界面数(以下同様)
    Number of faces shared with processor 1 = 44
    Number of faces shared with processor 2 = 46
    Number of processor patches = 2 上記の界面を共有するプロセッサ数
    Number of processor faces = 90 上記の共有界面数の合計(=44+46)
    Number of boundary faces = 3958 この領域における境界面の界面の合計

Processor 1 プロセッサ1の担当分割領域(ディレクトリprocessor1へ出力)
    Number of cells = 1915 格子数(プロセッサ0の格子数とほぼ同数)
(以下同様)
```



領域分割の実行 (続き)

(略)

```
Number of processor faces = 178
```
共有界面数の総数(小さいほうが良い)

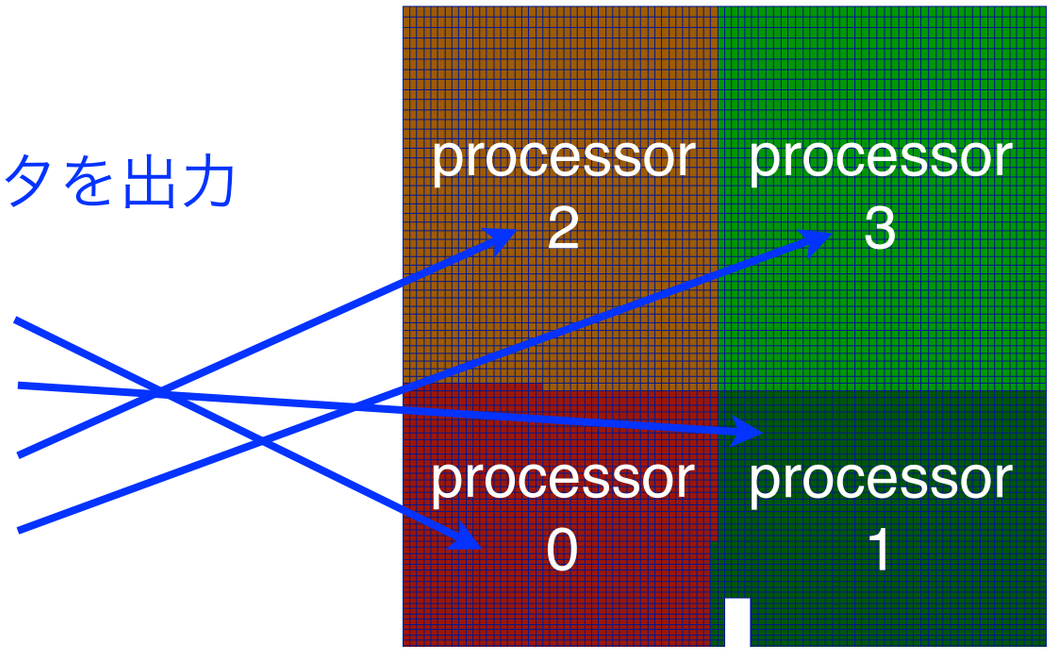
以下、全プロセッサ担当分割領域における各種統計値

プロセッサの計算能力が同等な場合、以下の量はバラツキが無いほうが良い

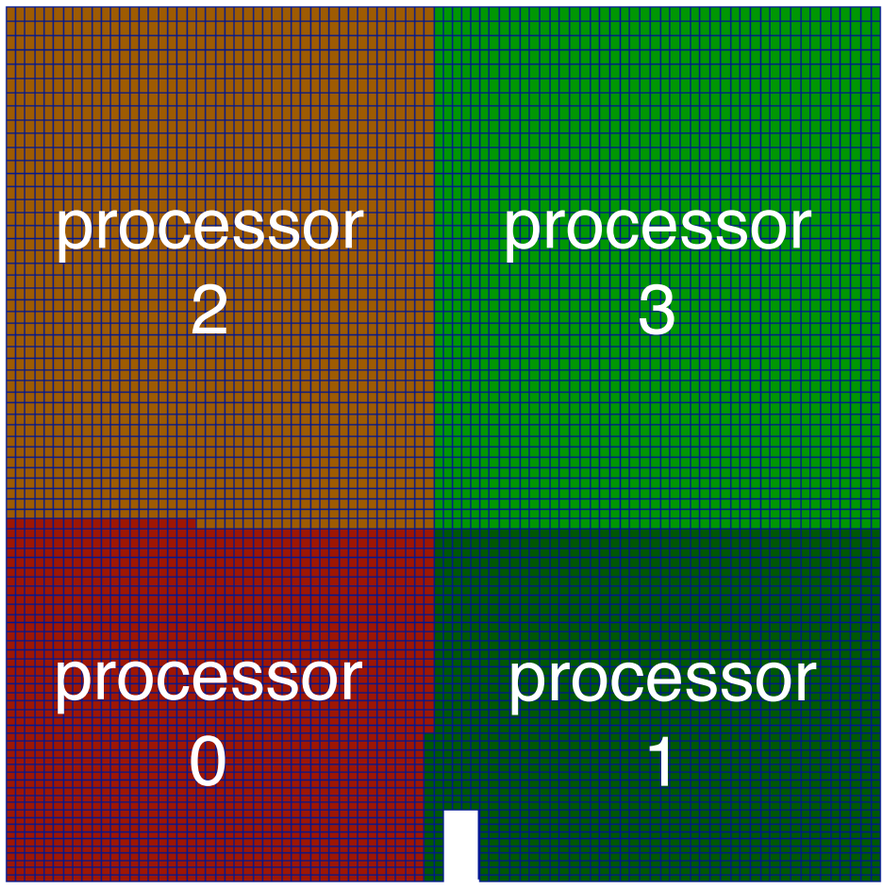
```
Max number of cells = 1935 (0.519481% above average 1925)
Max number of processor patches = 2 (0% above average 2)
Max number of faces between processors = 90 (1.1236% above average 89)

Time = 0
```

プロセッサ0のディレクトリに場データを出力

```
Processor 0: field transfer
Processor 1: field transfer
Processor 2: field transfer
Processor 3: field transfer

End.
```



領域分割結果




```
constant/
  polyMesh/ 格子データ
0/ 時刻ディレクトリ(初期値)
  alpha.water,U,p_rgh
processor0/ プロセッサディレクトリ
  constant
    polyMesh/ 領域分割後格子データ
      0/ 領域分割後フィールドデータ
        alpha.water,U,p_rgh
processor1/ 以下processor0と同様
processor2/
processor3/
```

領域分割前のデータ

領域分割により生成されたデータ

並列計算時のログ

ログ中のプロセッサ数や通信オプションを確認

