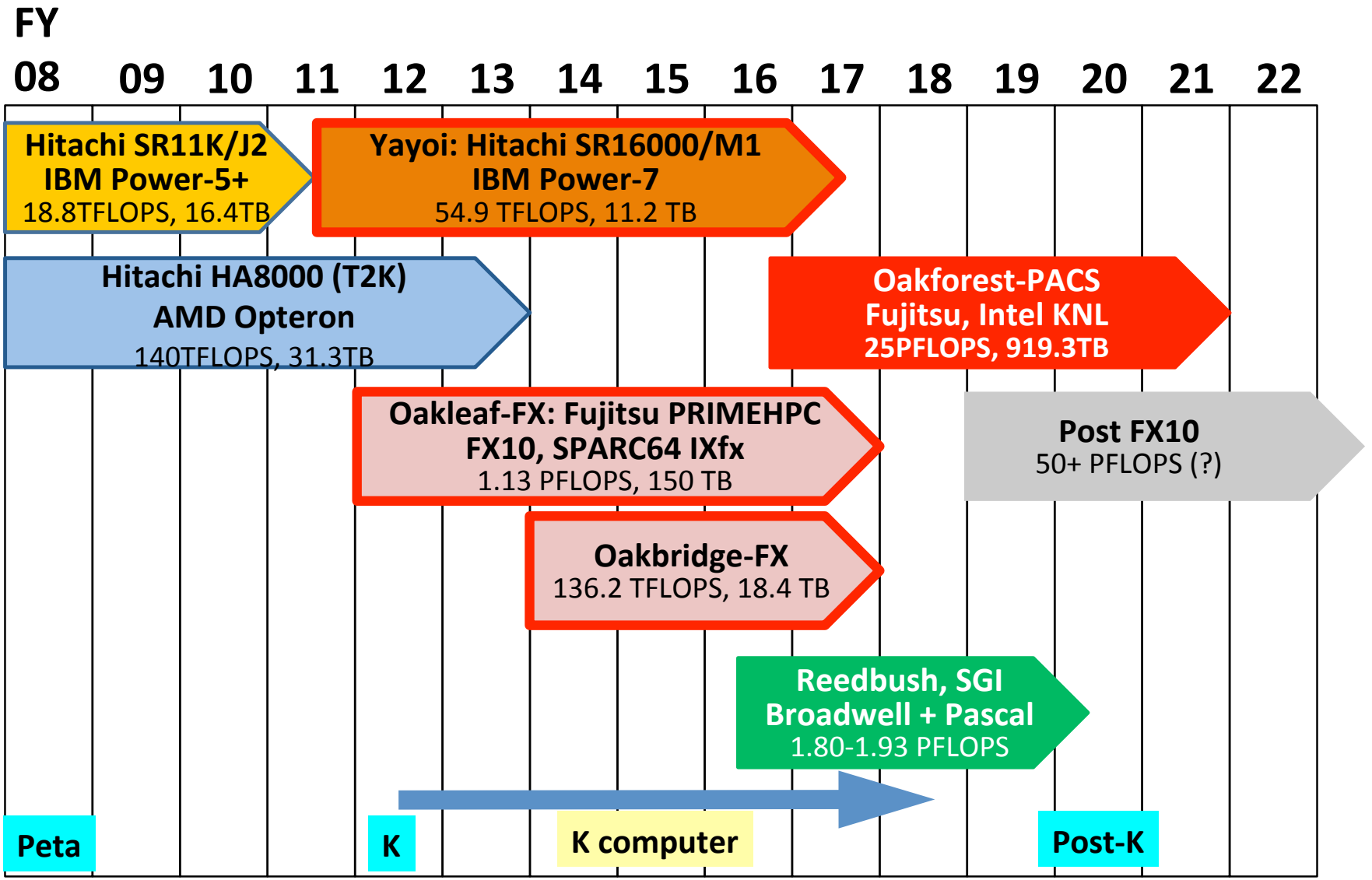
東京大学情報基盤センター
お試しアカウント付き並列プログラミング講習会

「Oakforest-PACS概要・OpenFOAM概要」

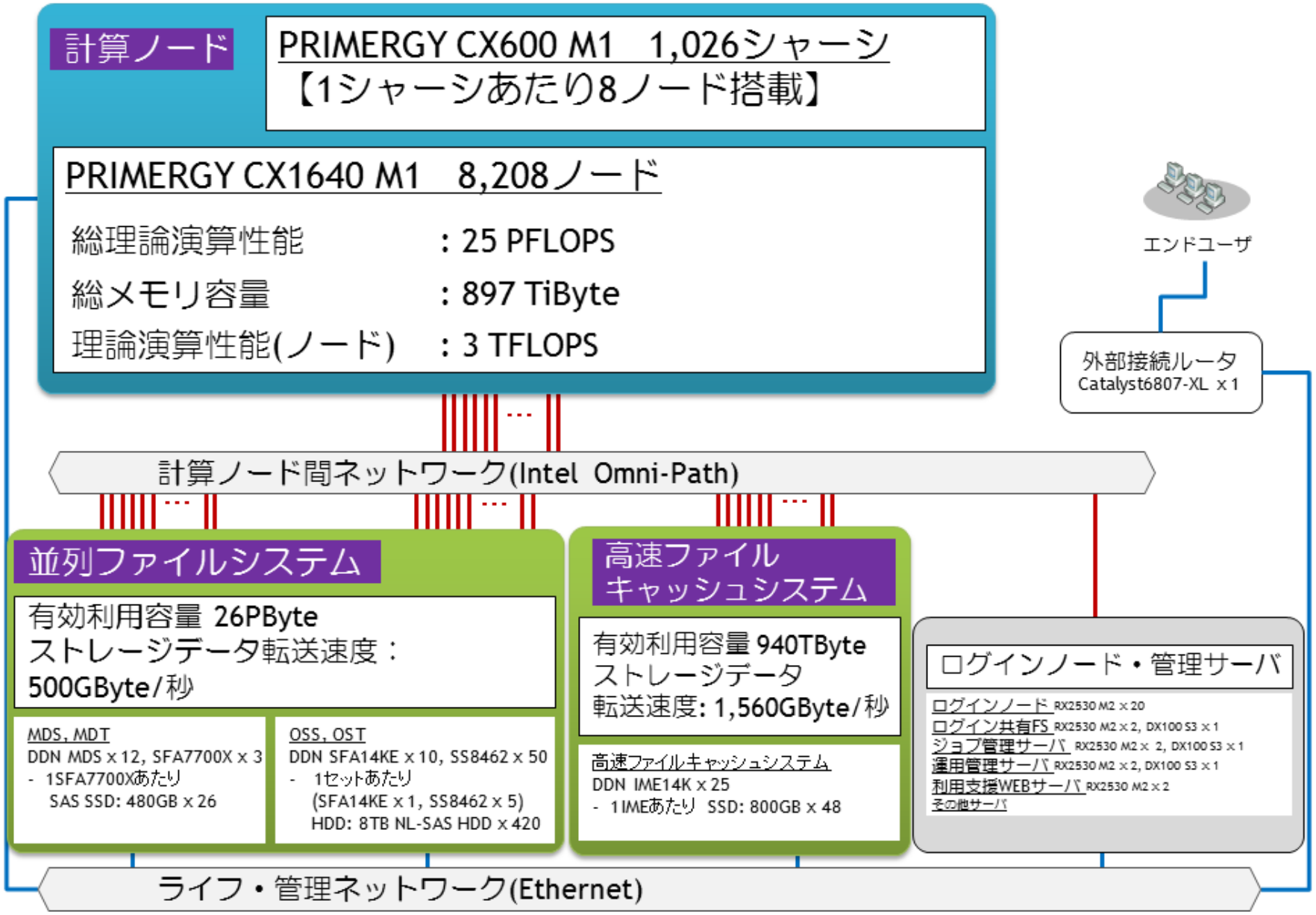
今野　雅
(東京大学客員研究員・株式会社OCAEL)

東京大学情報基盤センターの運用システム



これまでは概ね2基の大型システム，約6年サイクルで更新

Oakforest-PACS(OFP)のシステム構成



OFPのハードウェア全体構成

項目		仕様
総理論演算性能		25.004 PFlops
総ノード数		8208
総主記憶容量		897 TByte
ネットワークトポロジー		Full-bisection Fat Tree
並列ファイルシステム	システム名	Lustreファイルシステム
	サーバ(OSS)	DDN SFA14KE
	サーバ(OSS)数	10
	ストレージ容量	26 PB
	ストレージデータ転送速度	500 GB/sec
高速ファイルキャッシュシステム	サーバ	DDN IME14K
	サーバ数	25
	容量	940 TB
	ストレージデータ転送速度	1,560 GB/sec

OFPのノード構成

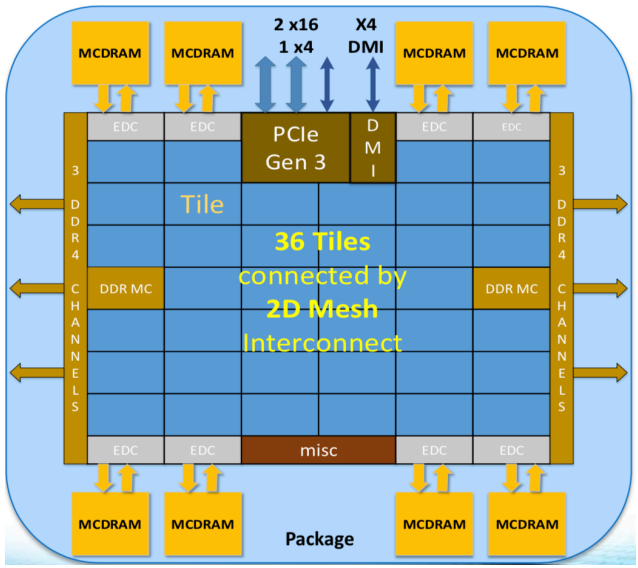
項目		仕様
マシン名		Fujitsu PRIMERGY CX1640 M1
CPU	プロセッサ名	Intel® Xeon Phi ™ 7250（開発コード名：Knights Landing)
	プロセッサ数(コア数)	1 (68)
	周波数	1.4 GHz
	理論演算性能	3.0464 TFlops
Memory		96 GB(DDR4)＋ 16 GB(MCDRAM)
インターコネクト		Intel ® Omni-Path ネットワーク(100 Gbps)

Xeon Phi 7250(Knights Landing, KNL)プロセッサ

- プロセッサ：34タイル
- タイル：コア×2 + 共有L2キャッシュ
- コア：Atom(1.4GHz)
- メモリ
  - ✓ MCDRAM：16GB，490～GB/s(注，高バンド幅)
  - ✓ DDR4：96GB，115.2GB/s(大容量)

図出典：Hotchips 27 (注) Stream Triadベンチマーク

タイル



特徴：単体性能が低いメニーコア，多くのVPU(ベクトル演算ユニット)，高バンド幅メモリ

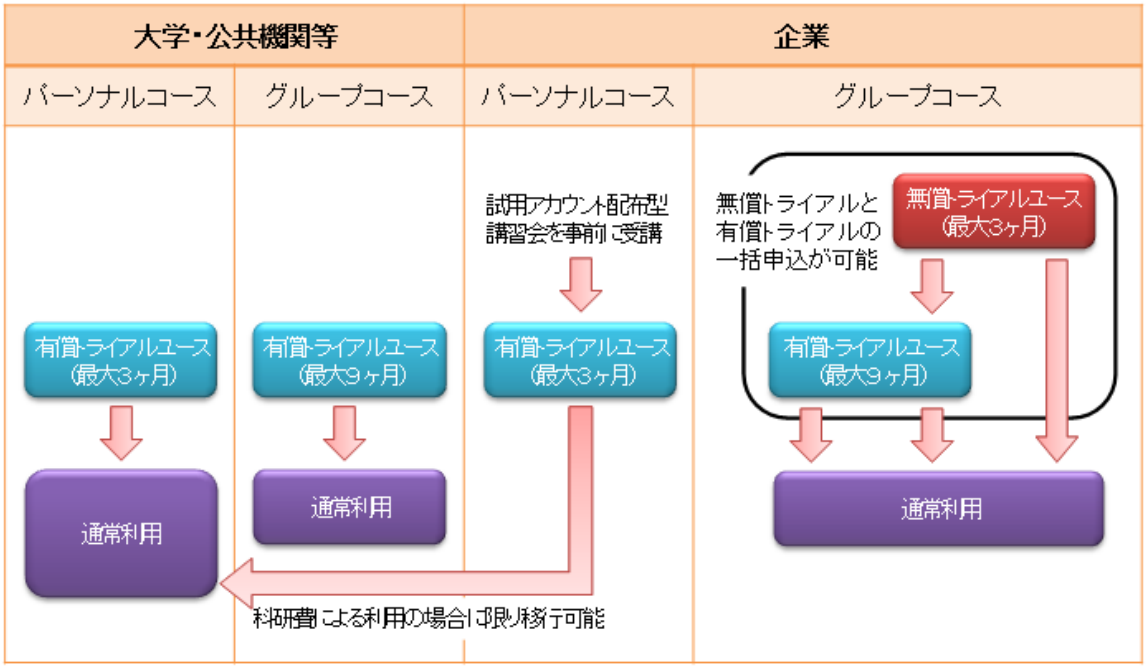
OFPの利用料金

- パーソナルコース：年間10万円(1口あたり)．最大3口まで
  - ✓ トークン：17,280ノード時間(2ノード×24時間×360日分)
  - ✓ 消費係数：1(8ノード以下)，2(8ノード超)
  - ✓ ディスク：並列ファイルシステム1TB
  - ✓ 企業利用無し
- グループコース：申込8ノードあたり年間40万円(企業48万円)
  - ✓ トークン：Nノード×24時間×360日分　(N：申込ノード数)
  - ✓ 消費係数：1(Nノード以下)，2(N超)
  - ✓ ディスク(グループ当り)：並列ファイルシステム8TB
- 1年未満で月数別利用負担制度もあり
- 企業利用(グループのみ)は利用審査があり，簡単な成果公開が必要
- Reedbushとのトークン相互交換可能(一般利用のみ．企業，HPCI等では不可)
- 1週間お試しアカウント付き講習会(年10回程度)，トライアルユース(無償・有償)有り

トライアルユース

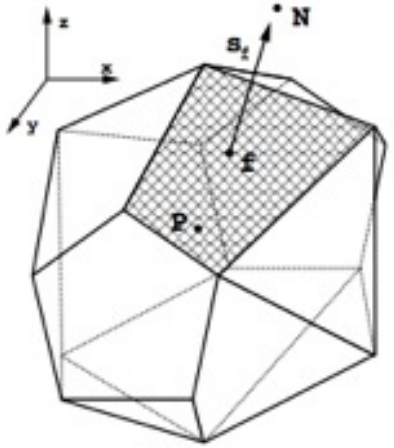
有償トライアルコース(安価)

無償トライアルコース(企業のみ)

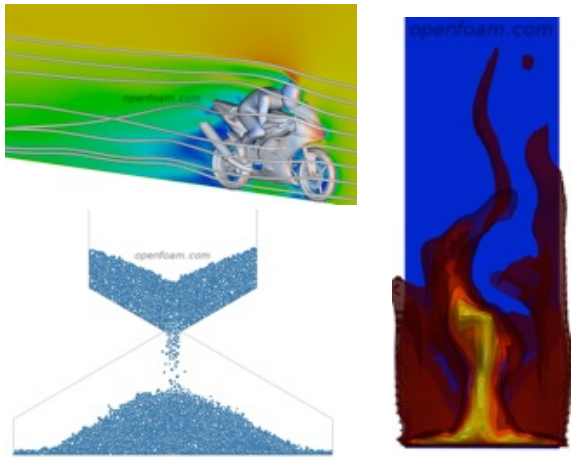


- アカデミック利用：最大３ヶ月
  - ✓ パーソナルコース
  - ✓ グループコース
- 企業利用
  - ✓ パーソナルコース：最大３ヶ月
    - 本講習会の受講が必須、審査無
  - ✓ グループコース
    - 無償トライアルユース：最大3ヶ月
    - 有償トライアルユース：最大9ヶ月
    - スーパーコンピュータ利用資格者審査委員会の審査が必要（年２回実施）
  - ✓ 双方のコースともに、簡易な利用報告書の提出が必要