```
more log-np4
```

```
nProcs : 4  プロセッサ数
Slaves :
3           マスタープロセスに従属するスレーブプロセス数(プロセッサ数-1)
( “ホスト名.PID(プロセスID)”
"a01t70081.21156" ホスト名やPIDは実行環境に応じて変わります
"a01t70081.21157"
"a01t70081.21158"
)
Pstream initialized with: 通信条件(以下がデフォルト．詳細は省略)
    floatTransfer      : 0
    nProcsSimpleSum    : 0
    commsType          : nonBlocking
    polling iterations : 0
```

意図したプロセッサ・ノード数で動いているか確認する

並列計算結果の再構築

並列計算時の解析結果

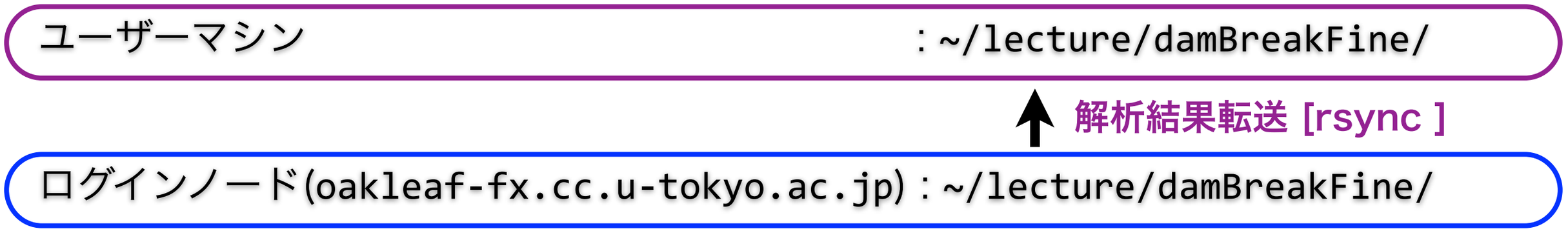
processor0/ プロセッサディレクトリ
  0.05/ 分割領域の時刻ディレクトリ
    alpha*,U,p,p_rgh,phi
  :
  0.4/
processor1/
  0.05/
    alpha*,U,p,p_rgh,phi
processor2/
  0.05/
    alpha*,U,p,p_rgh,phi
processor3/
  0.05/
    alpha*,U,p,p_rgh,phi

分割済み解析結果

再構築後の解析結果
(通常と同じ場所)

0.05/ 時刻ディレクトリ
  alpha*,U,p,p_rgh,phi
0.1/
  alpha*,U,p,p_rgh,phi
  :
0.4/
  alpha*,U,p,p_rgh,phi

解析結果の転送

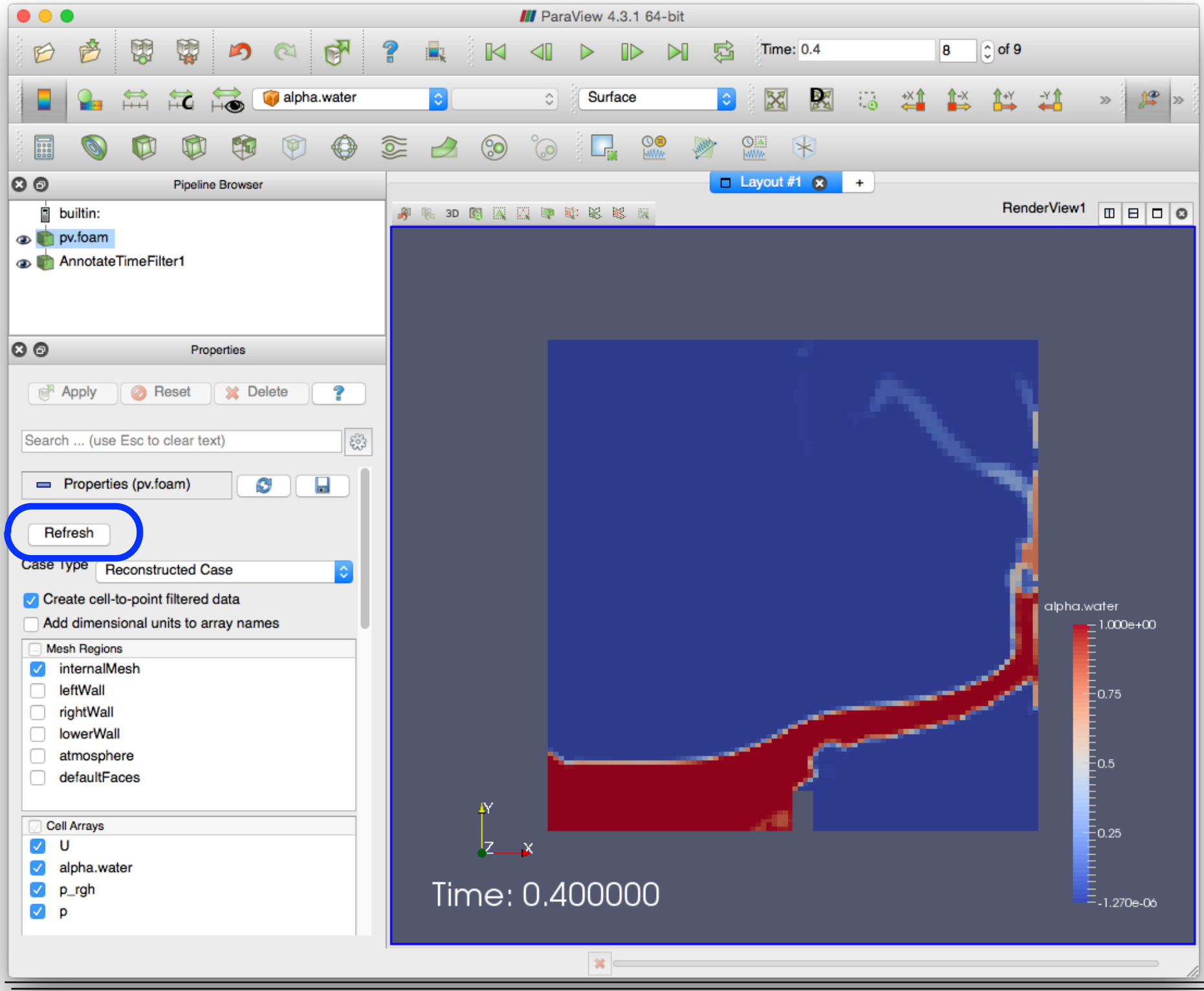


解析結果の転送

ユーザー名@ユーザーマシン ~/lecture/damBreakFine

```
$ rsync -auv tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/lecture/damBreakFine/ ./
```

ParaViewによる計算結果の可視化



並列計算の結果に更新されたので，Refreshボタンを押して更新する

1. Refresh

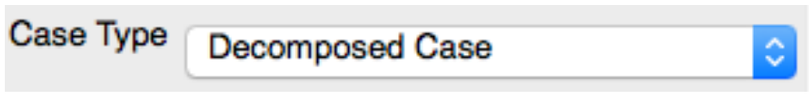
演習課題

課題1：全時間ステップの可視化画像を保存し，　非並列計算の画像と比較する.

方法：Fileメニュー/Save Animation/Save Animationボタン/File name: parallel，　Files of type: PNG(静止画のほうが比較が容易であるため)/OK.
parallel.連番.pngのPNGファイルが作成されるので，画像ビューワ等で表示する.

課題2：ParaViewは領域分割されたケース(Decomposed Case)のデータも可視化できるので，再構築前の領域分割されたデータを可視化する.

方法：Case Type/Decomposed Caseを選択してApply



課題3：Decomposed Caseの可視化画像を保存して，課題1で作成した再構築したケース(Reconstructed Case)の可視化画像と比較する.

課題4：並列計算のスピードアップ率および並列化効率(Strong scaling)を求める.

方法：非並列計算のログ(log)および並列計算のログ(log-np4)における，最後の時間ステップのExecutionTimeをそれぞれt1,t4として以下．効率が悪い主な原因を推測する.

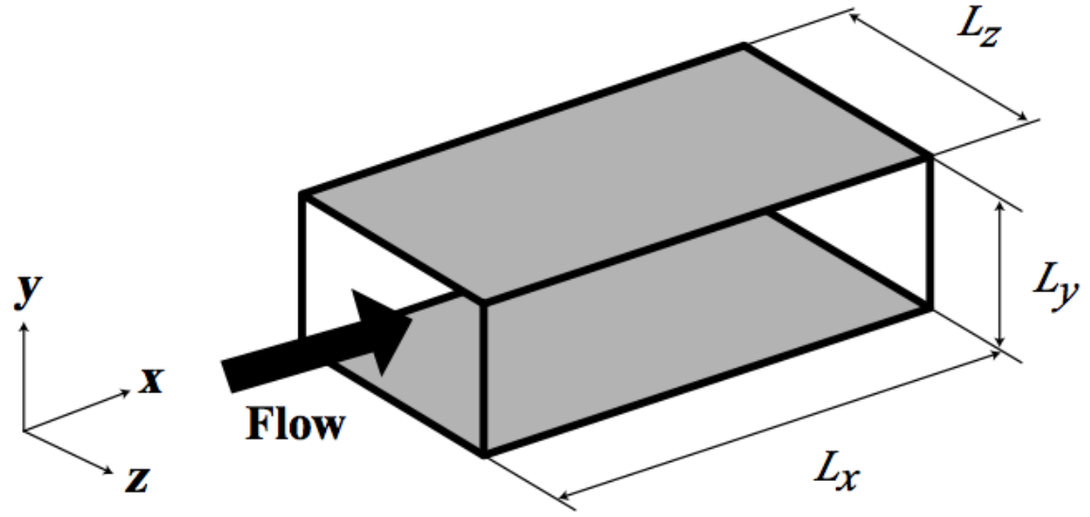
　スピードアップ率=t1/t4，　並列化効率=(t1/t4)/4

チャネル流れ演習

実行性能・並列化効率評価

チャネル流れケースchannelReTau110

- 摩擦速度とチャネル半幅で無次元化されたレイノルズ数が110のチャネル流れを非圧縮性の流体解析ソルバpimpleFoamを用いて解析するケース
- IwamotoらによるDNS計算のデータベース[1]がWEBで公開されているので，検証(Validation)が容易



図出典：松尾裕一，チャネル乱流，日本流体力学会，ながれ22，pp.35-40, 2003

解析条件

$L_x \times L_y \times L_z = 5\pi \times 2 \times 2\pi$

$Re_\tau = u_\tau \, \delta / \mu \; = 110$ [-]

ここで

u_τ：壁面摩擦速度 [m/s]

δ ：チャネル半幅 [m] ($=L_y/2$)

μ ：動粘性係数 [m^2/s^2]

主流方向(x)に一定の圧力勾配

主流方向(x)およびスパン方向(z)は周期境界

[1] K Iwamoto, K., 2002."Database of Fully Developed Channel Flow,"THTLAB Internal Report, No. ILR-0201.THTLAB, Dept. of Mech. Eng., The Univ. of Tokyo.

http://thtlab.jp/DNS/dns_database.html

台数効果と並列化効率

- 並列計算による台数効果 (スピードアップ) S_P

$$S_P = T_S / T_P$$

ここで

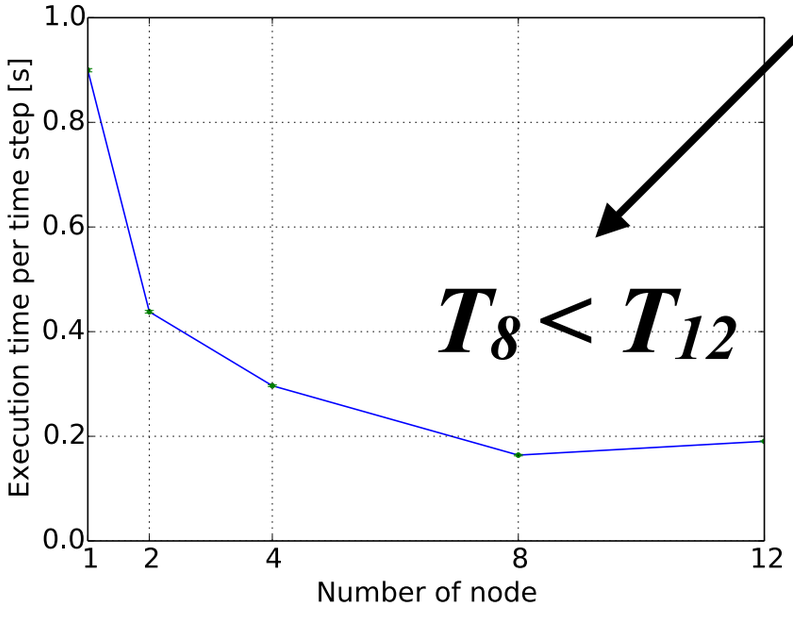
T_S ： 1コア(または1ノード)での実行時間

T_P ： Pコア(またはPノード)での実行時間

- 並列化効率 E_P

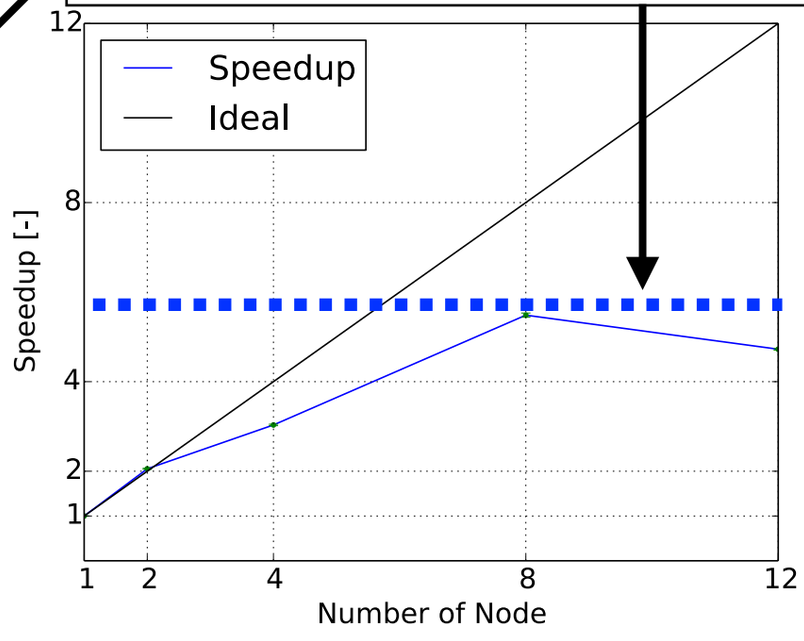
$$E_P = S_P / P \times 100\ [\%]$$

この問題では12ノード以上使用しても非効率
→台数効果の飽和(Saturation)をベンチマークテストで事前に把握し，非効率な計算を行わないことが重要

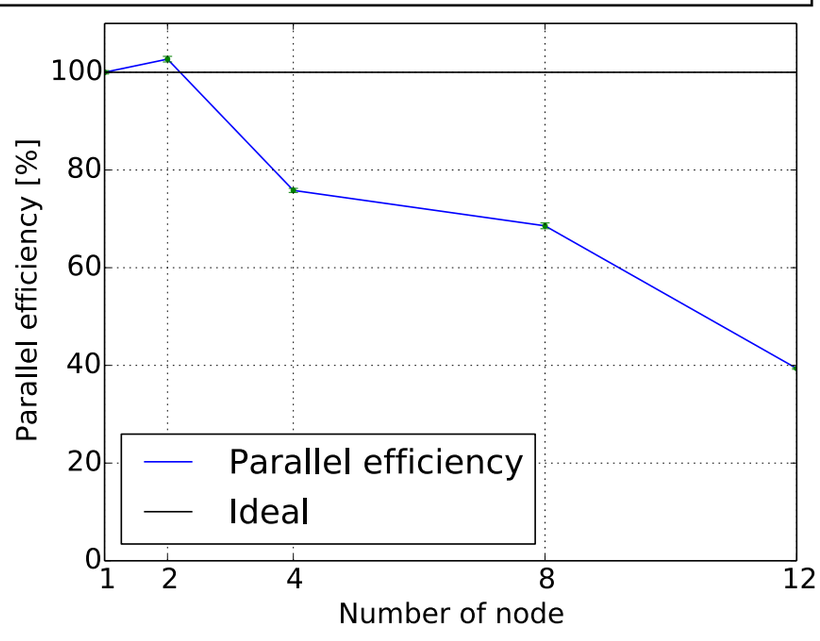


$T_8 < T_{12}$

実行時間



台数効果(スピードアップ)



並列化効率

ベンチマークテスト

- ベンチマークテスト：プログラム実行時間やFLOPS値などの性能指標の計測
- 並列計算機でのベンチマークテストは重要
  - ✓ 並列数を変更させて，台数効果(スピードアップ)や並列化効率を調べ，効率の良い並列数を決定できる
  - ✓ 時間ステップ数や反復数が小さい予備計算で検討するのが効率的
- OpenFOAM等のCFDコードでは圧力線型ソルバのベンチマークテストも重要
  - ✓ 実行時間は圧力線型ソルバの種類や前処理方法に強く依存
  - ✓ 線型ソルバーの速さは並列数にも依存
    - 並列数小→AMG(代数マルチグリッド) > PCG(前処理付き共役勾配法)
    - 並列数大→PCG > AMG

チャネル流れケースを対象に，少ない時間ステップ数で

格子数，並列数，圧力線型ソルバを変化させたベンチマークテストを行う

ベンチマークケースディレクトリ

ベンチマークケースケースのディレクトリへ移動