OpenFOAM概要



有限体積法
ポリヘドラル

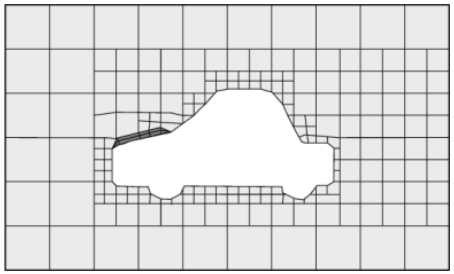


マルチフィジックス

$$\frac{\partial T}{\partial t} + \nabla \cdot (\boldsymbol{U}T) + \nabla \cdot (\alpha \nabla T) = S_T$$

```
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

図出典： [OFF] The OpenFOAM Foundation (http://www.openfoam.org/)

OpenFOAMの歴史

- 1989年ー2000年：**研究室のFORTRANコード時代**、開発元：英Imperial CollegeのGosman研(Star-CDの開発元)の Henry Weller, Charlie Hill
- 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(元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
- 2014年：**Henryらが独立(CFD Direct社)．ソースはFoundationが管理**

標準ソルバー・チュートリアルのカテゴリ

カテゴリ名	カテゴリ内容	カテゴリ名	カテゴリ名
DNS	直接数値解析	finiteArea	有限面積法
basic	基礎的なCFDコード	heatTransfer	熱輸送
combustion	燃焼	stressAnalysis	固体応力解析
compressible	圧縮性流れ	incompressible	非圧縮性流れ
discreteMethods	離散要素法	lagrangian	ラグランジアン粒子追跡
electromagnetics	電磁流体	multiphase	多層流
financial	金融工学		

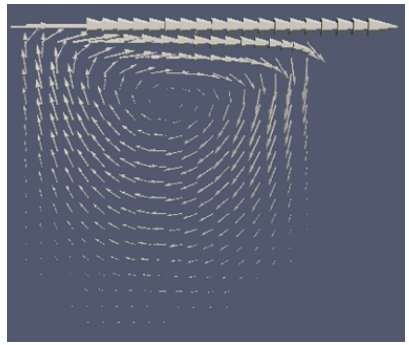
リリース年	2016				2017			2018		
リリース月	Jun	Oct	Jul	Dec	Jun	Jul	Dec	Jun	Jul	Dec
バージョン	4.0	4.1	v1606	v1612	v1706	5.0	v1712	v1806	6	v1812
カテゴリ	12	12	12	12	12	12	13	13	12	13
ソルバ	82	82	86	86	95	86	101	98	80	99
チュートリアル	207	207	226	253	284	229	298	322	240	337

注) 青色バージョン：OpenFOAM Foundation系，赤色バージョン：Plus(ESI)系

Foundation系はv1606+以降に採用されたMPIのメモリ使用量最適化がなされておらず，概ね2,000並列以上で使用量が莫大に増加するので，大規模並列ではPlus系を使用する必要がある

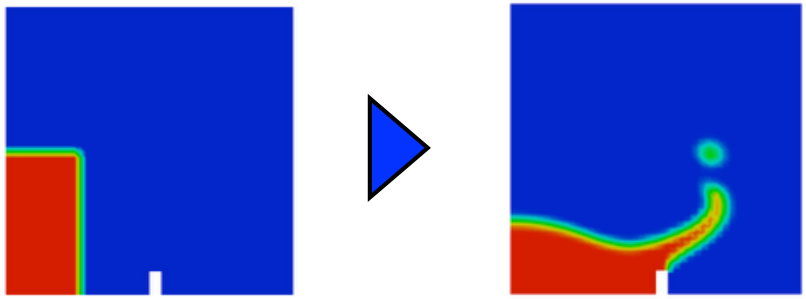
チュートリアルとは

- チュートリアルとは
  - ✓ OpenFOAM標準ソルバーの実行例
  - ✓ `foamRunTutorials` コマンドにより自動的に解析が実行できる
- ユーザガイド第2章で扱っているチュートリアル
  - ✓ cavity：天井駆動のキャビティ流れ

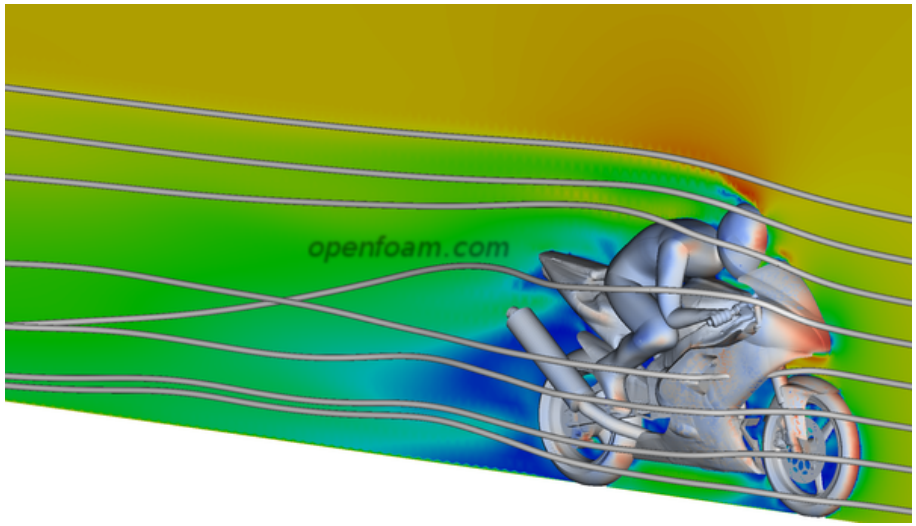


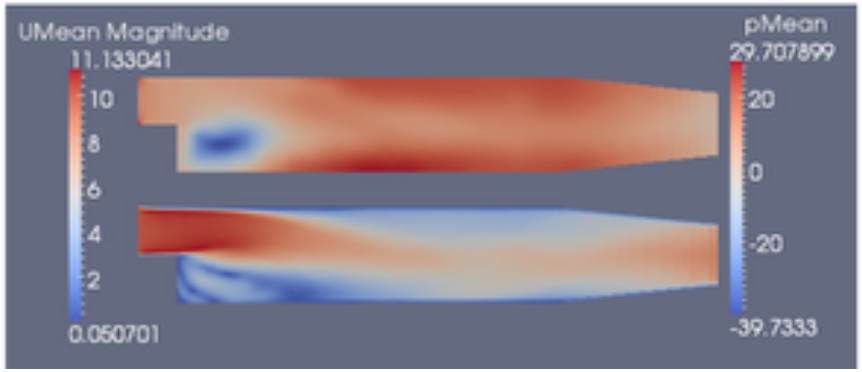
  - ✓ plateHole：穴あき板の応力解析

  - ✓ damBreak：ダムの決壊



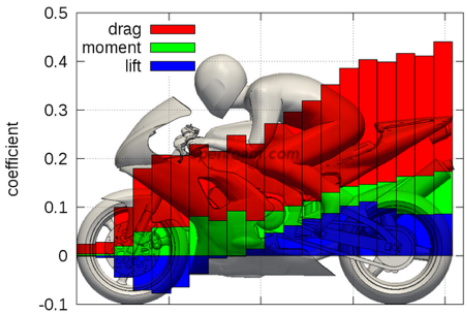
単相等温流れのチュートリアル例



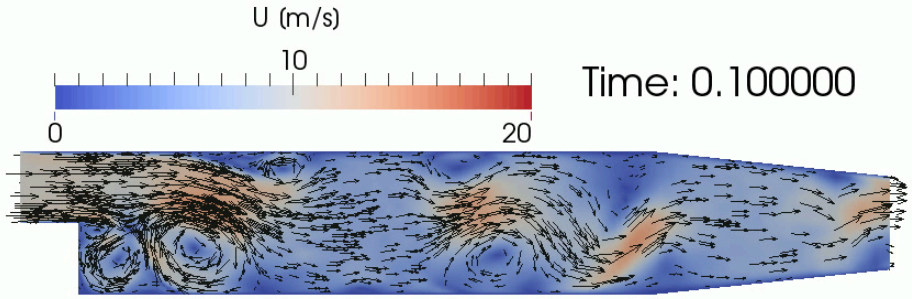


時間平均
圧力(p)

時間平均
速度(U)



バイク周りの流れ
moterBike

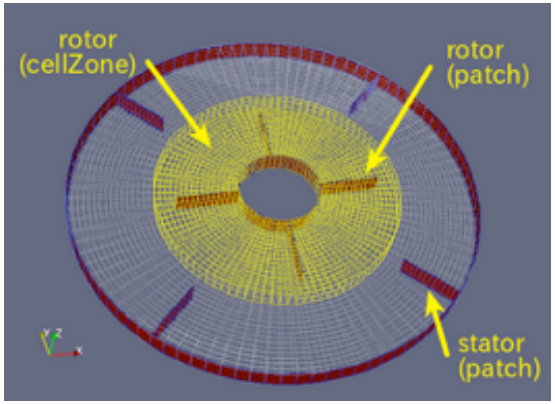


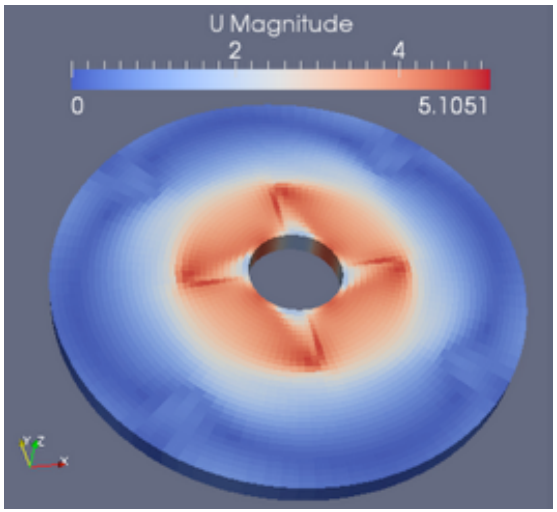
バックステップ流れ(LES)
pitzDaily

図出典： [OFF]

[OFT] 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

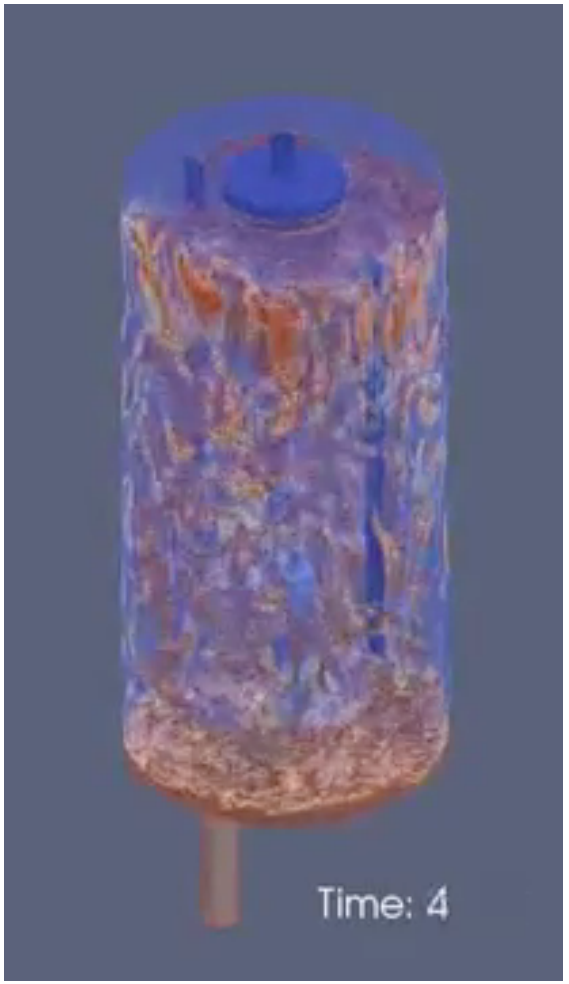
回転撹拌槽のチュートリアル例

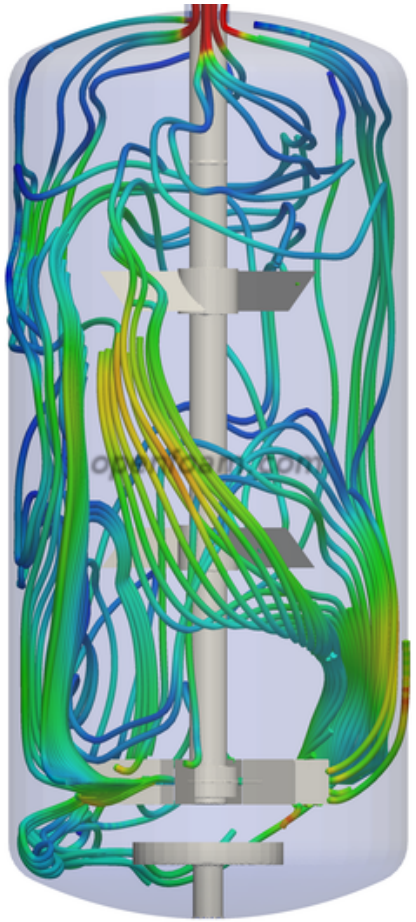




回転攪拌槽の流れ(MRF)
mixerVessel2D

図出典
[OFT]

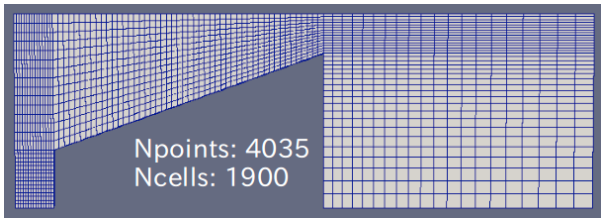


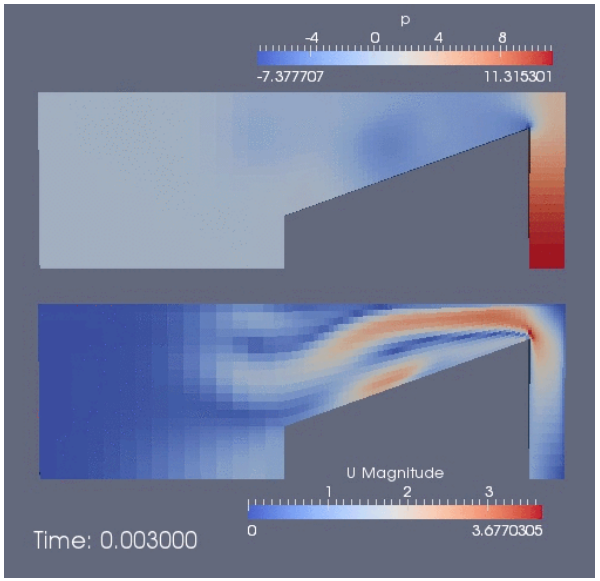


回転攪拌撹拌槽内の流れ(移動格子・AMI)
mixerVesselAMI

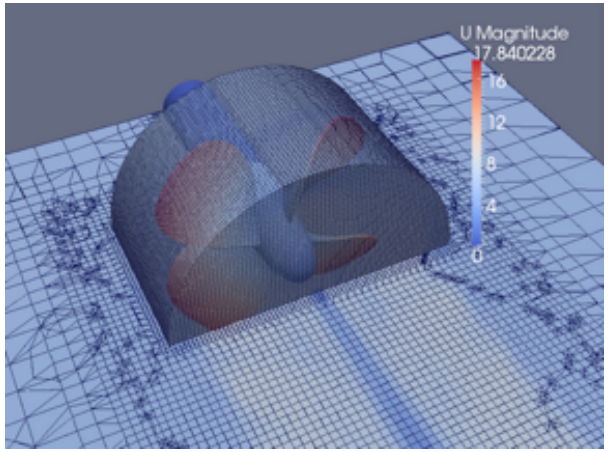
MRF: Multiple Reference Frame(回転領域内で遠心・コリオリ力を付加し，回転座標系で解く)
AMI: Arbitrary Mesh Interface(移動格子と静止格子上でのパッチ間で諸量の補間処理等を行う)

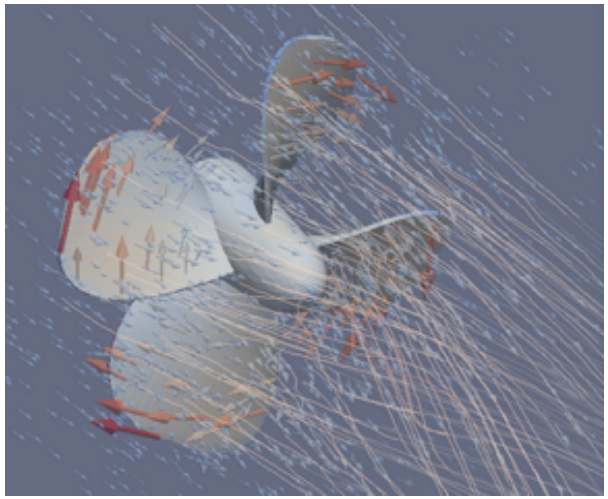
移動格子のチュートリアル例



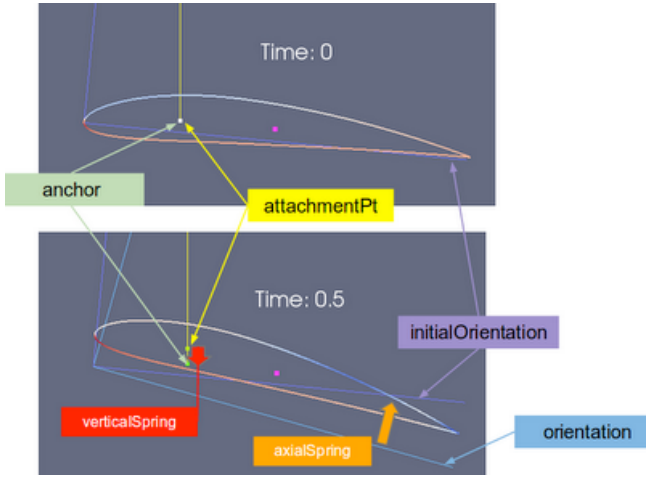


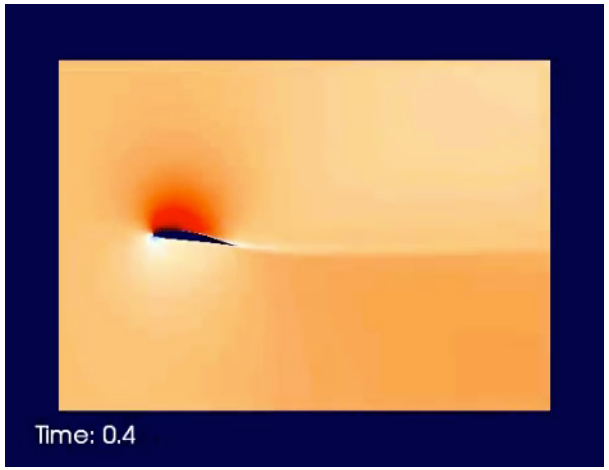
ピストン押し込み流れ
movingCone





スクリューの回転流れ場
propeller

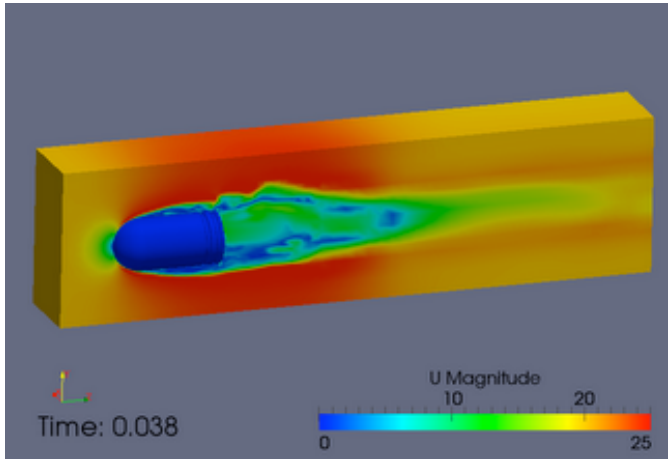




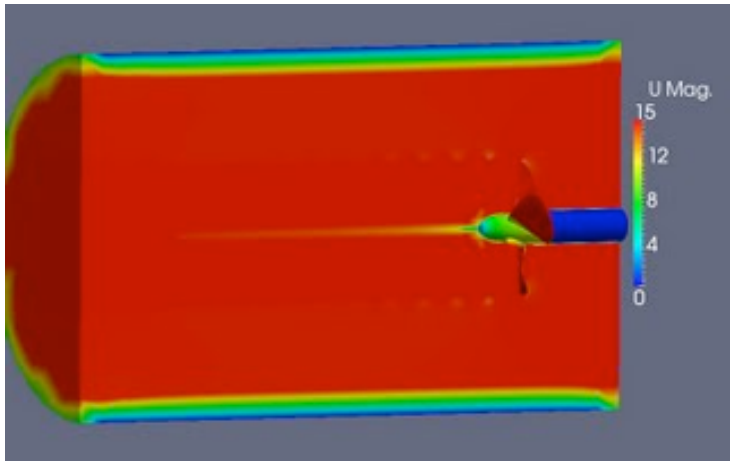
翼型の6自由度剛体運動
wingMotion

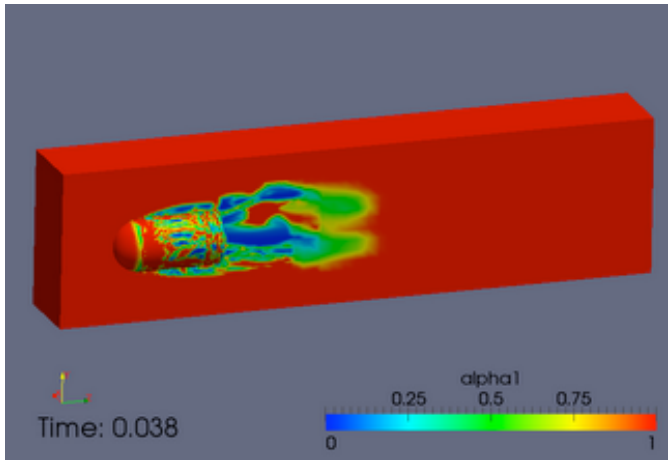
図出典 [OFT]

相変化のチュートリアル例