```
cd ~/lecture/channelReTau110
```

ファイル・ディレクトリ構成を表示 (主なファイルのみ)

```
tree
```

```
├── Oakleaf-FX
│   ├── batchScript            ジョブスクリプト
│   │   ├── blockMesh.sh       blockMesh用ジョスクリプト
│   │   ├── decomposePar.sh    decomposePar用ジョスクリプト
│   │   └── solve.sh           解析ソルバ(pimpleFoam)用ジョスクリプト
│   └── benchmark.conf         ベンチマークケースのパラメータ設定(benchmark.sh用)
├── README                     説明書
├── bin
│   ├── benchmark.sh           ベンチマークスクリプト
│   ├── fippplot.py            基本プロファイラfipp結果表のプロットスクリプト(Python)
│   ├── fipptable.sh           基本プロファイラfippの結果表作成スクリプト
│   ├── plot.py                プロットスクリプト(Python)
│   └── table.sh               ベンチマーク結果の集計スクリプト
└── template                   channelReTau110ケースのテンプレート
```

ベンチマークケースのパラメータ編集

ベンチマークケースのパラメータ編集

```
cd Oakleaf-FX
emacs -nw benchmark.conf vi,geditなど各自慣れているエディタで編集する
```

nArray(格子数倍率nのベンチマークケース)

格子数倍率nは，格子数374400(nx=120,ny=65,nz=48)をベースにとした格子の倍数である.
ベンチマークケースnArrayの要素にはnを0埋めの5桁の整数値，かつ文字列形式で記述する.
大きい倍数を指定すると解析時間がかかるので，演習時は”00002”または”00001”，もしくは”00001” “00002”の両方とする．演習後は大きい倍数を指定可能だが，ジョブの制限時間が15分であることに注意して指定する.

比較したいケースの先頭の#を取る.

```
nArray=(\
# "00001" \
# "00002" \
# "00004" \
# "00008" \
)
```

ベンチマークケースのパラメータ編集

mpiArray(MPI並列数mpiのベンチマークケース)

MPI並列数mpiのベンチマークケースmpiArrayの要素には，mpiの値を0埋めの5桁の整数かつ文字列形式で記述する.

演習中は変更しない.

```
mpiArray=( "00192" "00128" "00064" "00032" "00016" )
```

simulationTypesArray(乱流モデルのベンチマークケース)

乱流モデルのベンチマークケースsimulationTypesArrayの要素には，以下の形式の文字列を羅列する.

”(simulationTypesの値)-LESModel_(LESModelの値)-delta_(deltaの値)"

層流モデル以外にもLESのSmagorinskyモデルもケースに含める場合には，先頭の#を除く

```
simulationTypesArray=(\
  "laminar-LESModel_laminar-delta_cubeRootVol" \
#  "LESModel-LESModel_Smagorinsky-delta_vanDriest" \
)
```

ベンチマークケースのパラメータ編集

solversArray(圧力に対する線型ソルバのベンチマークケース)

圧力に対する線型ソルバのベンチマークケースsolversArrayの要素には,
以下の形式の文字列を羅列する. “PCG-preconditioner_(preconditionerの値)" (PCGの場合). “GAMG-smoother_(smootherの値)” (GAMGの場合)
一般的にMPI並列数が少ない場合にはGAMG, 多い場合にはPCGのほうが速いので, 両方を比べてみるのが良い.

比較したいケースの先頭の#を取る.

```
solversArray=(\
#  "PCG-preconditioner_FDIC" \
#  "PCG-preconditioner_DIC" \
#  "PCG-preconditioner_diagonal" \
#  "GAMG-smoother_FDIC" \
#  "GAMG-smoother_DIC" \
#  "GAMG-smoother_DICGaussSeidel" \
#  "GAMG-smoother_GaussSeidel" \
#  "GAMG-smoother_nonBlockingGaussSeidel" \
#  "GAMG-smoother_symGaussSeidel" \
)
```

ベンチマークケースの実行

ベンチマークケースの実行(時間がかかるので，バックグラウンドで実行する)

```
../bin/benchmark.sh >& log.benchmark.sh &
```

ログのモニター

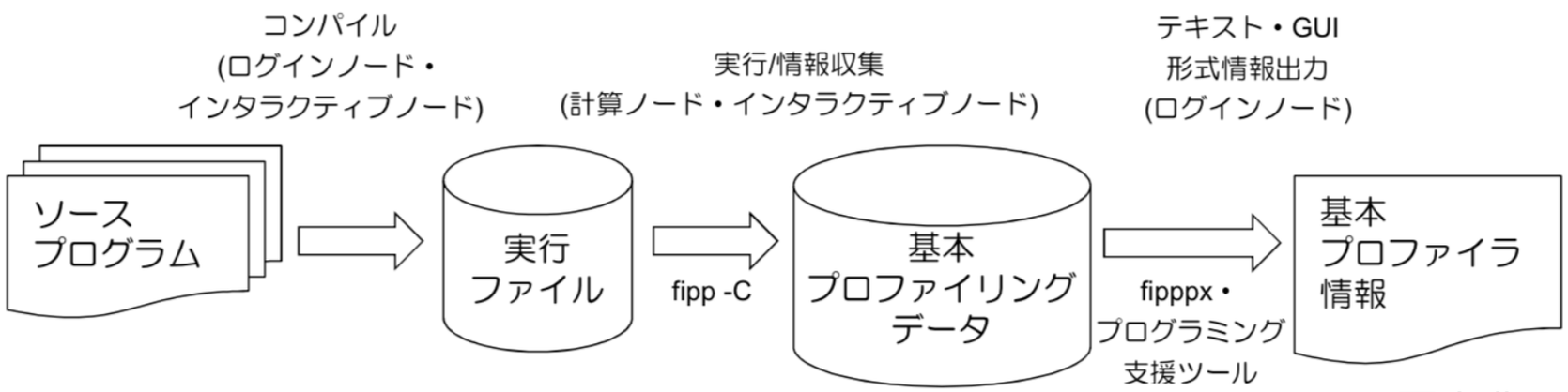
```
tail -f log.benchmark.sh ^Cで終了して良い(ベンチマーク自体は止まらない)
```

ジョブ状態確認

```
pjstat
```

基本プロファイラfipp

- サンプリングによる情報収集に基づいた，以下のようなハードウェアモニタ情報収集やコスト分析が可能
  - ✓ FLOPSおよび，FLOPS/PEAK[%] (浮動小数点演算の実行性能)
  - ✓ MIPSおよび，MIPS/PEAK[%] (整数演算の実行性能)
  - ✓ メモリスループット，メモリスループット/PEAK[%]
  - ✓ SIMD(single instruction multiple data)率 (ベクトル演算率)
  - ✓ コールグラフおよびコスト分析
- 全区間測定であればソース変更・ビルドは不要
- より詳細な情報を収集できるプロファイラfappもあるが，ソースに測定範囲を指定して再ビルドする必要がある.



基本プロファイラfippの概要

図出典：Oakleaf-FX/
Oakbridge-FX 利用手引書

基本プロファイラを併用したソルバ実行

~/lecture/channelReTau110/OakleafFX10/batchScript/solve.sh

```
!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture"
#PJM -g "gt00"
#PJM –j
#PJM -S
#------- Program execution -------#
module load OpenFOAM/2.3.0
source $WM_PROJECT_DIR/etc/bashrc


getNumberOfProcessors，getApplication等の関数が定義されているbashスクリプト
. $WM_PROJECT_DIR/bin/tools/RunFunctions

env 環境変数も念のため記録しておく

log=log.${PJM_JOBID}

基本プロファイラfippを用いて，ハードウェアモニタ情報およびコールグラフ情報を収集
fipp -d $log.fippdir -C -Ihwm,call -H  -L shared \
mpiexec --stdout $log -np $(getNumberOfProcessors) $(getApplication) -parallel
```

基本プロファイラのデータ参照

~/lecture/channelReTau110/bin/fipptable.sh

fipppx -A -Ihwm,call -d プロファイラディレクトリ > プロファイラの結果

プロファイラの結果(格子数約0.74M，16MPI並列，laminar，PCG-FDICの例)

```
Performance monitor event:Statistics

   ****************************************************************************
       Application - performance monitors
   ****************************************************************************

     Elapsed(s)       MFLOPS MFLOPS/PEAK(%)          MIPS   MIPS/PEAK(%)
   ----------------------------------------------------------------------------
      27.5677    3488.7907          1.4749    22058.7821         18.6509   Application
```

実行性能

(略)

```
                  Mem throughput   Mem throughput
     Elapsed(s)   _chip(MB/S)      /PEAK(%)         SIMD(%)
   ----------------------------------------------------------------
      27.5677    613003.1898        44.9089          1.6306   Application
   ----------------------------------------------------------------
```

負荷が高い関数には#のマークが付く

(略)

```
                  #####|  26%          <4>
Foam::FDICPreconditioner::precondition(Foam::Field<double> &, const Foam::Field<double> &,
unsigned char) const [57]
```

ベンチマーク結果の集計・プロット

ベンチマーク結果の集計

```
../bin/table.sh 集計表table.csvが作成される
```

ベンチマーク結果のプロット(Pythonスクリプト)

```
../bin/plot.py プロット図plot.pdfが作成される
```

基本プロファイラの結果の集計

```
../bin/fipptable.sh 集計表fipptable.csvが作成される
```

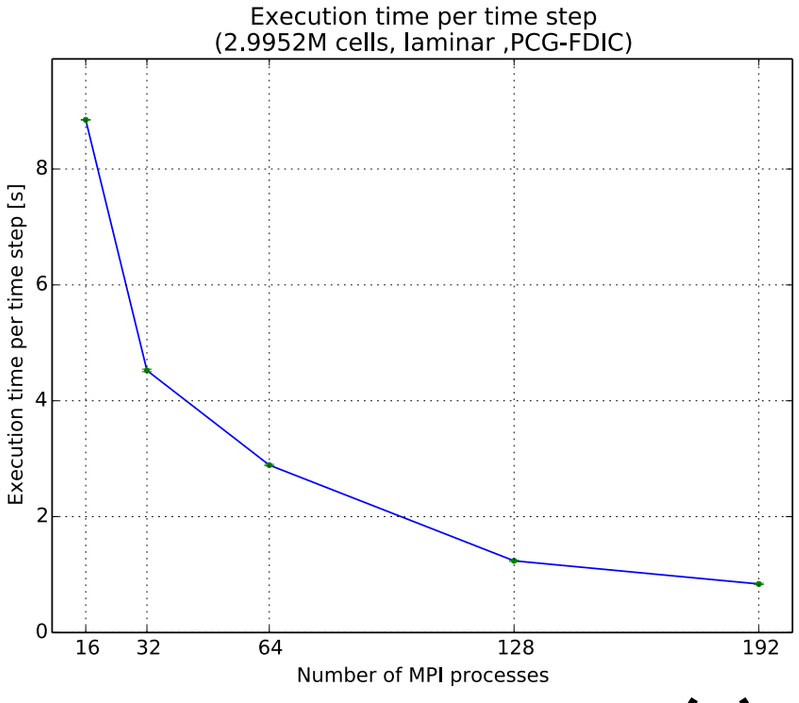
基本プロファイラの結果のプロット(Pythonスクリプト)