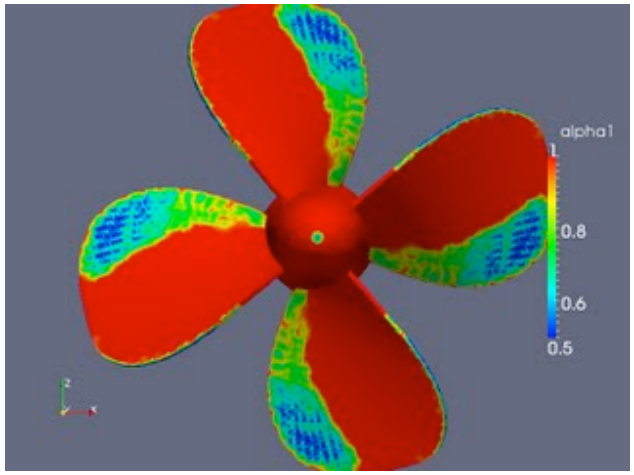


速度





相率



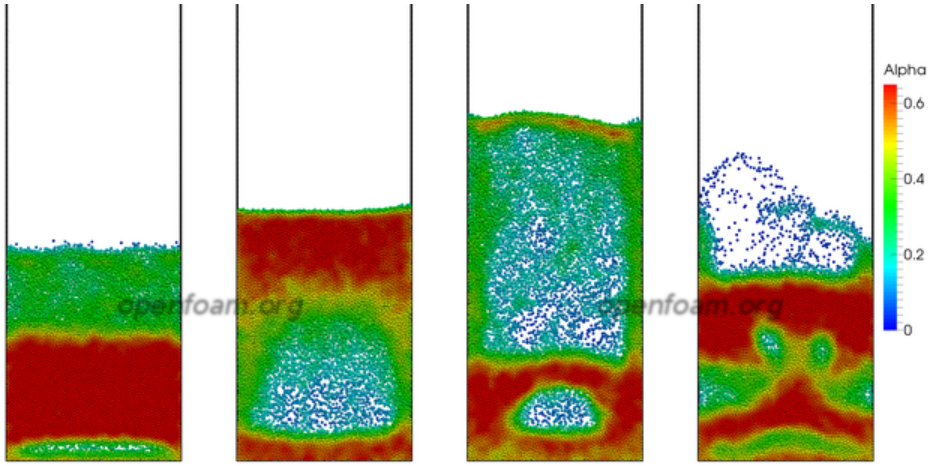
弾丸周りのキャビテーション
cavitatingBullet

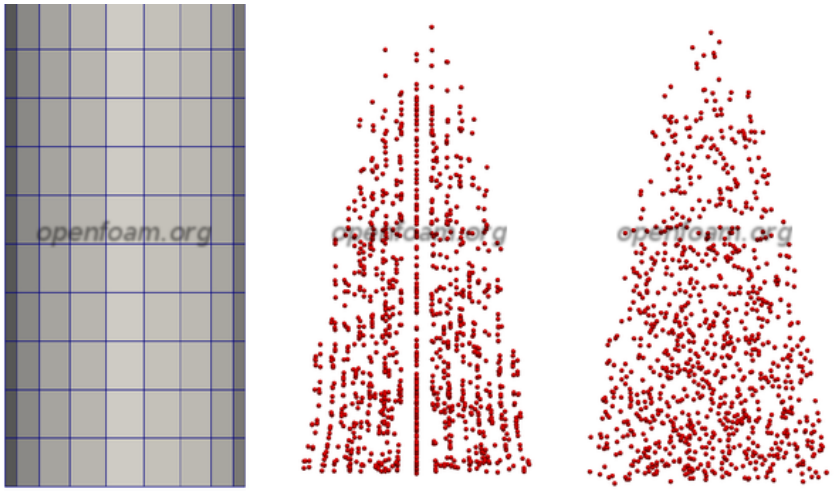
プロペラ周りのキャビテーション
propeller

図出典 [OFT]

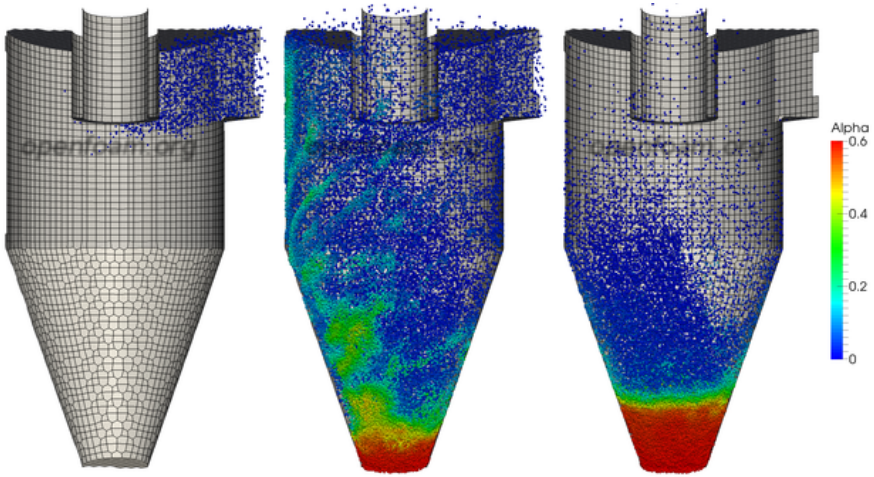
粒子計算のチュートリアル例

図出典： [OFF]

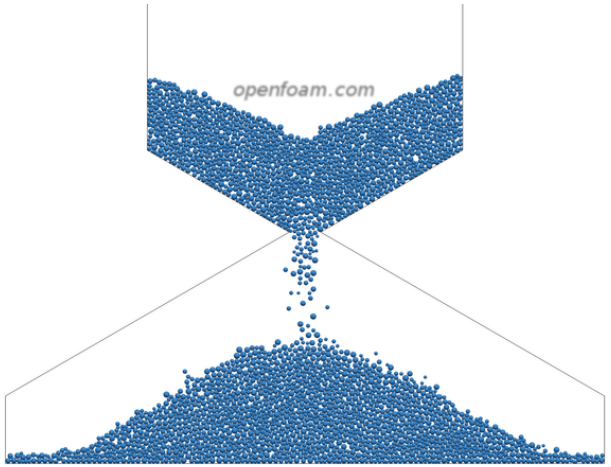




Particle Tracking

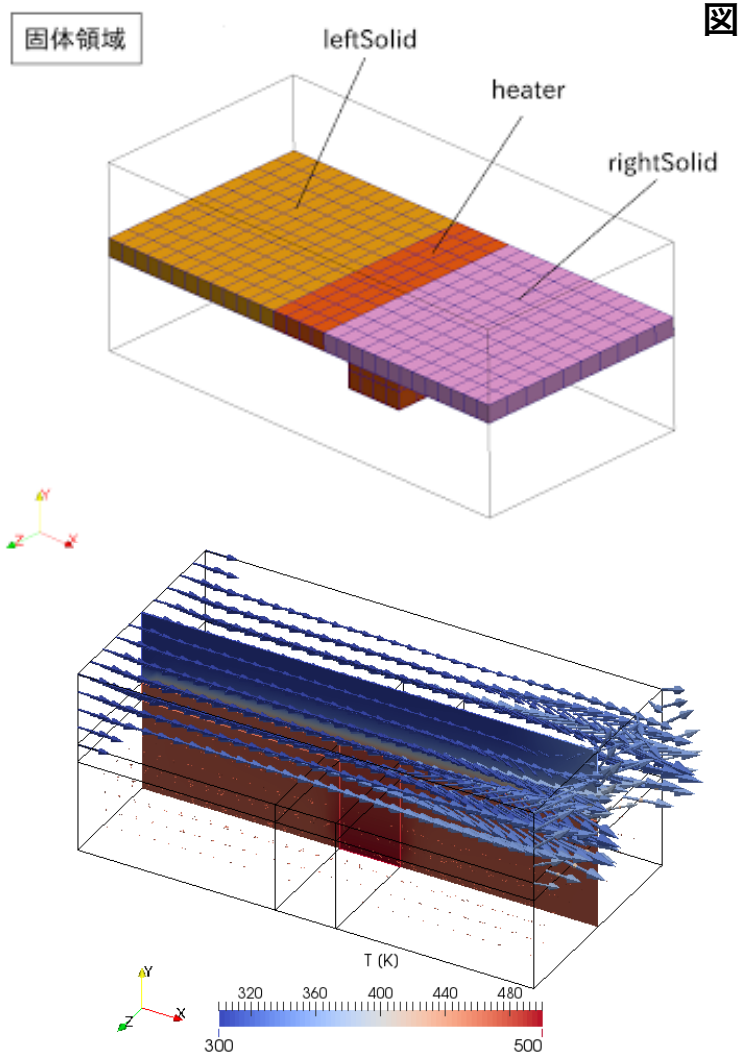


Multiphase Particle-in-Cell
(MP-PIC)



Discrete Element Modeling
(DEM)

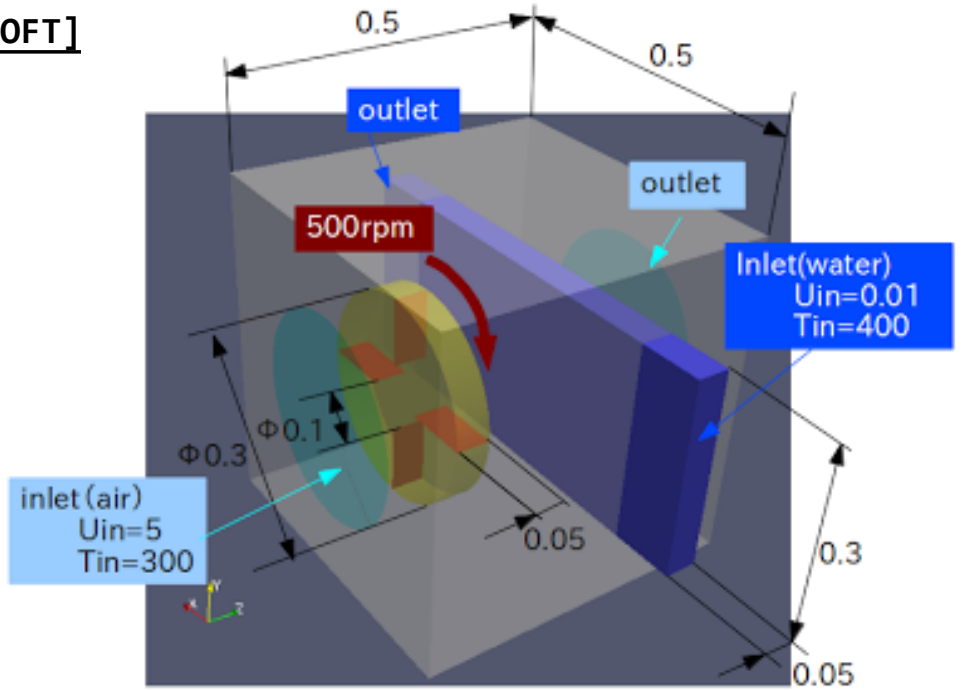
連成熱伝達解析(CHT)のチュートリアル例



連成熱伝達解析

multiRegionHeaterRadiation

図出典 [OFT]



回転する熱交換機解析(MRF)

heatExchanger

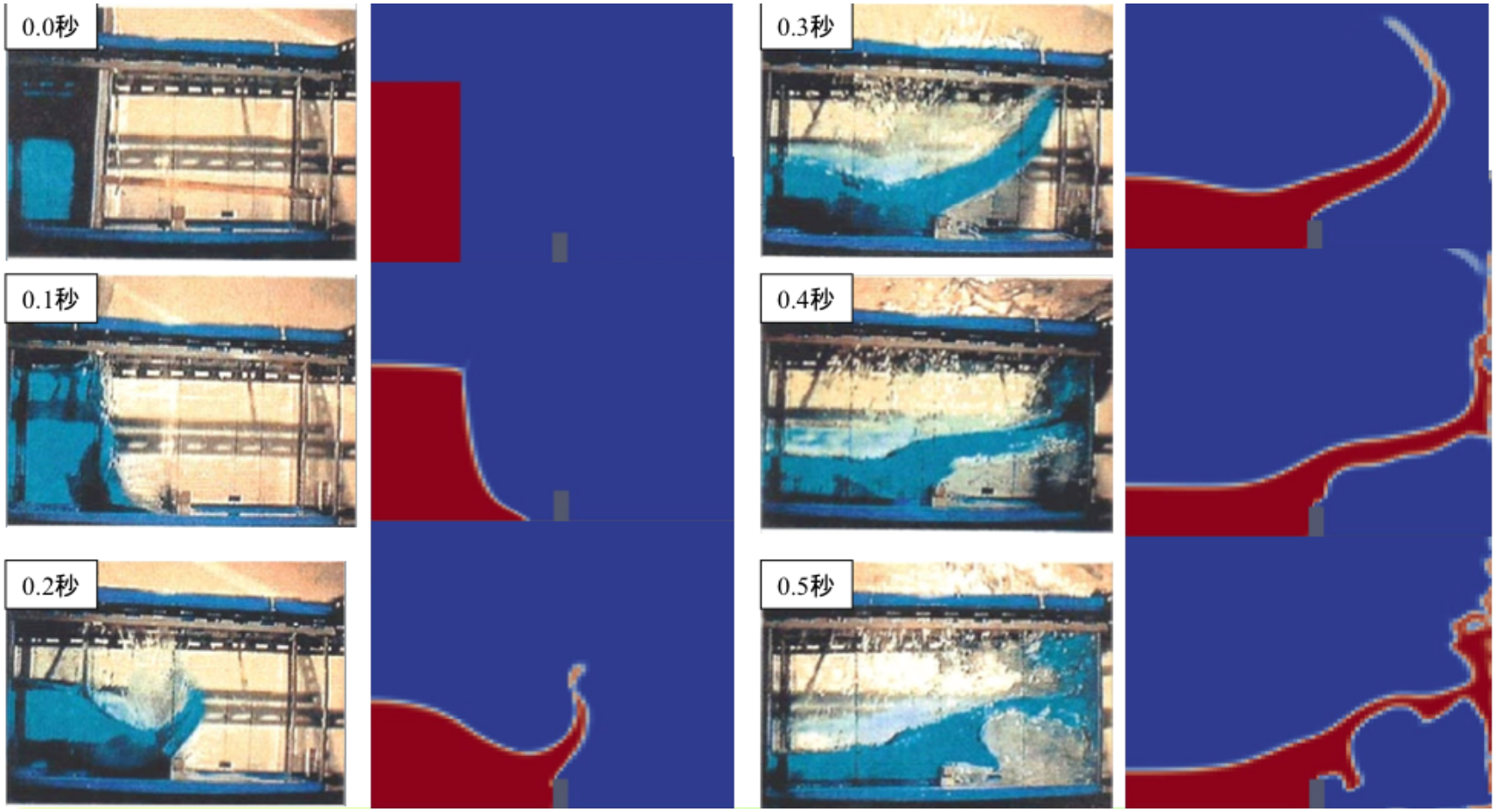
OpenFOAMのCHT解析における欠点

- エネルギー保存式を全領域で連成せず，領域毎に解くので収束が遅い
- 形態係数を用いた放射解析の精度が悪い
- 熱収支計算が容易ではない(熱解析共通)

Dam breakのチュートリアル例

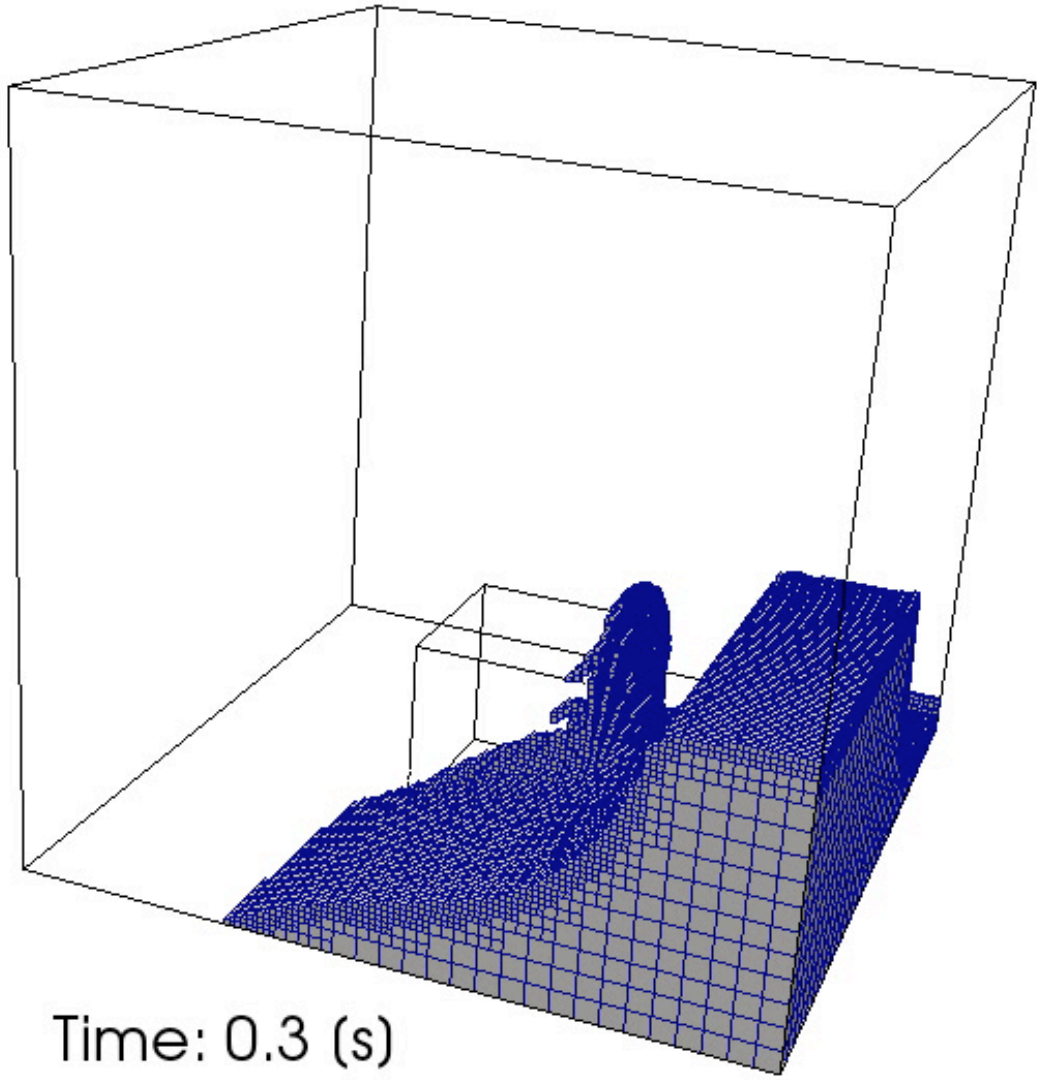
Koshizukaら[1]によるdam break(ダム崩壊)の実験をVOF(Volume of Fluid)法の二相流ソルバinterFoamを用いて，二次元層流モデルで解析したもの

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

Dam breakのチュートリアル例(動的格子)



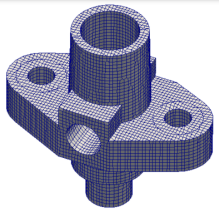
damBreakWithObstacleチュートリアル

- 格子が粗いと，液面(液相と気相の間の相界面)の形状の再現精度が著しく悪化する.
- 解析領域で全て格子を細かくすると，計算負荷が高くなる.
- 本チュートリアルでは，動的格子の機能を用いて，格子の液相率(VOF値)が0.001から0.999である相界面の格子のみ，格子幅を1/4に細分割し，格子数をあまり増加させずに，液面形状の再現性を上げている.

OpenFOAMで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(サードパーティ)
[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]

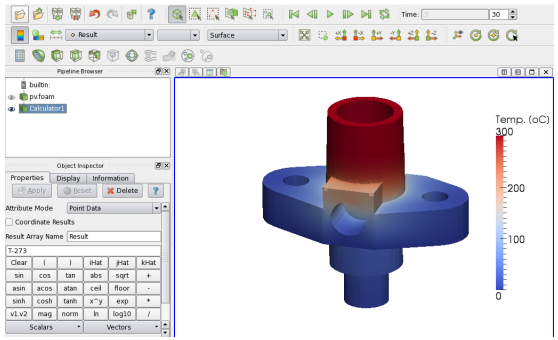
領域分割(並列計算時)
[decomposePar]

解析ソルバ
[icoFoam等]

領域統合(並列計算時)
[reconstructPar]

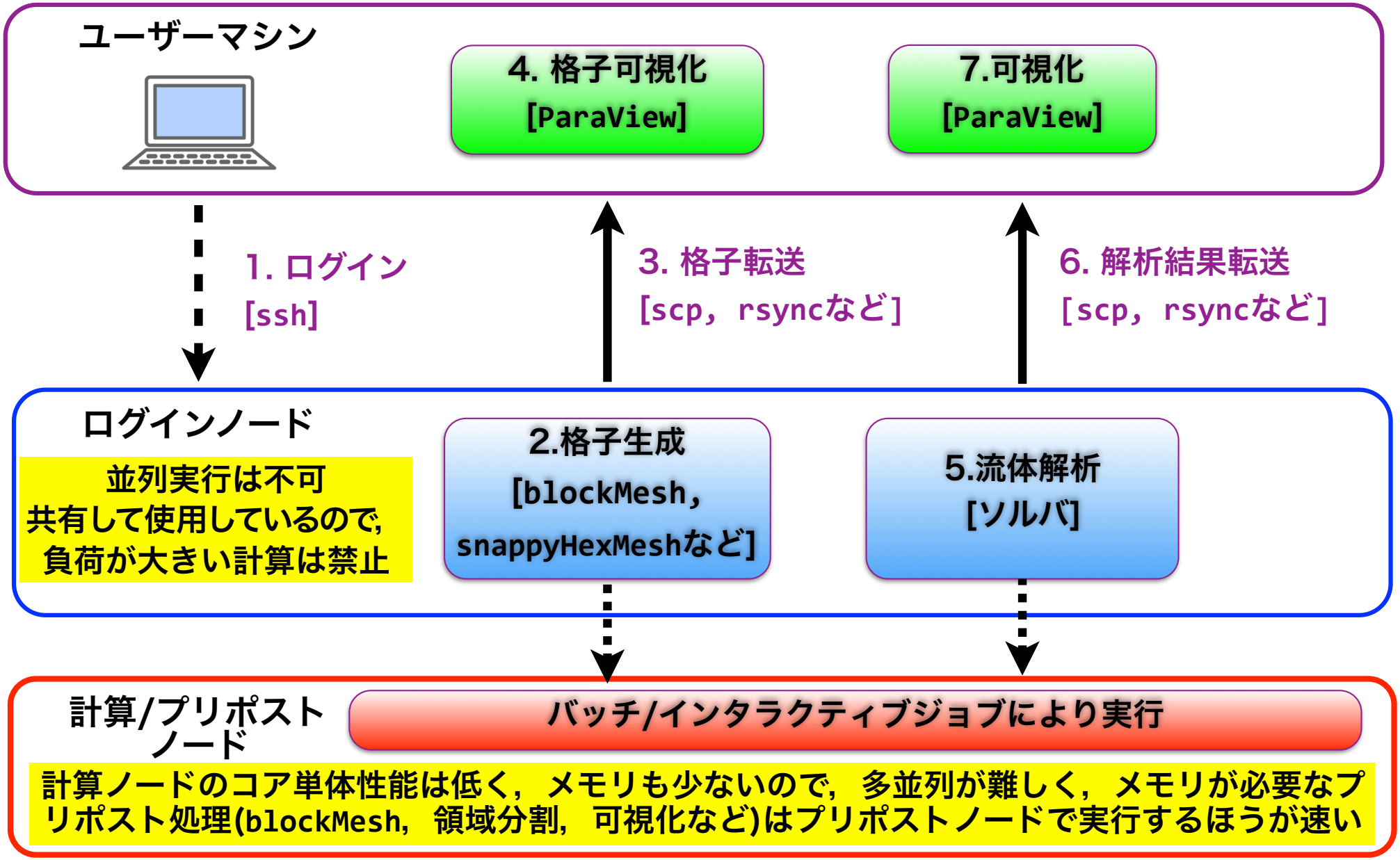
後処理(可視化など)

可視化(サードパーティ)
[ParaView, Visit,
商用可視化ツール等]



様々な結果後処理
[sample等]

OFPでのOpenFOAMの代表的な解析手順



OpenFOAMの稼働環境

Linux, Mac, Windows機で動作

FOCUS
東工大 TSUBAME
東大 Reedbush-U, H, L
名古屋大学FX100
JCAHPC Oakforest-PACS
九州大学ITO
大阪大学 OCTOPUS，etc.

Laptop　PC　クラスタ　クラウド　スーパーコンピュータ　フラッグシップ
～100万格子　～1000万格子　～10億格子　～1000億格子

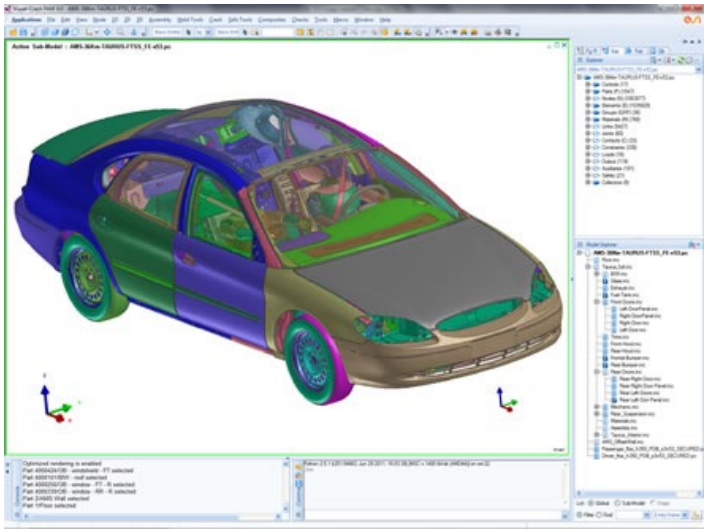
Amazon EC2(GPU機あり)
Microsoft Azure(GPU機あり)
富士通TCクラウド
Etc.

京 (SPARC64機)
RISTがHPCI課題の
京ユーザ向けに最適
化を支援

OpenFOAMの主な派生版(Fork)

- **商用版**

図出典：ESI (https://www.esi.co.jp/news/2014/PressRelease_0128.html)



Visual-CFD(ESI)

✓ **HELYX(Engys):** OF拡張版+GUI

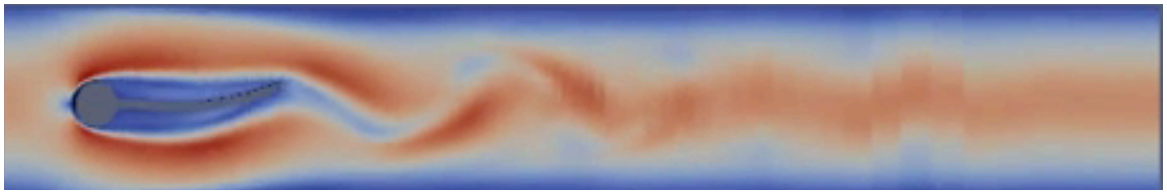
✓ **iconCFD(IDAJ, ICON):** OF拡張版+GUI

✓ **Visual-CFD(ESI):** GUI

- **オープンソース版**

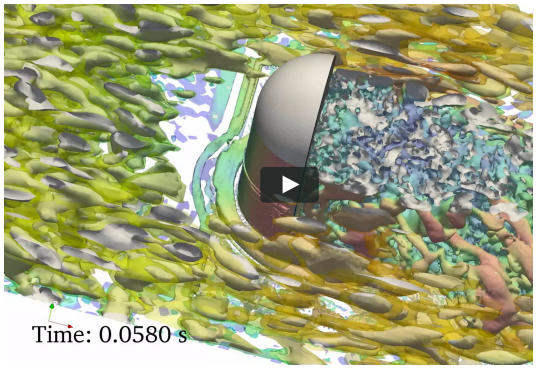
✓ **HELYX-OS(Engys):** GUI

✓ **foam:** Hrvoje Jasak(クロアチア ザグレブ大学教授，Wikki社 代表)が主導するコミュニティベース版．FSIやBlock coupledソルバ等の公式版に無い機能を実装



foamの 流体・構造連成(FSI)

✓ **OpenFOAM+(ESI):** 安定化と機能拡張

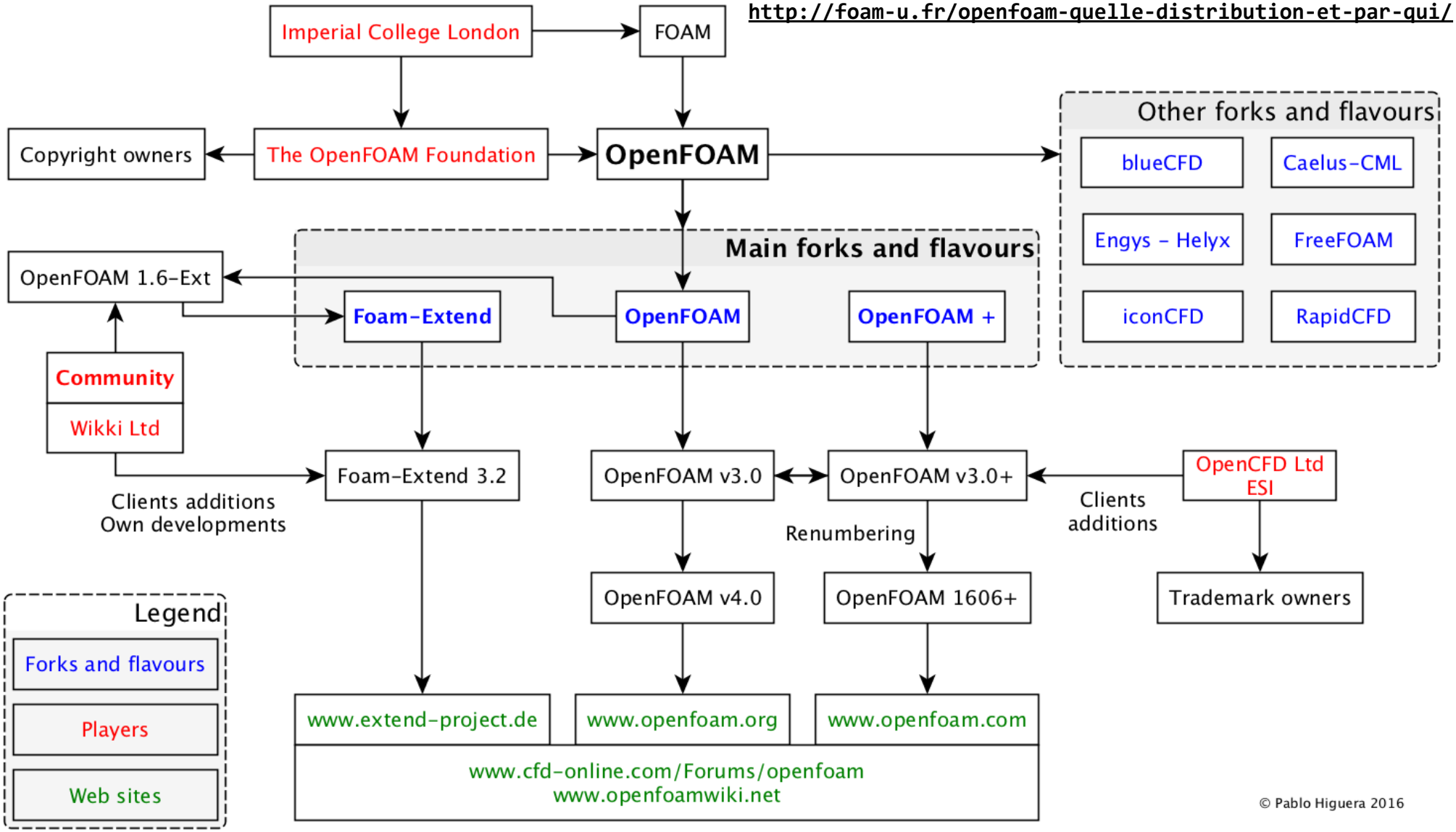


OpenFOAM+の変動風生成

図出典：www.openfoam.com

OpenFOAMの派生図

図出典：olaFoam
http://foam-u.fr/openfoam-quelle-distribution-et-par-qui/

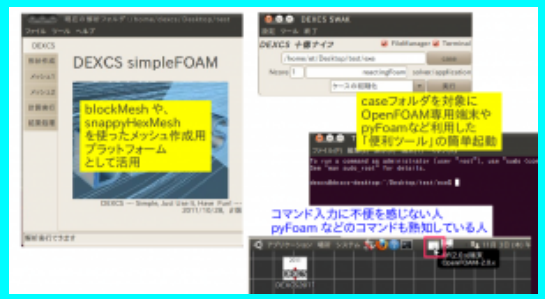


OpenFOAMの課題

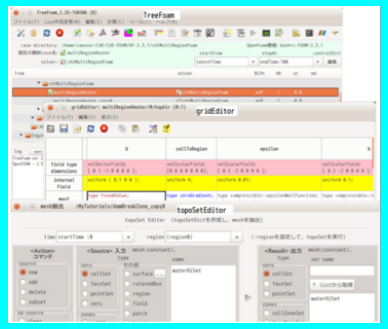
- 設定用GUIが無く，全てテキストファイルで設定する必要があるが，詳細な公式マニュアルがほとんど無い → ソースコードを読まないと詳細な設定方法がわからないので，初心者には設定が困難．解析条件に応じた推奨設定も不明