```
../bin/fippplot.py プロット図fipp.pdfが作成される
```

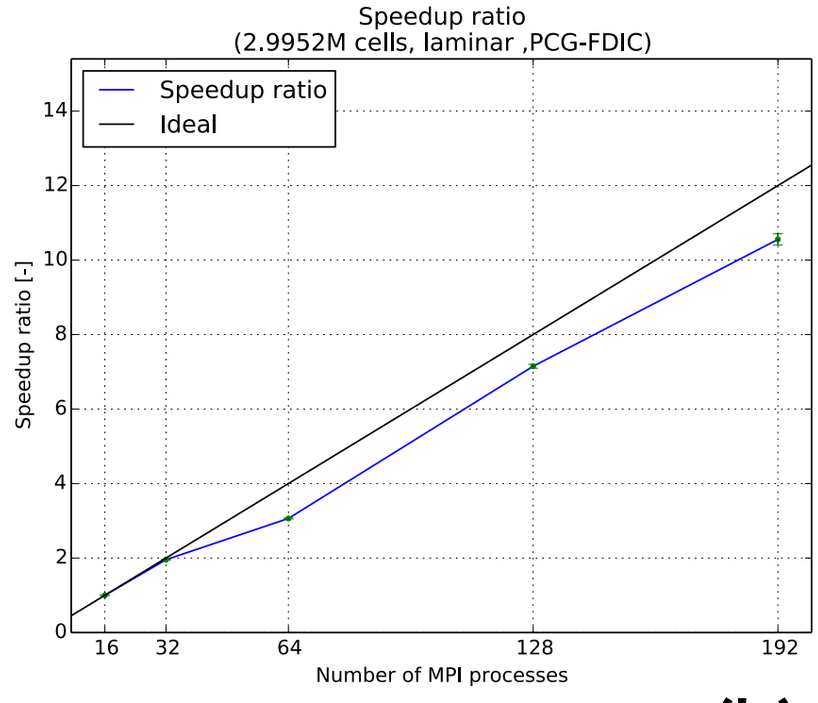
ベンチマーク結果の転送と結果閲覧