- GUIに関しては商用(iconCFD, Visual-CFD)，オープンソース(HelyxOS, DEXCS, TreeFoam)などが続々登場してきた

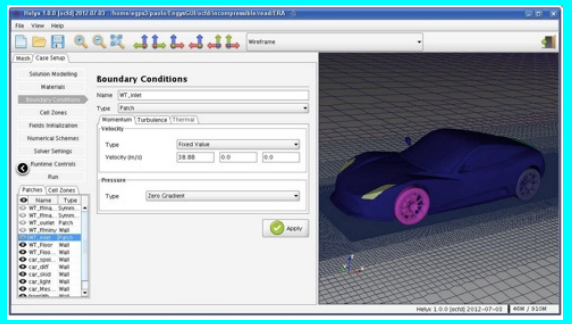
オープンソースGUI例



DEXCS (野村氏作)



TreeFoam(藤井氏作)



HelyxOS(engys社)

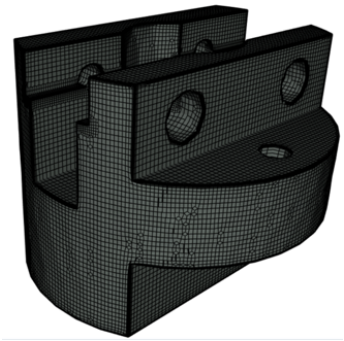
OpenFOAMの課題

- メッシャー(blockMesh, snappyHexMesh)で格子を生成するのが遅く，質の良いレイヤを貼るのは困難 → 初心者は格子生成で断念

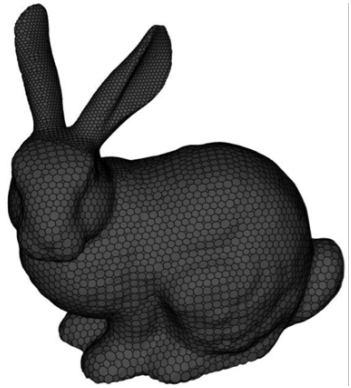
- Pointwise，HEXPRESSなどの商用メッシャーはOpenFOAMの格子を出力できるようになっている.
- Helyx, iconCFDなどの商用ForkではsnappyHexMeshの機能を改善
- オープンソースでハイブリッド並列，レイヤ付加性能に優れたOpenFOAM用メッシャーcfMeshも登場(v1712から取り込まれた)

cfMesh

cfMeshによるレイヤ付き六面体格子

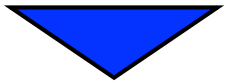


cfMeshによるポリヘドラル格子



OpenFOAMの課題

- ハイブリッド並列の非対応 → GPGPUやXeon Phi等のメニーコア機で非効率．ハイブリッド並列よりMPIプロセス数が多くなるので，MPIプロセス間の通信コストがかかる．



以下のような様々な研究や実装が行われている

- Amani AlOnazi: Design and Optimization of OpenFOAM-based CFD Applications for Modern Hybrid and Heterogeneous HPC Platforms, Master Thesis, King Abdullah University of Science and Technology, 2014, URL
- 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, URL
- 内山，フックら: OpenFOAMによる流体コードのHybrid並列化の評価，情報処理学会, 2015, URL
- 内山，フックら: 非構造格子に対応したPCG法のthread並列化手法，情報処理学会, 2016, URL
- 山岸，井上ら: OpenFOAMのメニーコア・GPUへの対応に向けた取り組みの紹介，オープンCAEシンポジウム, 2017, URL
- 富岡，吉藤ら: OpenFOAMスレッド並列化のための基礎検討, オープンCAEシンポジウム, 2017, URL
- 今野: OpenFOAMにおけるCommunication-Avoiding CG法の実装と性能評価, オープンCAEシンポジウム, 2017, URL
- simFlow社: RapidCFD(NVIDIA CUDA用フルGPU版OpenFOAM), オープンソース, URL

並列計算による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

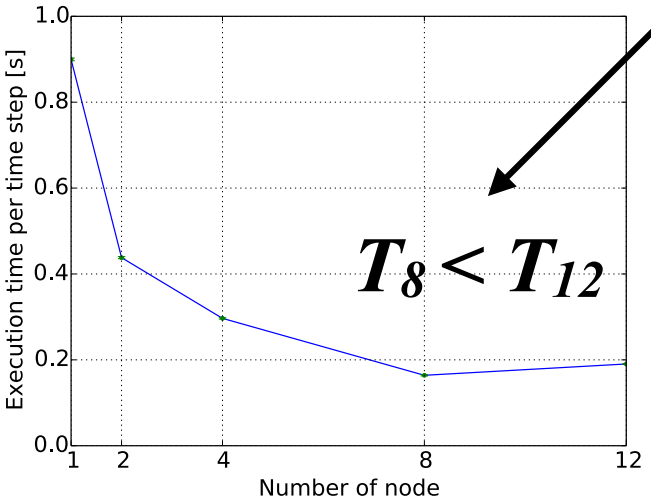
T_S ： ベースとなるプロセス数(1プロセス，1ノード等)での実行時間

T_P ： ベースとなるプロセス数×Pでの実行時間

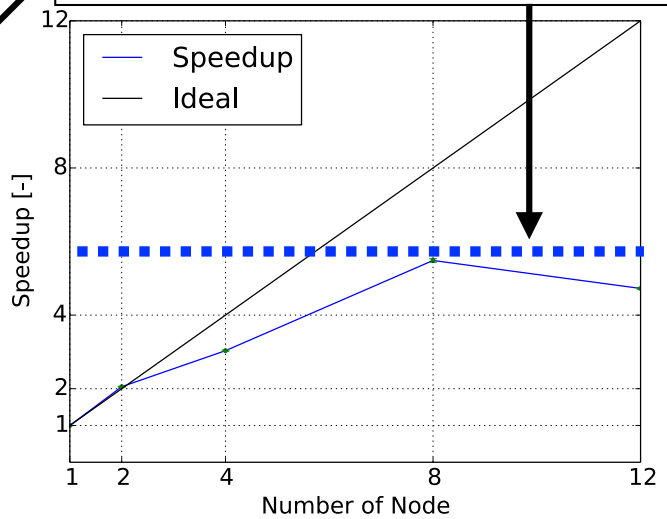
- 並列化効率 E_P

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

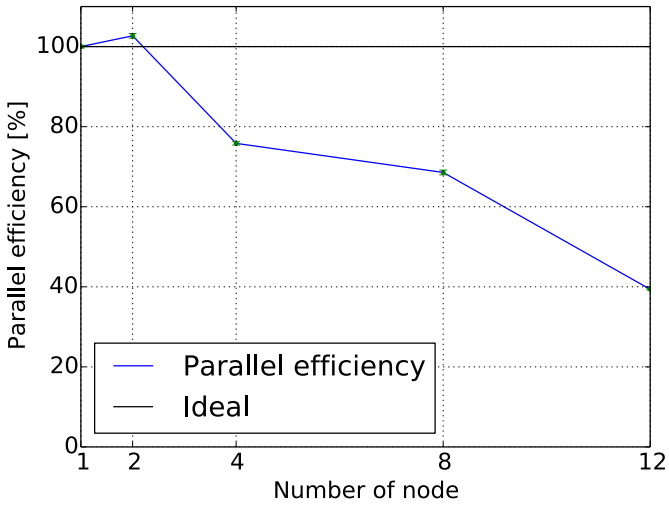
このケースでは12ノード以上使用しても非効率
→台数効果の飽和(Saturation)をベンチマークテストで事前に把握し，非効率な計算を行わないことが重要



実行時間



台数効果(スピードアップ)



並列化効率

並列計算の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・スムーズ方法に強く依存
  - ✓ 線型ソルバーの速さは並列数にも依存
    - 並列数小→GAMG(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 > GAMG
  - ✓ 問題にも依存するが，PCGの前処理としてGAMGを用いると，広い並列数の範囲で概ね最速．なお，GAMGのスムーザはDIC(不完全コレスキー分解)またはDICGaussSeidel(DICとガウスザイデル法の併用でより安定)が概ね最速
  - ✓ さらに並列数が増えると，前処理がDICのPCGが概ね最速

OpenFOAMの並列計算手法

OpenFOAMの並列計算手法

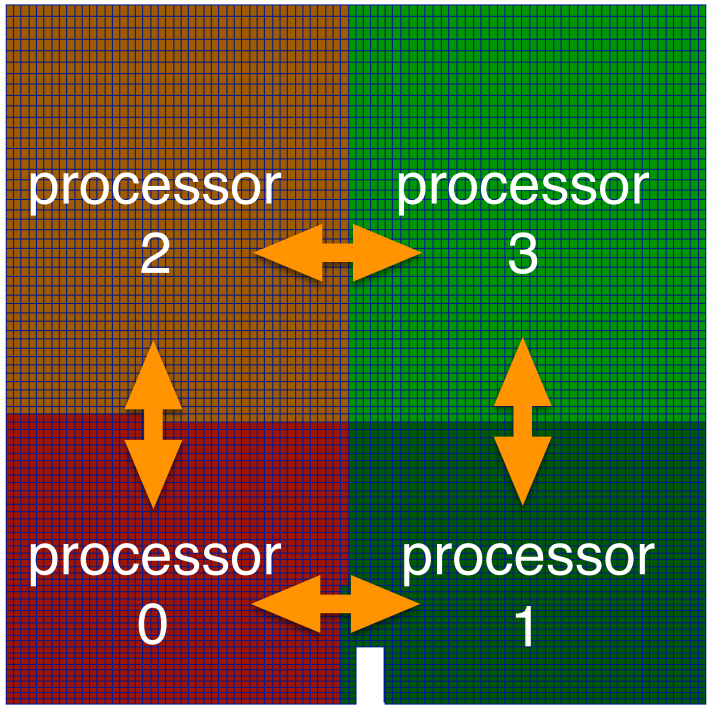
1. 格子生成
2. 領域分割 (decomposePar)
3. MPI並列でソルバを実行

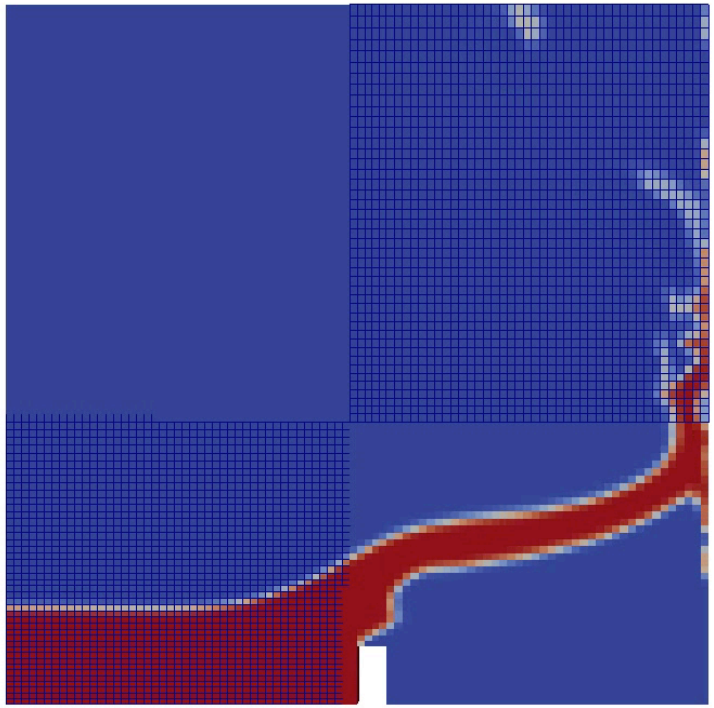
(MPI+OpenMPのハイブリット並列は標準では未実装．研究例有り)

4. 領域毎の解析結果を再構築 (reconstructPar)

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，計算効率を低下させる



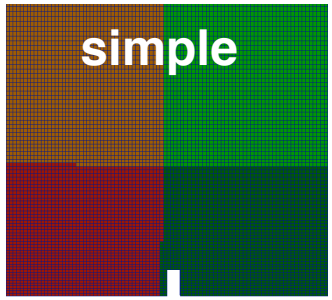


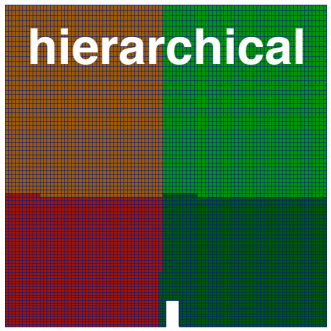
damBreakFineチュートリアル

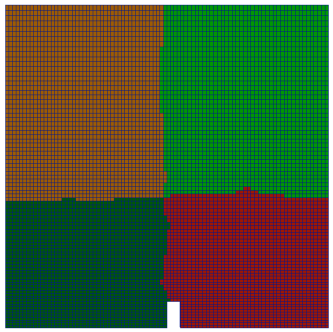
領域分割の設定

system/decomposeParDict

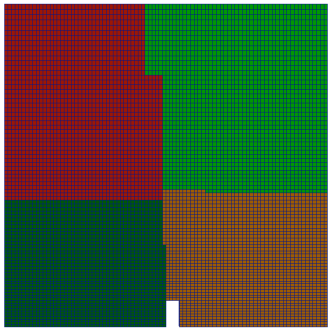
```
numberOfSubdomains 4;        //領域分割数

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



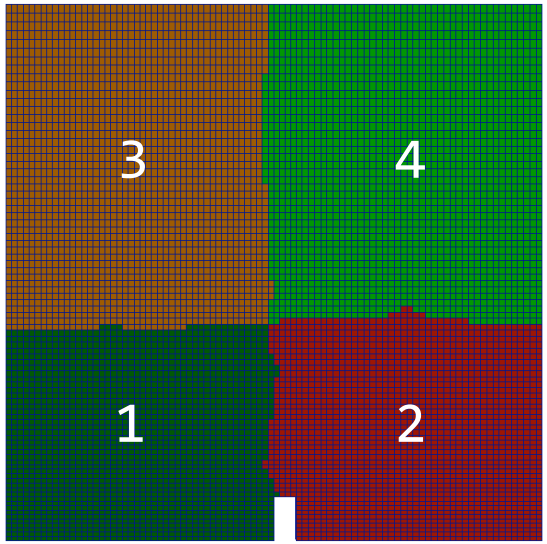
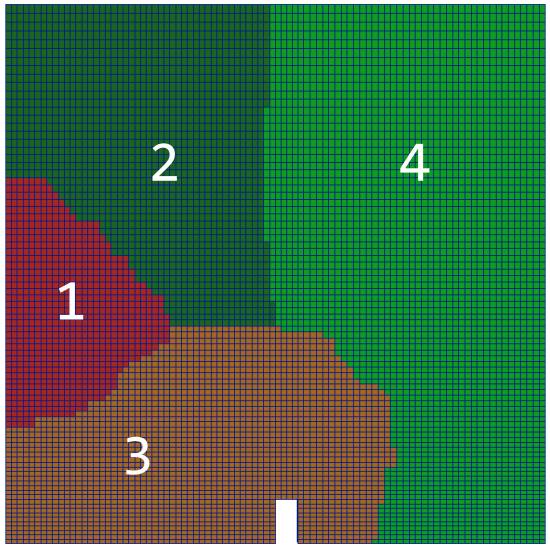
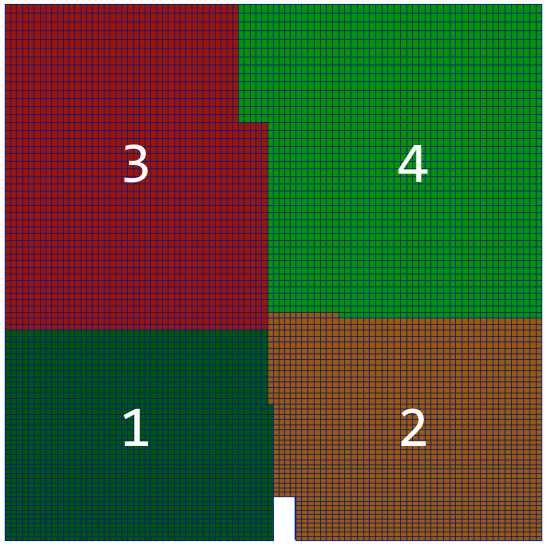
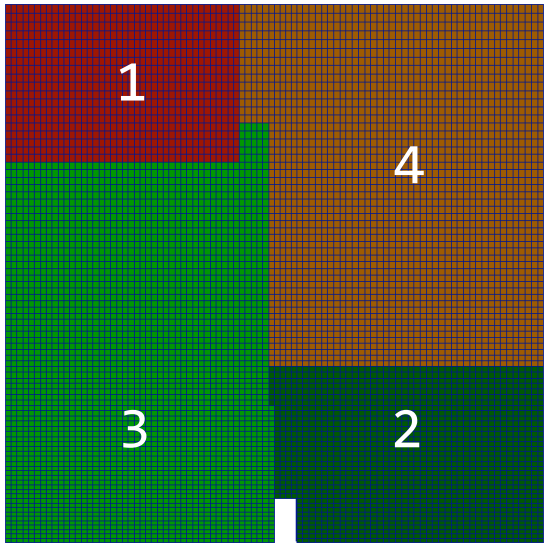
sctoch

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		

system/decomposeParDict

```
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
//省略時は重み無し
}
```

重み付けは，ノード間で性能が異なる場合に用いる場合があるが，通常は用いない

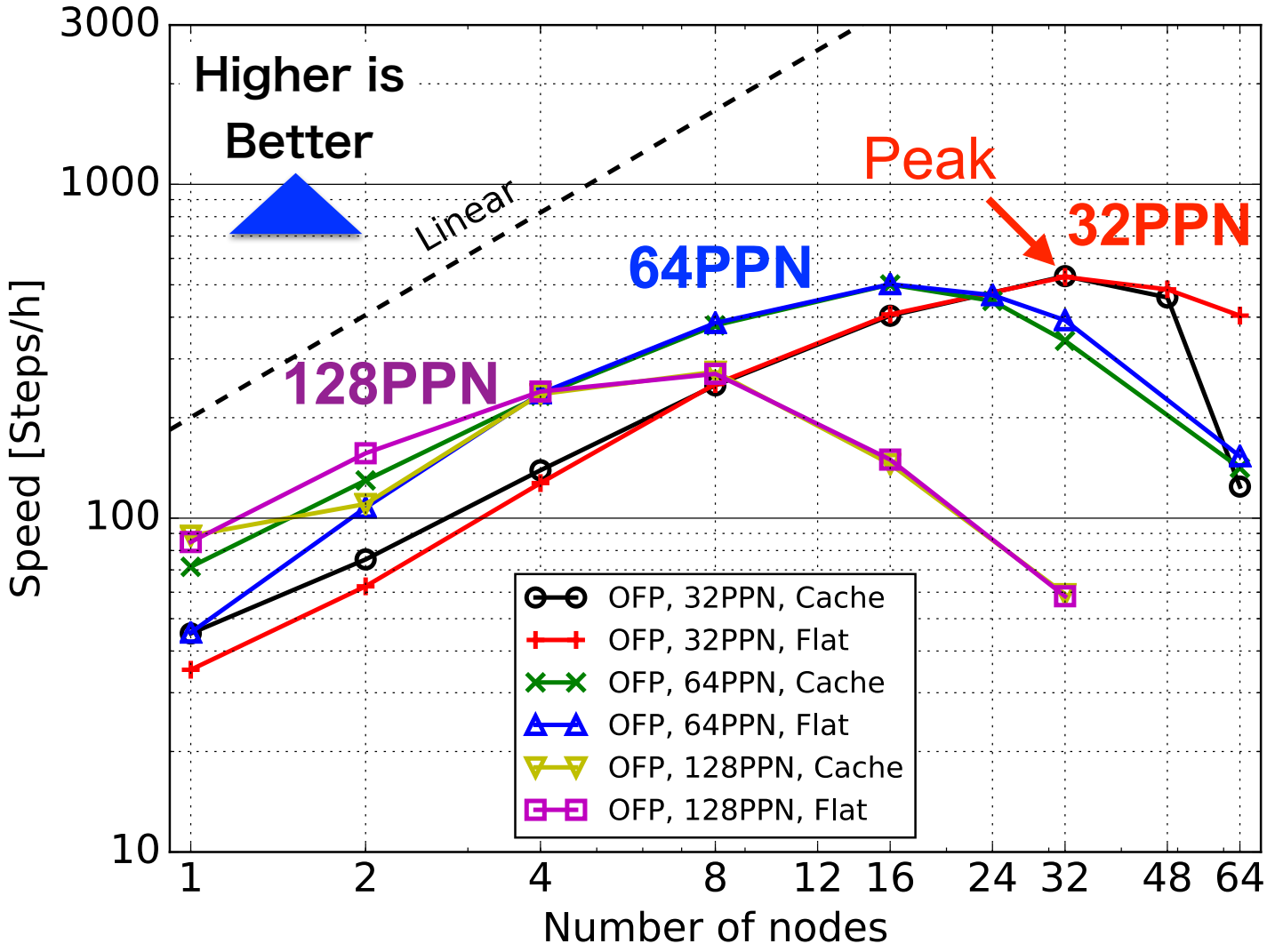
ボックスファンベンチマーク

- ボックスファン(産応協HPCものづくりWS作成の共通ベンチマーク問題) [1]
  - ✓ 翼枚数：9 [枚]
  - ✓ プロペラ外径：111[mm]
  - ✓ 回転数：3000 [rpm]
- 非定常RANS解析(ソルバ：OpenFOAM-v1612+，pimpleDyMFoam)
- 乱流モデル ：k-ω SST，移流項スキーム：1次風上
- 時間刻み：5×10^{-5}[s] (400step/rev)
- 2〜21ステップのCPU時間(Execution time)から1時間あたりのステップ数を算出
- 圧力線型ソルバ : PCG(前処理FDIC)，許容残差10^{-7}
- 圧力以外の線型ソルバ : Gauss Seidel，許容残差10^{-5}
- 領域分割手法：scotch