```
ユーザー名@ユーザーマシン ~/lecture/damBreakFine
$ cd ../channelReTau110/Oakleaf-FX

ユーザー名@ユーザーマシン ~/lecture/channelReTau110/Oakleaf-FX
$ rsync -auv tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/lecture/channelReTau110/
Oakleaf-FX/ ./ --exclude=n_* n_で始まるケースディレクトリは大きいので転送しない
画像ビューワでプロット図(*.pdf)を閲覧．集計表(*.csv)を閲覧
```

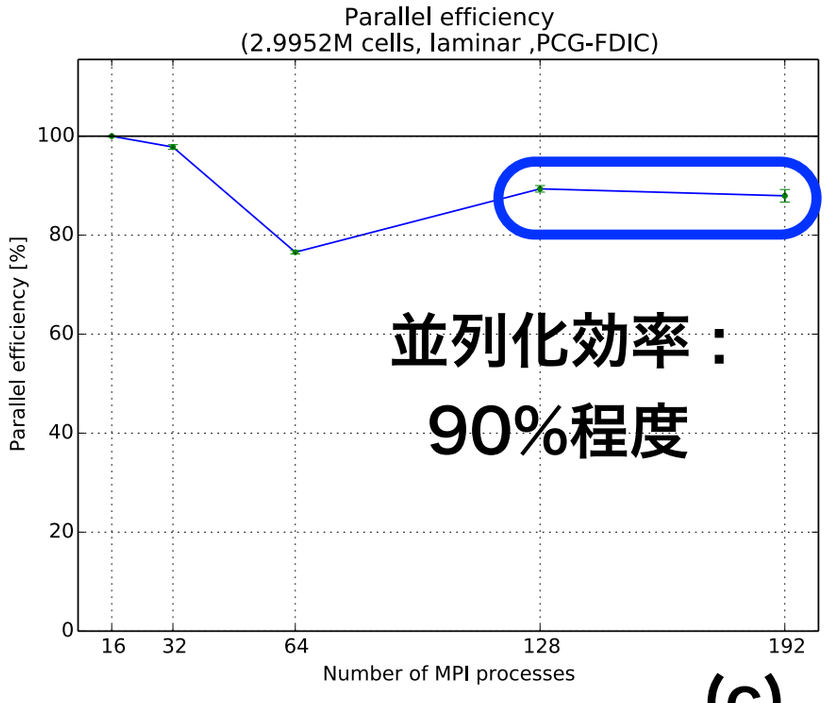
ベンチマーク結果プロット図の例



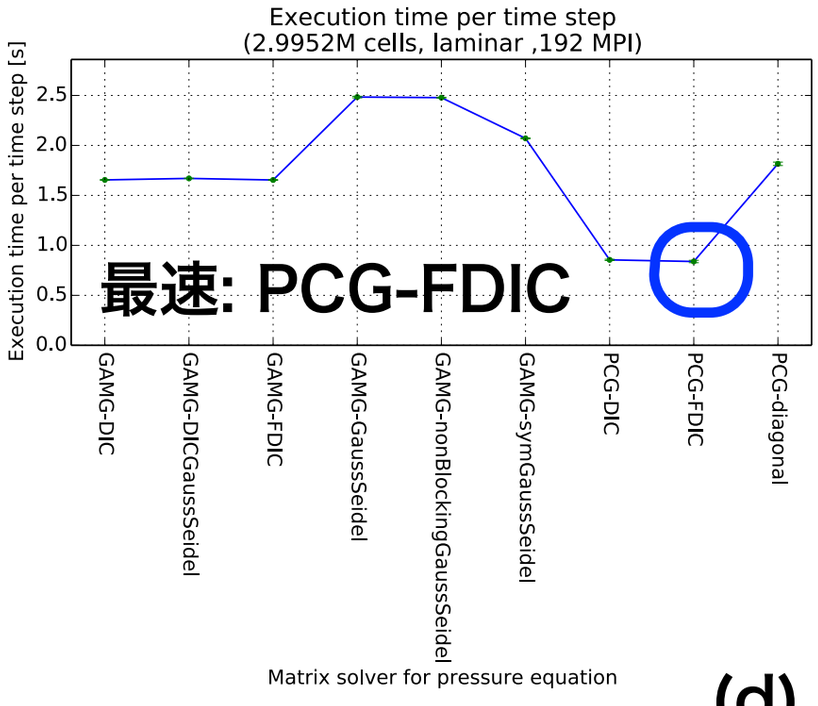
(a)



(b)



(c)



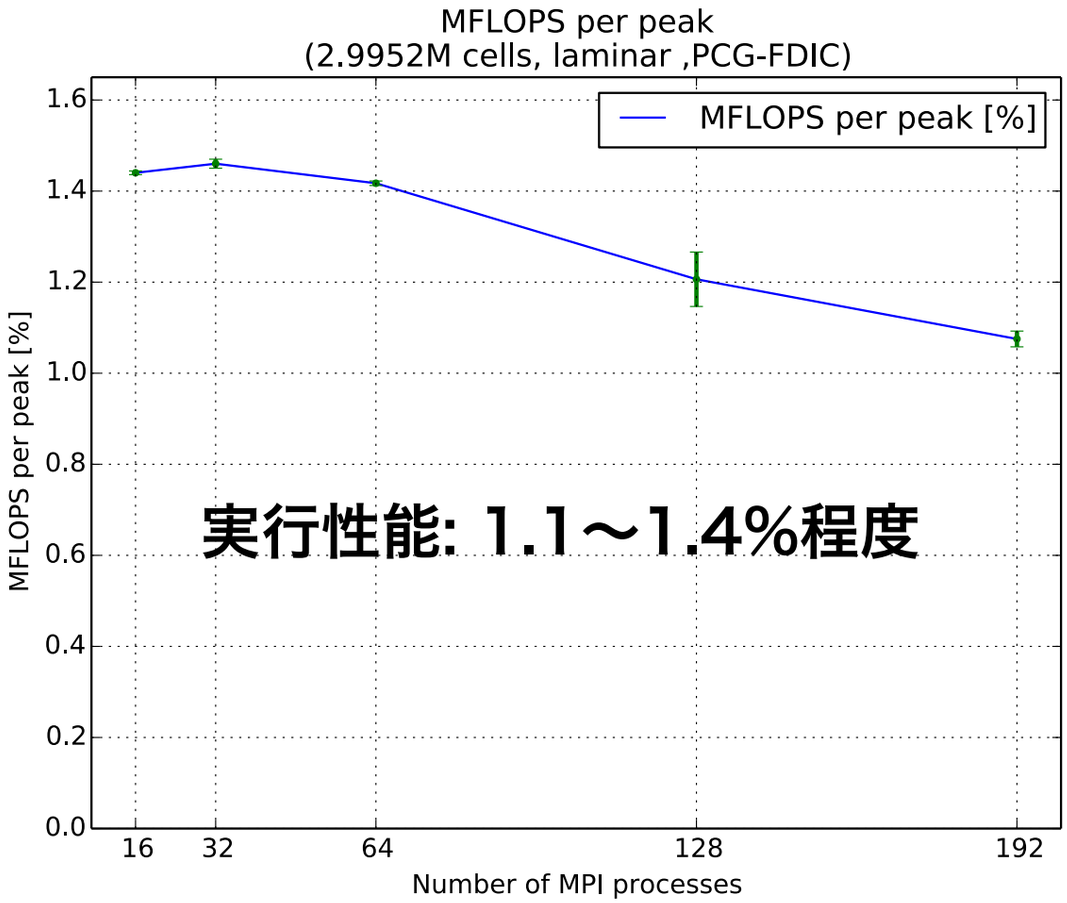
(d)

- 格子数: 約3M(格子数倍率: n=8)
- 乱流モデル: 無し (laminar)
- 圧力線型ソルバ: a),b),c)PCG-FDIC

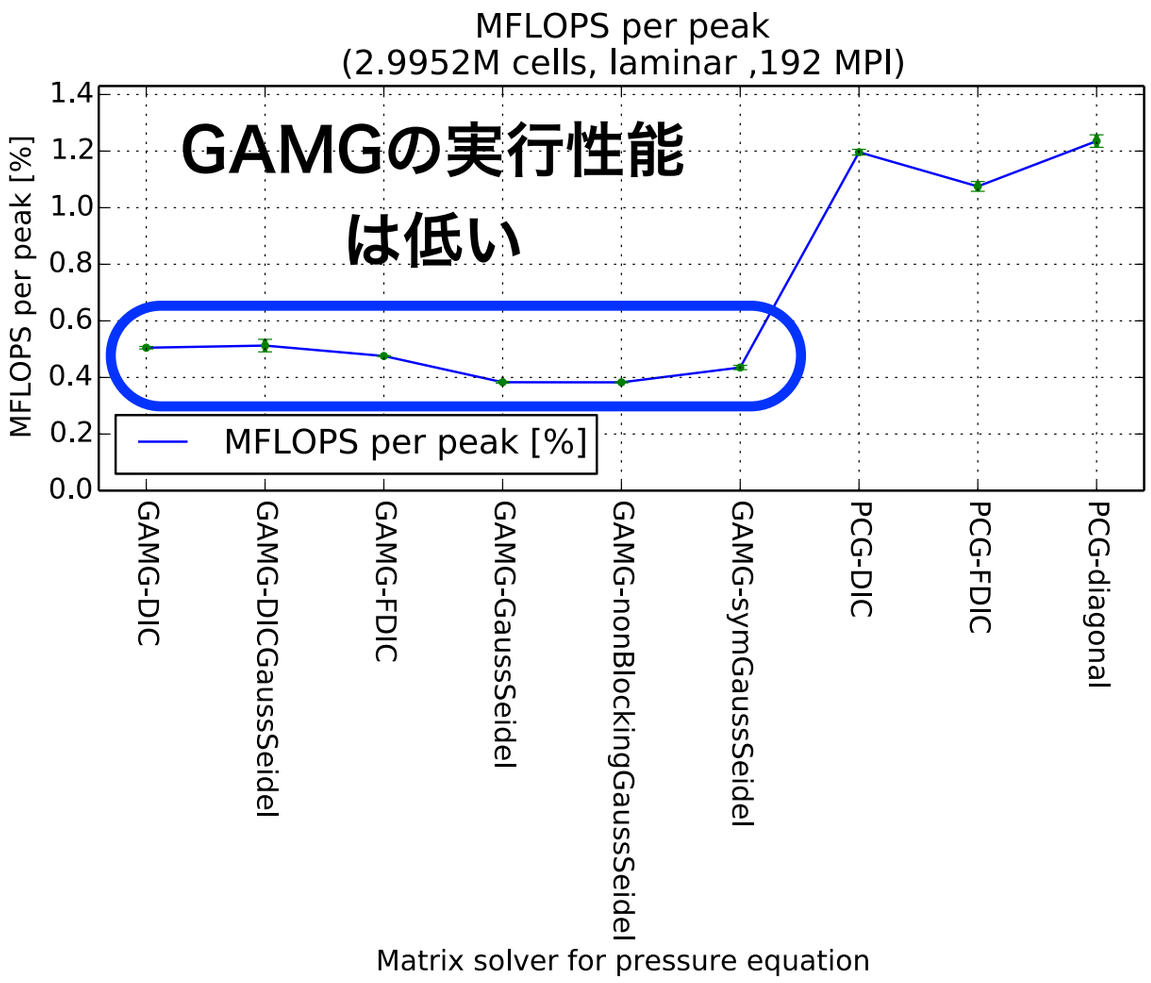
a) 時間ステップ当りの計算時間
$= (T_{11step}-T_{1step})/10$
b) スピードアップ
c) 並列化効率
d) 192並列での圧力線型ソルバの比較

実行性能プロット図の例

- 格子数: 約3M(格子数倍率: n=8)
- 乱流モデル: 無し(laminar)



MPI並列数と実行性能の関係
(圧力線型ソルバPCG-FDIC)



圧力線型ソルバと実行性能の関係
(192MPI並列)

チュートリアルの実行(自習課題)

チュートリアルの実行(自習課題)

DNS	直接数値シミュレーション
basic	基礎的なCFDコード
combustion	燃焼
compressible	圧縮性流れ
discreteMethods	離散要素法
electromagnetics	電磁流体
financial	金融工学
heatTransfer	熱輸送
incompressible	非圧縮性流れ
mesh	格子生成
multiphase	多層流
lagrangian	ラグランジアン粒子追跡
resources	形状データ等の共用リソース置き場
stressAnalysis	固体応力解析

カテゴリ別に約170のチュートリアルがある．以下のサイトやカテゴリ，ケース名等を参考に，実行したいケースを選ぶ．

オープンCAE勉強会@関西
OpenFOAMチュートリアルドキュメント作成プロジェクト
https://sites.google.com/site/freshtamanegi/

チュートリアルとジョブスクリプトのコピー

チュートリアルケースのコピー(最初に一回のみ行う)

```
cd
cp -r /usr/local/OpenFOAM/2.3.0/OpenFOAM-2.3.0/tutorials/ .
cd tutorials
ls
```

```
Allclean Allrun Alltest DNS basic combustion compressible
discreteMethods electromagnetics financial heatTransfer incompressible
lagrangian mesh multiphase resources stressAnalysis
```

カテゴリのディレクトリが表示される

チュートリアルケース(cavity)に移動し，逐次計算か並列計算か調べる

```
cd incompressible/icoFoam/cavity
ls
```

```
0  constant  system
```

Allrunが無い場合やAllrunにrunParallelの文が無い場合は逐次計算ケース

逐次計算用ジョブスクリプトのコピー

```
cp ~/lecture/foamRunTutorials.PJM ./
```

チュートリアルの実行(逐次計算の場合)

foamRunTutorials.PJM

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture"
#PJM -g gt00
#PJM -L "node=1"
#PJM --mpi "proc=1" 1MPI(逐次計算)
#PJM -j
#------- Program execution -------#
module load OpenFOAM/2.3.0
source $WM_PROJECT_DIR/etc/bashrc

foamRunTutorials チュートリアルを実行するスクリプト
(Allrunがあればそれを実行し，無ければblockMeshとソルバを実行する)
```

ジョブの投入・ジョブ確認・ログ確認

```
pjsub foamRunTutorials.PJM
pjstat
tail -f log.icoFoam ジョブの実行開始後に行う
```

チュートリアルの実行(並列計算の場合)

チュートリアルケース(motorBike)に移動し，逐次計算か並列計算か調べる

```
cd ~/tutorials
cd incompressible/simpleFoam/motorBike
ls
```

```
0.org  Allclean  Allrun  constant  system
```

Allrunがある場合，Allrunの中身を調べる

```
more Allrun
```

```
略

runParallel snappyHexMesh 6 -overwrite
略

runParallel patchSummary 6
runParallel potentialFoam 6 -noFunctionObjects -writep
runParallel $(getApplication) 6
runParallel $application 6
略
```

runParallel(並列実行用bash関数)の第二引数が並列数であるが，全て6並列

チュートリアルの実行(並列計算の場合)

ジョブスクリプトのコピーと修正

```
cp ~/lecture/foamRunTutorials.PJM ./
emacs -nw foamRunTutorials.PJM
```

```
#!/bin/bash
#------ pjsub option --------#
#PJM -L "rscgrp=lecture"
#PJM -g gt00
#PJM -L "node=1"
#PJM --mpi “proc=1 6" ←6MPIに変更する
#PJM -j
#------- Program execution -------#
module load OpenFOAM/2.3.0
source $WM_PROJECT_DIR/etc/bashrc

foamRunTutorials
```

チュートリアルの実行(並列計算の場合)

最終時刻の変更

```
emacs -nw system/controlDict
```

endTime ~~5~~100; lecureキューでは時間制限が15分間と短く，このままだと計算が終了しないので，最終時刻を短くする.

ジョブの投入・ジョブ確認・ログ確認

```
pjsub foamRunTutorials.PJM
pjstat ジョブの実行開始(RUNNING)を確認したら以下のログ確認に進む
tail -f foamRunTutorials.PJM.o* 並列計算ではソルバのログは
foamRunTutorials.PJM.oJOBID に出力される(foamRunTutorialsではmpiexecのstdoutオプションを用いて，標準出力をファイルに出力しないため)
```

チュートリアルの実行(並列計算の場合)

チュートリアルの転送とParaViewによる可視化

ユーザー名@ユーザーマシン ~/任意

```
$ cd
$ mkdir tutorials
$ cd tutorials
$ rsync -auv tYYxxx@oakleaf-fx.cc.u-tokyo.ac.jp:~/tutorials/ ./
 --exclude=processor*   再構築済の場合，領域分割データ(processor*)は転送不要
$ cd incompressible/simpleFoam/motorBike
$ touch pv.foam ParaViewで可視化する.
```

1. Fileメニュー/Open/motorBikeのディレクトリのpv.foamを開く
2. Fileメニュー/Open/postProcessing/cuttingPlane/100/U_yNormal.vtk
3. Coloring/・U

チュートリアルの初期化

チュートリアルのログが残っていると，foamRunTutorialsによるチュートアリアルの再実行ができないので，再実行する場合には以下のように初期化する.

OpenFOAMの環境設定
(ログイン後に一度だけ実行する．または，~/.bashrc に以下を記述する)

```
module load OpenFOAM/2.3.0 #OpenFOAM/2.3.0のmoduleの設定
source $WM_PROJECT_DIR/etc/bashrc #OpenFOAMの環境設定
```

チュートリアルの初期化

```
foamCleanTutorials
```

さらに学ぶには

1. 東京大学情報基盤センター，スーパーコンピューティング部門，FX10 スーパーコンピュータシステム (http://www.cc.u-tokyo.ac.jp/system/fx10/)
   a) 利用の手引，利用支援ポータル内の「Oakleaf-FX/Oakbridge-FX 利用手引書」
2. オープンCAE学会(http://www.opencae.or.jp/)
   a) OpenFOAMユーザガイド和訳，プログラマズガイド和訳
   b) The ParaView Tutorial和訳
   c) 過去のシンポジウム・ワークショップ・講習会資料
   d) オープンCAE勉強会：毎月全国数ヶ所で開催されている．過去資料や動画も有り．
3. 一般財団法人 高度情報科学技術研究機構主催のOpenFOAMワークショップ，　第1回 (http://www.hpci-office.jp/pages/ws_openfoam_130927)，第2回(http://www.hpci-office.jp/pages/ws_openfoam2_141017)：FX10の兄である「京」でのOpeFOAM解析例やチューニング例.
4. The OpenFOAM Foundation (http://www.openfoam.org/)

本日の講習は以上です

大変お疲れさまでした

質問をどうぞ!