[1] 今野: OpenFOAMによる流体解析ベンチマークテスト FOCUS・クラウド・スパコンでのチャネルおよびボックスファン流れ解析, 2017

Oakforest-PACSの解析速度

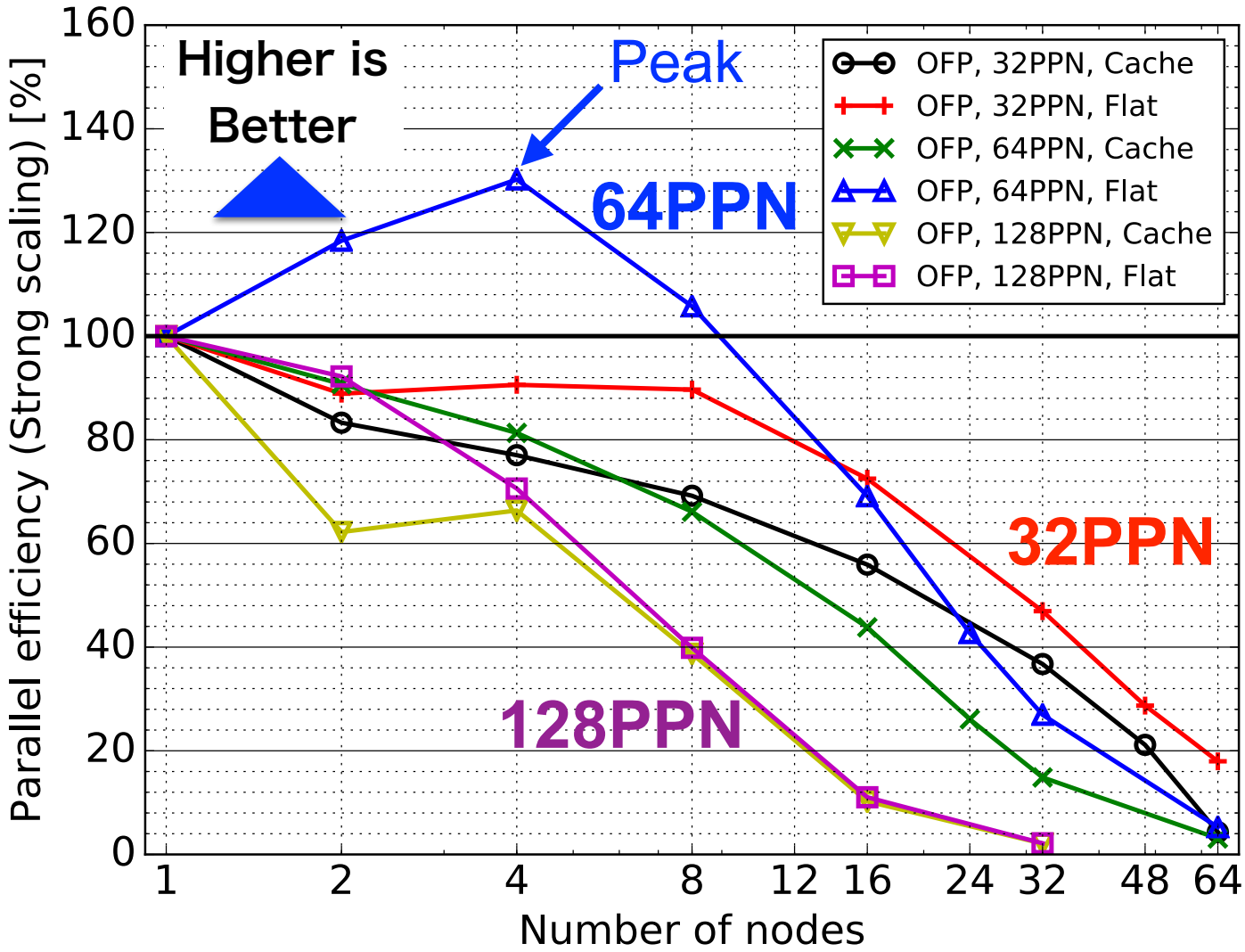
コンパイラ：Icc 2017.4，KNL用最適化：-O3 -DvectorMachine -xmic-avx512
MPI：Intel MPI 2017.3，タイル0除外，Flatモードではlibhbm使用



- 最適化オプションはKNL用がデフォルト設定(-O3)より僅かに速い
- 2ノード以下では，PPN(ノードあたりの使用プロセッサ数)がHyper thread利用の128が高速
- 多ノードではCacheよりFlatのほうが高速

解析速度最大条件：
32PPN，32ノード(1024プロセッサ)，Flat

Oakforest-PACSのstrong scaling並列化効率

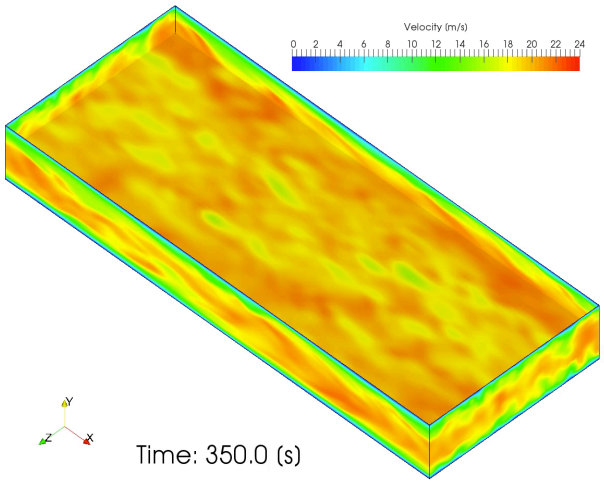


- Flat > Cache
- ～8ノード：
  64 > 32 >128PPN
- 16ノード～：
  32 > 64 >128PPN

並列化効率最大条件：
64PPN，4ノード(256プロセッサ)，Flat

オープンCAE学会チャネル流ベンチマーク

チャネル流れ ($Re_\tau = 110$)



格子数約3M (0.37, 24M 等他の格子数の結果はレポジトリ参照)

- ✓ メッシュ生成に時間を要しない
- ✓ 構造格子のため，格子数変更が容易
- ✓ 圧力と速度のみ解くので，「圧力線形ソルバの解析時間が支配的」という非圧縮性流体解析の特性を素直に示す

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

L_x, L_y, L_z：各方向のチャネル幅 [m]

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): 一定の圧力勾配

主流方向(x), スパン方向(z): 周期境界

ソルバ: OpenFOAM pimpleFoam

乱流モデル: 無し(laminar)

速度線型ソルバ: BiCG (前処理DILU)

圧力線型ソルバ: PCG (前処理DIC)

領域分割手法：scotch(周期境界面は同領域)

- 2〜51ステップの平均CPU時間(Execution time)から1時間あたりのステップ数を算出

計測システム

機関	システム (略称)	CPU [Intel Xeon] (周波数[GHz])	CPU (コア)	倍精度性能 [GFlops]	メモリ[GiB] (帯域幅[GB/s])	インターコネクト (帯域幅[Gbps])
JCAHPC	Oakforest-PACS (OFP)	Phi 7250 (1.4)	1(68)	3046	96(115.2), MCDRAM 16(490)	Intel Omni-Path (100)
東京大学	Reedbush-U (RBU)	E5-2695 v4(2.1-3.3)	2(36)	1210	256(76.8×2)	Infiniband EDR(100)
FOCUS	A(FA)	L5640(2.26)	2(12)	108	48(25.6×2)	Infiniband QDR(40)
	D(FD)	E5-2670 v2(2.5)	2(20)	400	64(51.2×2)	Infiniband FDR(56)
	F(FF)	E5-2698 v4(2.2)	2(40)	1152	128(76.8×2)	
	H(FH)	D-154(2.1)	1(8)	205	64 (34.1)	10GbE(10)×2 or 4
Amazon	c4.8xlarge(c48x)	E5-2666 v3(2.9)	2(18)	310	60(不明)	10GbE(10)
Microsoft	Azure A9(A9)	E5-2670(2.6)	2(16)	333	112(不明)	Infiniband QDR(40)
	Azure H16r(H16r)	E5-2667 v3(3.2)	2(16)	691	112(不明)	Infiniband FDR(56)
大阪大学	OCTOPUS(OCT)	6126 (2.6)	2(24)	1997	192(255.9)	Infiniband EDR(100)
九州大学	ITO A(ITO)	6154 (3.0-3.7)	2(36)	3456	192(255.9)	Infiniband EDR(100)
Oracle	BM HPC 2.36(ORA)	6154 (3.0-3.7)	2(36)	3456	384(255.9)	RoCE v2 (100)

注) 倍精度性能は，機関のWEBページに明記された値，または，AVXの動作周波数を考慮して算出したWEBページから引用

使用OpenFOAMバージョン・コンパイラ・MPI

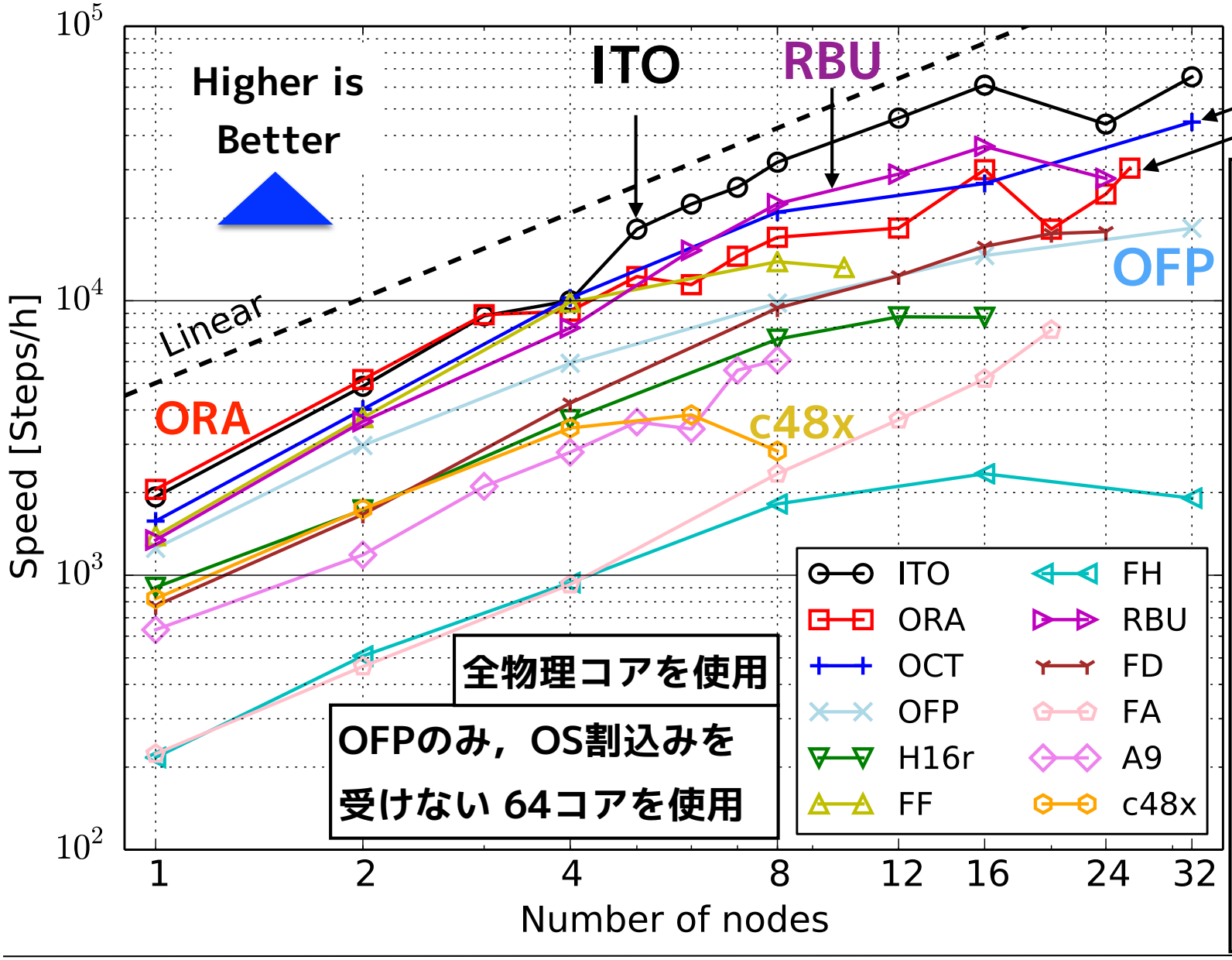
機関	システム	バージョン	コンパイラ	MPI
JCAHPC	OFP(Flatモード)	OpenFOAMv1612+	Icc 2019.1	Intel MPI 2019.1
大阪大学	OCT		Icc 2018.2	Intel MPI 2018.2
東京大学	RBU	OpenFOAM 2.3.0	Gcc-4.8.5	SGI MPI 2.14
九州大学	ITO			Intel MPI 2018.1
Oracle	ORA			Intel MPI 2018.1
Microsoft	H16r			Intel MPI 5.0.3
	A9		Gcc 4.8.3	Intel MPI 5.1.1
Amazon	c48x			OpenMPI 1.8.5
FOCUS	FOCUS			OpenMPI 1.6.5

実行時の主な設定

OFP) WM_COMPILER=IccKNLでコンパイル．unset KMP_AFFINIY KMP_HW_SUBSET=1T I_MPI_PIN_PROCESSOR_EXCLUDE_LIST=0,1,68,69,136,137,204,205
I_MPI_PIN_DOMAIN=4 MPI_BUFFER_SIZE=6291456 HBM_THRESHOLD=4 HBM_SIZE=240 LD_PRELOAD=libhbm.so
OCT) I_MPI_DYNAMIC_CONNECTION=0 I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
ITO) I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
ORA) I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
H16r) I_MPI_FABRICS=shm:dapl I_MPI_DAPL_PROVIDER=ofa-v2-ib0 I_MPI_DYNAMIC_CONNECTION=0 I_MPI_DAPL_DIRECT_COPY_THRESHOLD=655360
A9) I_MPI_FABRICS=shm:dapl I_MPI_DAPL_PROVIDER=ofa-v2-ib0 I_MPI_DYNAMIC_CONNECTION=0
c48x) mpirun -bind-to core FOCUS) mpirun -bind-to-core -mca btl openib,sm,self

OFPではMC-DRAMの設定が必要だが，グリッドサーチによる最適値を用いた．

各システムの解析速度比較



最高速

・1～3ノード：ORA (Skylake 6154, RoCE v2)

・4～32ノード：ITO (Skylake 6154, IB EDR)

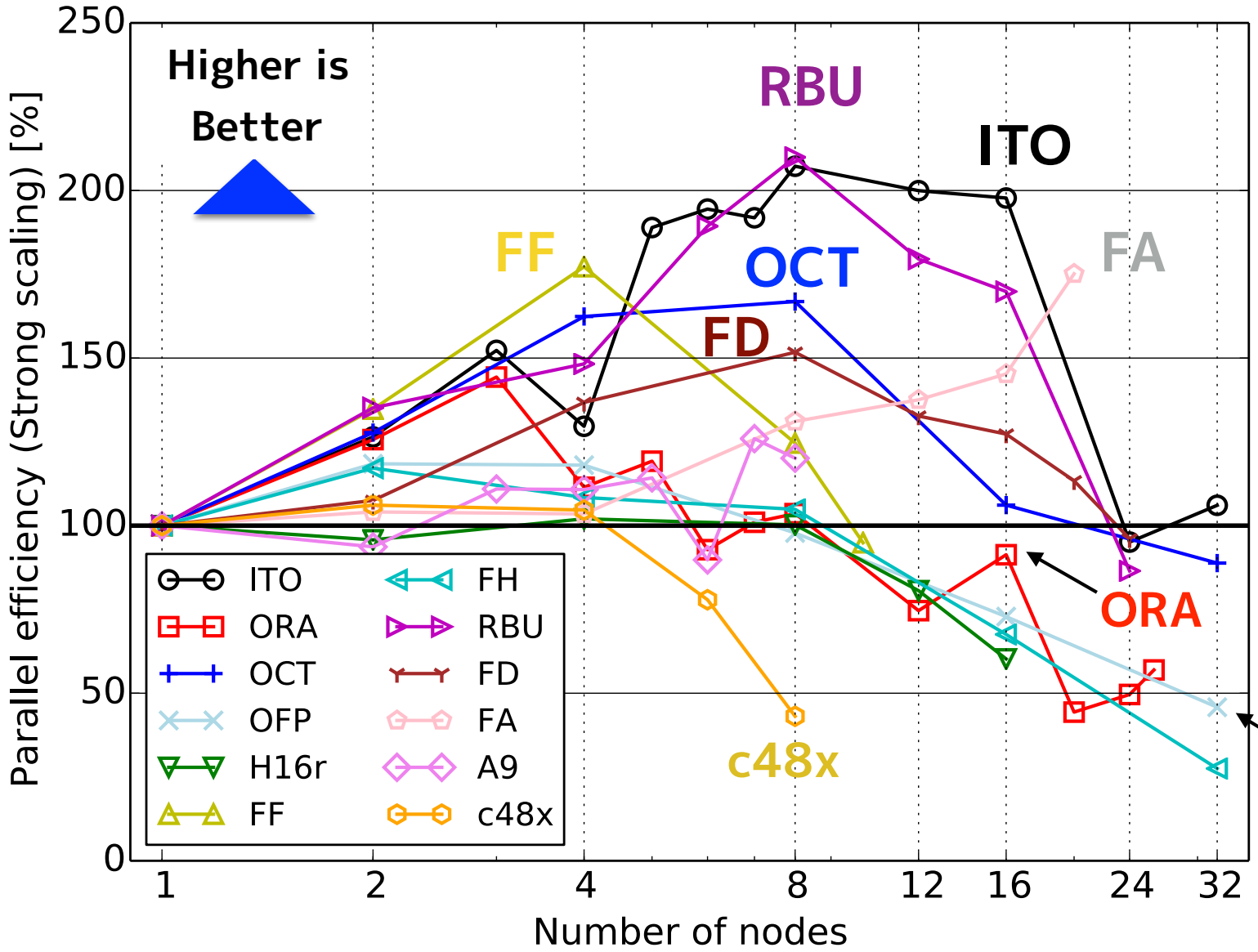
次点

・OCT (Skylake 6126, IB EDR)

・RBU (Broadwell, IB EDR)

10GbEであるc48xは6ノードで飽和

各システムのStrong scaling並列化効率比較



- InfinibandのRBU，ITO，OCT，FD，FAは16ノードまでスーパリニア
- RoCE v2のORAは16ノードまで概ねスケール
- OFPは12ノード以上で悪化
- 10GbEのc48xは6ノードから劣化

OFPフラットモードにおけるlibhbmの設定

- OFPのFlatモードにおけるOpenFOAMの実行には，libhbmを用いている．
- libhbmは，memkindのautohbmと同様にHBM(High-Bandwidth Memory)のメモリ確保をするが，アプリケーション開始時に，プロセス毎に固定サイズのヒープを確保し，アプリケーションが終了するまで開放しない点が異なる．
- libhbmを用いた実行時には，以下の表に示す環境変数を適切に指定する必要があり，libhbmの配布先(レポジトリ)では推奨値が示されている．

環境変数	説明	推奨値	推奨値の根拠
HBM_SIZE	ヒープサイズ	256MB	HBMサイズ(16GB) / プロセス数(今回64)
HBM_THRESHOLD	HBMに割り当てる下限サイズ	16KB	HBMの断片化を避けると共に，2つのメモリのバンド幅を活かす(サイズが小さい場合，レイテンシが小さいDDR4を使う)．
MPI_BUFFER_SIZE	MPIバッファのサイズ	1MB	大抵のソルバでは大きな値を必要としないので，MC-DRAMを浪費しないように1MB程度に小さくする．

libhbm設定のベイズ最適化(格子数3M)

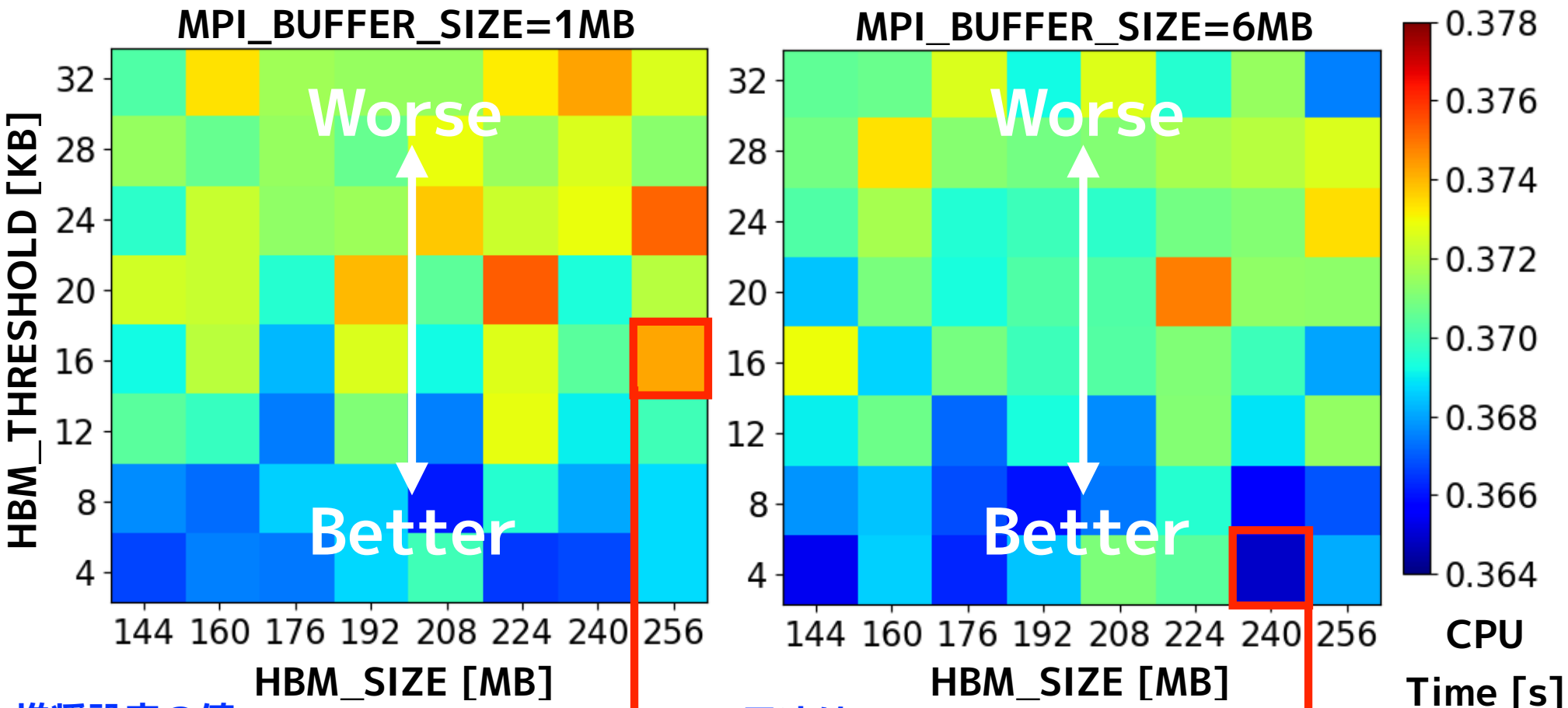
- MPIバッファサイズやlibhbm設定の最適値は問題に依存し，チャネル流での最適値は推奨値と異なる可能性があるので，グリッドサーチの最適化を行った．
- 一方，CFD解析の試行時間は長いので，よりパラメータ数が多く，パラメータ空間が広い場合，グリッドサーチの最適化コストは莫大となる可能性がある．
- そこで，ハイパーパラメータ自動最適化ツールOptunaを用いたベイズ最適化を行い，以下に示すパラメータ空間におけるグリッドサーチとの比較により，その収束性や効率を検討した．

表：グリッドサーチおよびベイズ最適化のパラメータ空間 (総数512)

環境変数	設定値	サンプル数
HBM_SIZE	144MB～256MB，16MB刻み	8
HBM_THRESHOLD	4～32KB，4KB刻み	8
MPI_BUFFER_SIZE	1MB～8MB，1MB刻み	8

- 同パラメータでの解析を5回以上行い，CPU時間の平均値を最適化した．
- 格子数3Mとし，Strong scalingが1に近い8ノード・512並列を対象とした．

グリッドサーチによる平均CPU時間ヒートマップ



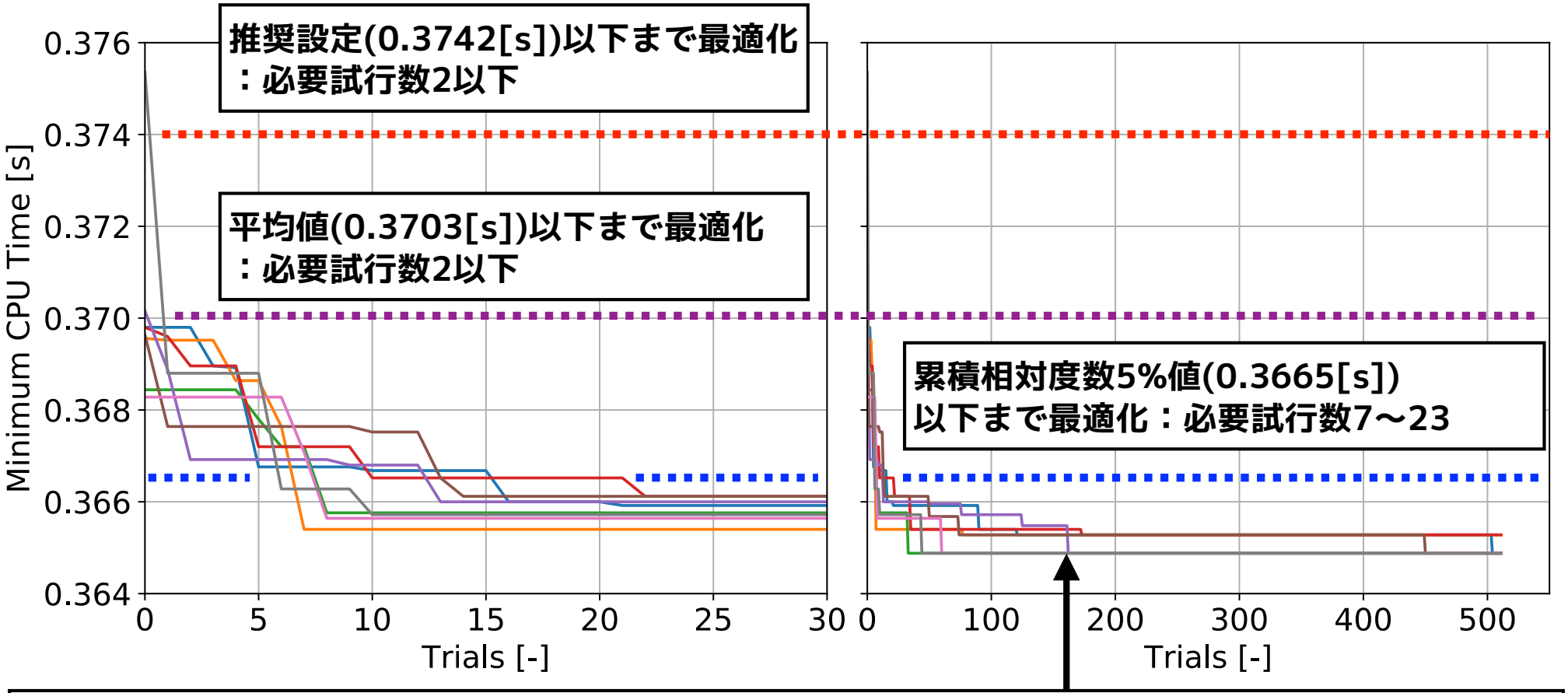
推奨設定の値

CPU Time: 0.3742[s]
累積相対度数=97%(非常に遅い)
MPI_BUFFER_SIZE=1MB
HBM_SIZE=256MB, THRESHOLD=16KB

最適値

CPU Time: 0.3649[s]
推奨設定より約2.5%高速
MPI_BUFFER_SIZE=6MB
HBM_SIZE=240MB, THRESHOLD=4KB

試行数512×8回のベイズ最適化結果



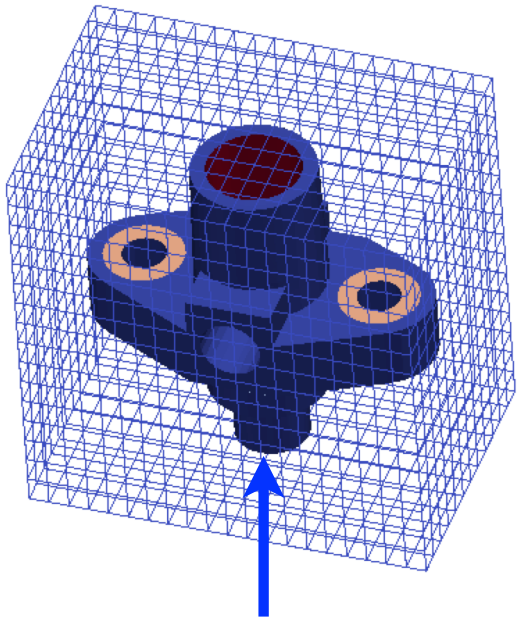
8回中6回は最適値0.3649[s]まで概ね160試行で到達するが，2回は最適値ではなく2位の値となる．
ただ，その値0.3653[s]は最適値より0.1%大きいだけであり，累積相対度数は0.6%である．
なお，MPI_BUFFER_SIZE=5MB，HBM_SIZE=160MB，HBM_THRESHOLD=4KBであった．

- 累積相対度数5%以下の値まで最適化するのに必要な試行数は7～23と，必要試行数は全サンプル数512の4.5%以下であり，大変効率的に最適化される．

snappyHexMeshによる格子生成

blockMesh
構造格子メッシャー

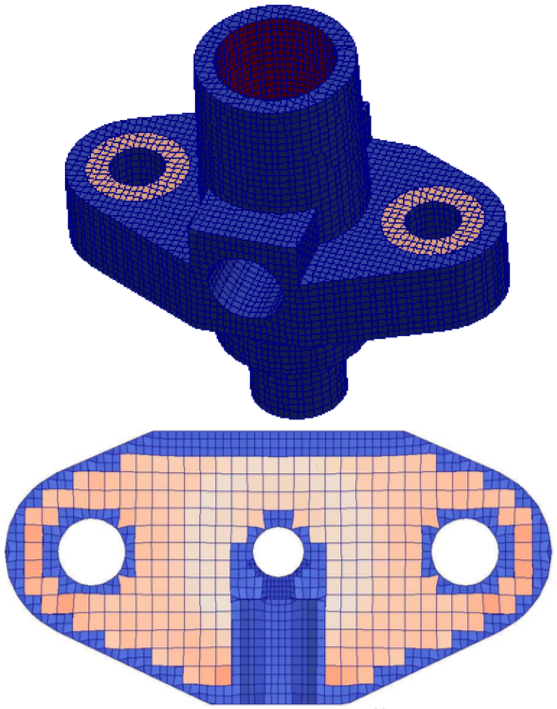
解析領域を含む構造格子のベース格子を生成



解析領域

snappyHexMesh
構造格子メッシャー

ベース格子を細分割して，形状に適合したメッシュを自動的に生成



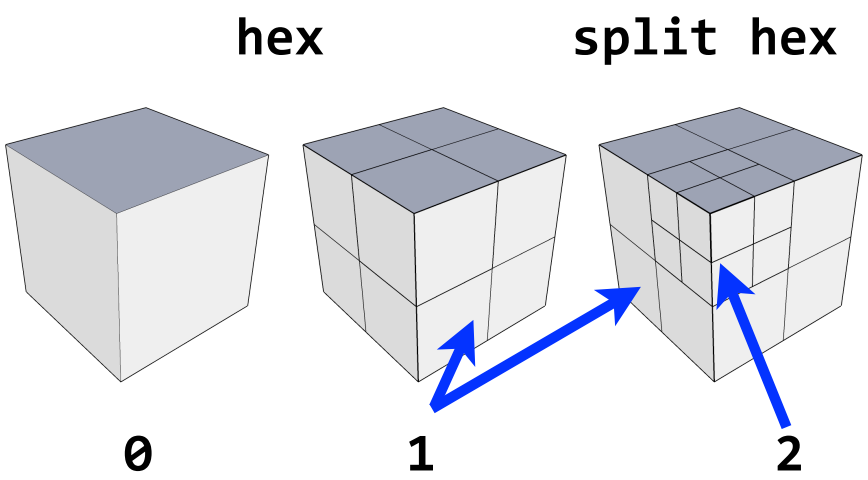
○ ほぼ六面体の格子が自動的に生成可能

× 任意間隔のベース格子が作成が難しい

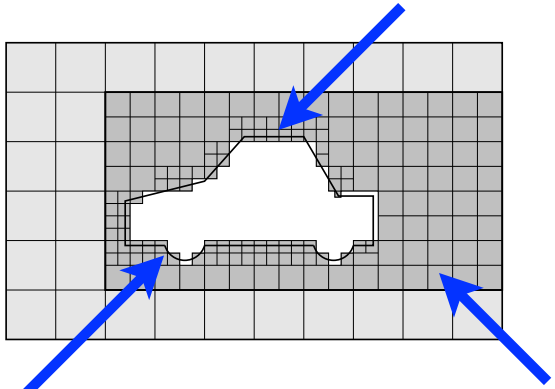
× 生成格子がベース格子の形状に大きく依存

snappyHexMeshの特徴

- 六面体格子(hex)または”面が分割された六面体的状格子”(split hex)からなる格子を自動生成する
- STL形式等の三角形分割曲面形状に適合した格子が生成できる(辞書で定義される直方体や球面等の基礎形状も利用可)
- 格子の細分割は8分木(2x2x2の分割を再帰的に行う)
- 曲面形状や基礎形状の表面の細分割レベルを指定できる(表面ベースの細分割)
- 細分割領域を曲面形状や基礎形状を用いて別途定義できる (領域ベースの再分割)

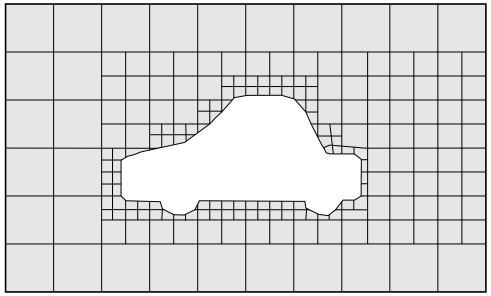


三角形分割曲面形状
(STL, wavefront OBJなど)

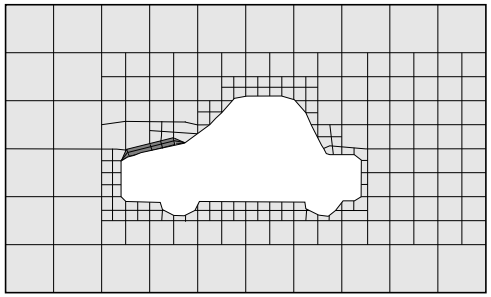


snappyHexMeshの特徴(続き)

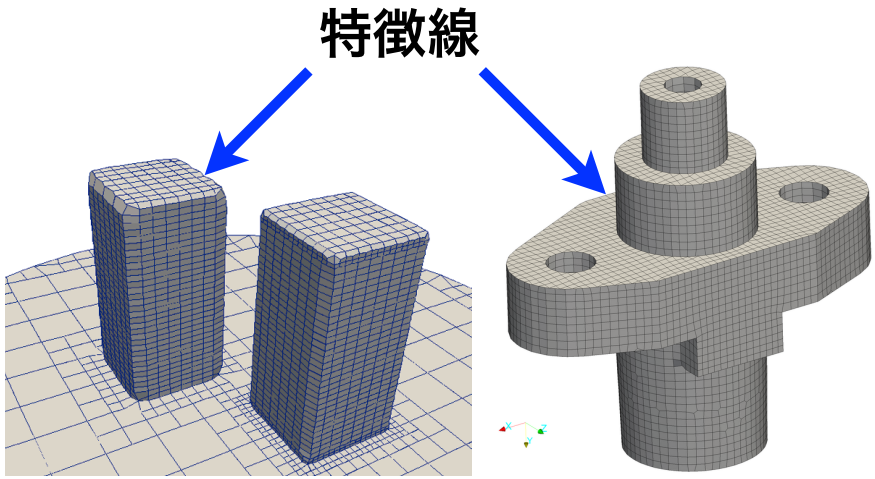
- STL表面や基礎形状の表面に境界適合するように格子を滑らかに移動させることができる
- STL表面や基礎形状の表面にレイヤを滑らかに挿入できる
- 分割領域毎の格子数をロードバランシングしながら，並列に格子生成ができる
- 境界適合における特徴辺の再現はVer. 2までは実装されておらず、特徴辺は丸くなったが，Ver. 2から特徴辺再現機能が実装され，形状再現性が高まった
- 自動格子生成が可能だが，設定が繁雑で，特にレイヤ挿入の設定・制御は困難



境界適合
(snapping)



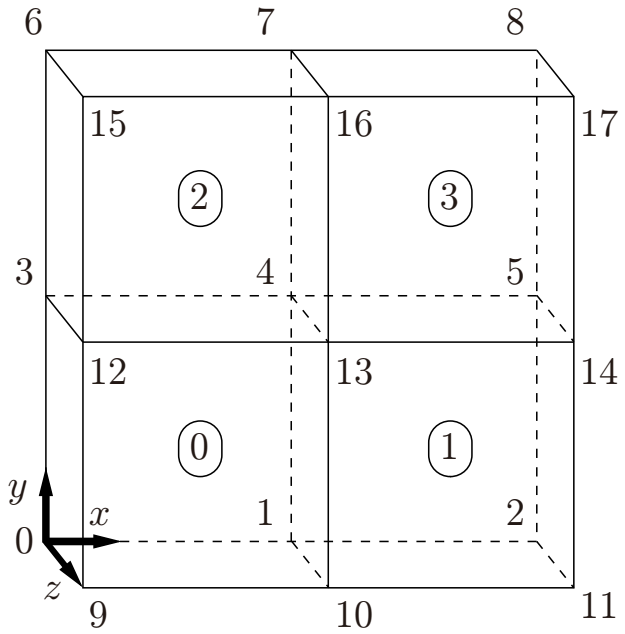
レイヤ挿入



Ver. 2未満　　Ver. 2以降

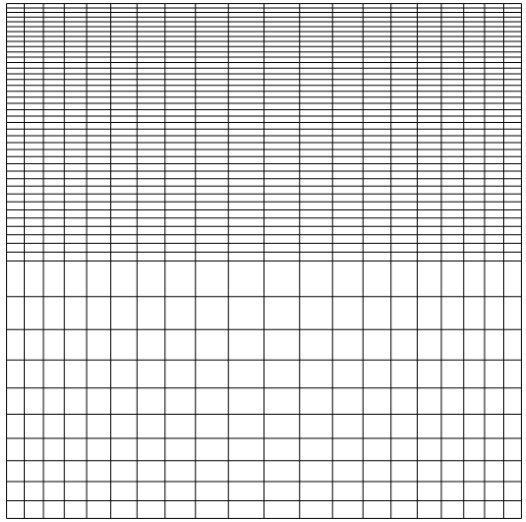
blockMesh

節点座標や分割方法を
`blockMeshDict`で定義



`blockMesh`

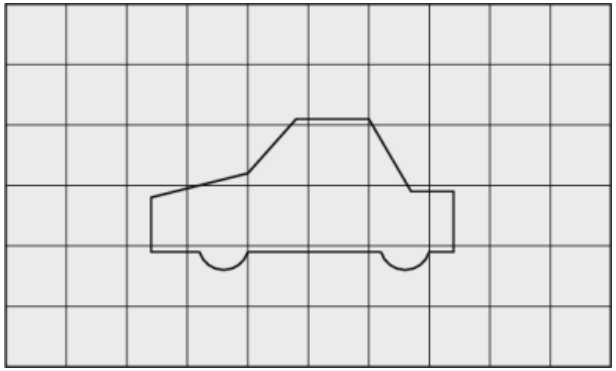
生成格子



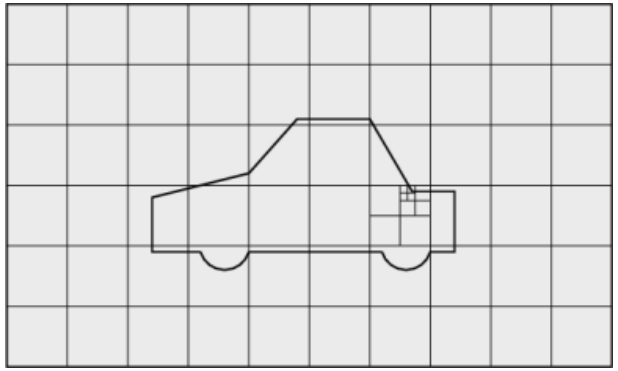
基礎的な六面体構造格子を生成する

(図引用元: OpenFOAM User Guide)

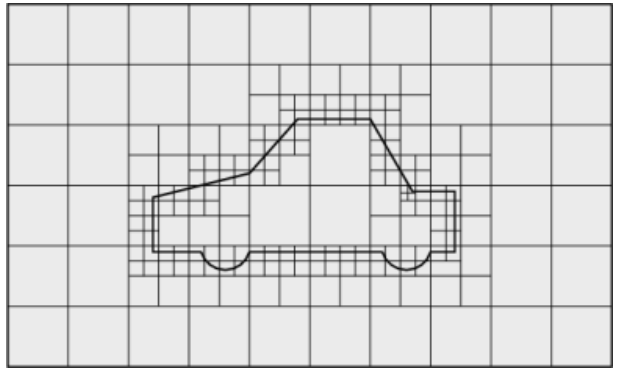
snappyHexMesh



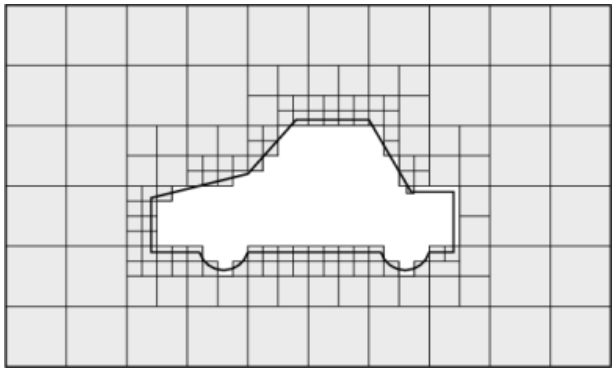
1. ベース格子



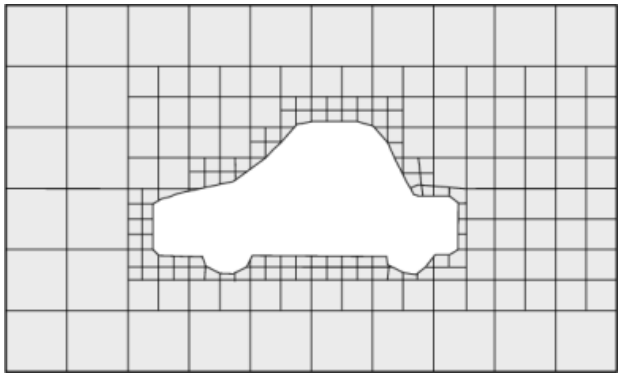
2. 物体表面の細分割



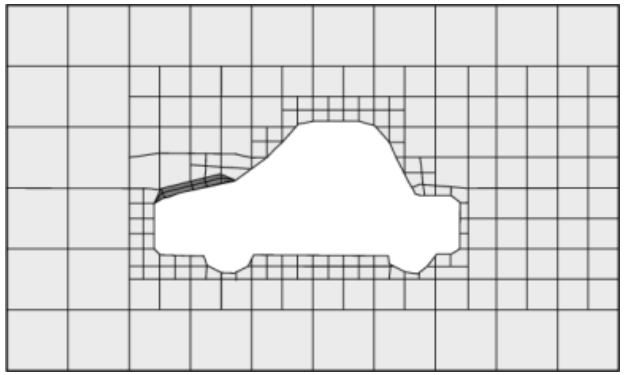
3. スムージング



4. 階段状格子



5. 境界適合(snap)

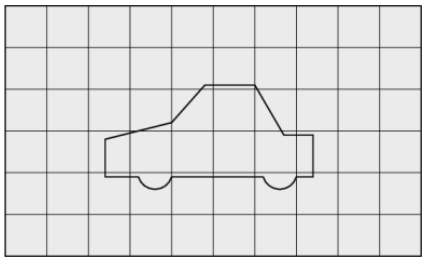


6. レイヤ付加

ベース六面体を元に複雑格子を生成

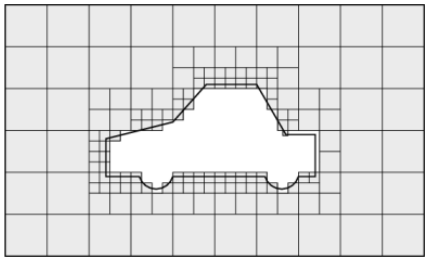
(図引用元: OpenFOAM User Guide)

格子の生成プロセス

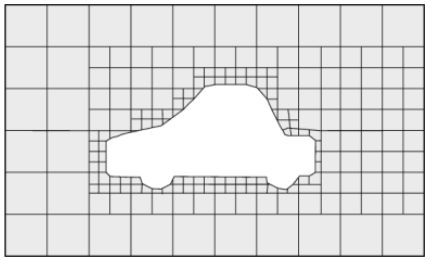


ベースの
六面体格子

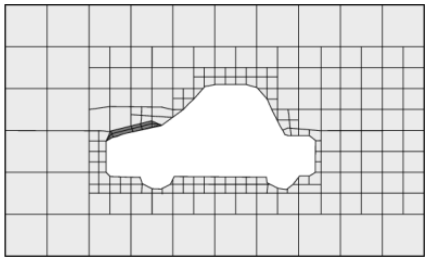
格子生成場所：
`constant/polyMesh`



階段状格子
`1/polyMesh`



境界適合
`2/polyMesh`



レイヤ付加
`3/polyMesh`

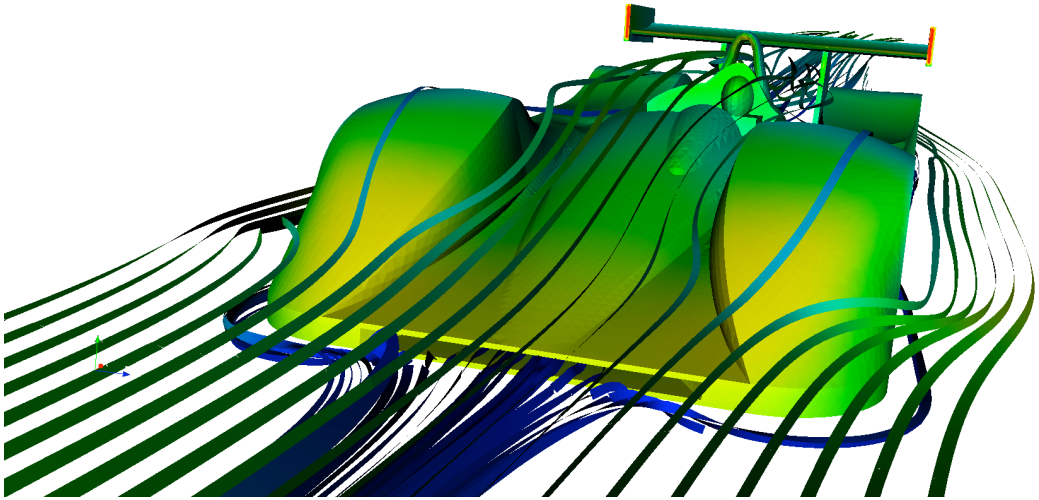
注)`overwrite`オプションを付けると
`constant/polyMesh`に上書き

`blockMesh` → `snappyHexMesh`

(図引用元: OpenFOAM User Guide)

ParaViewとは?

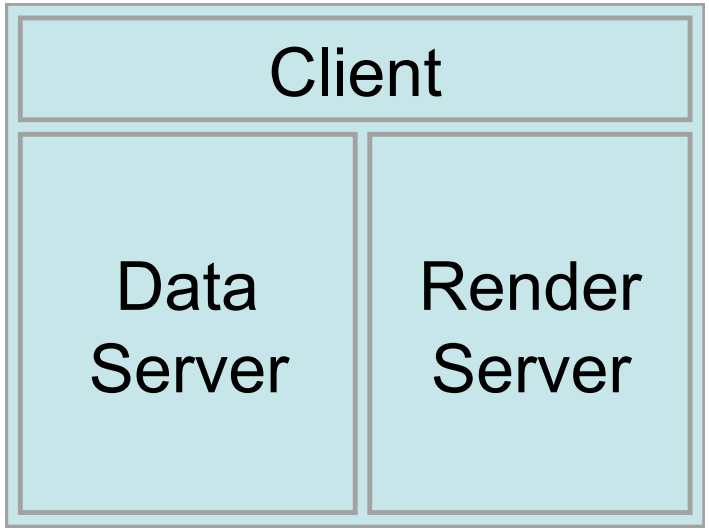
- オープンソース、スケーラブル、かつマルチプラットフォームな可視化アプリケーション
- 大容量データセットを処理するための分散型計算手法のサポート
- オープン、柔軟かつ直感的なユーザインターフェイス
- オープンな規格に基づいた拡張性の高いモジュール化構造
- 柔軟な3条項BSDライセンス
- 有償の保守およびサポート

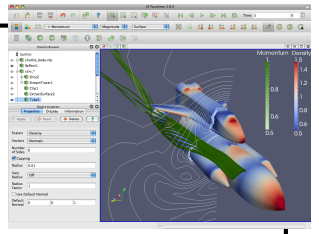


ル・マンのレースカー周りの気流
(ブラジル リオ・デ・ジャネイロ NACAD/COPPE/UFRJ Renato N. Elias)

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

ParaViewの3層構造





- クライアント
  - ✓ 可視化の作成を担当(GUI)
  - ✓ オブジェクトの作成、実行、削除を制御するが実際のデータは全く保持しない
  - ✓ 常にシリアル実行

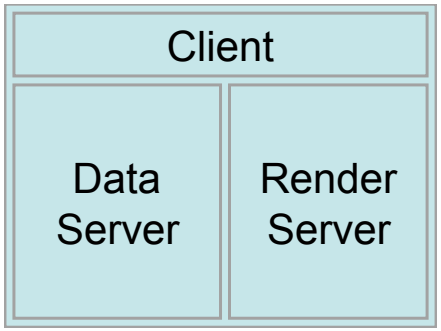
- データ・サーバ
  - ✓ データの読込み、フィルタリング、書出しを担当
  - ✓ 全てのパイプライン・オブジェクトはデータ・サーバが保持
  - ✓ 並列実行可能

- レンダー・サーバ
  - ✓ レンダリングを担当を担当
  - ✓ 並列実行可能 (内蔵の並列レンダリング機能が有効化)

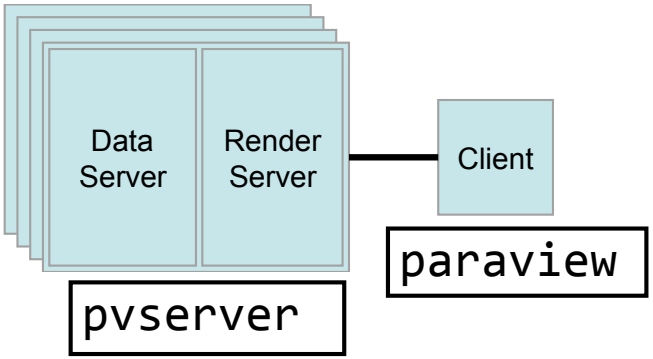
図出典：The ParaView Tutorial for Version 4.4
https://www.paraview.org/Wiki/The_ParaView_Tutorial

ParaViewの3層構造

コマンド名

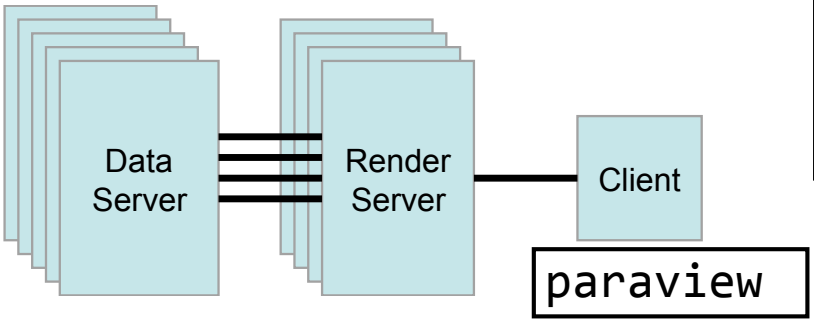


paraview



paraview

pvserver



paraview

pvdataserver

pvrenderserver

スタンドアローン・モード

- 全てが統合してシリアル動作

クライアント–サーバ・モード

- 並列可視化可能
- リモート可視化も可能

クライアント–レンダー・サーバー–データ・サーバ・モード

- サーバ間の通信量が多く，3つが独立動作する利点が少ないので，非推奨

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**

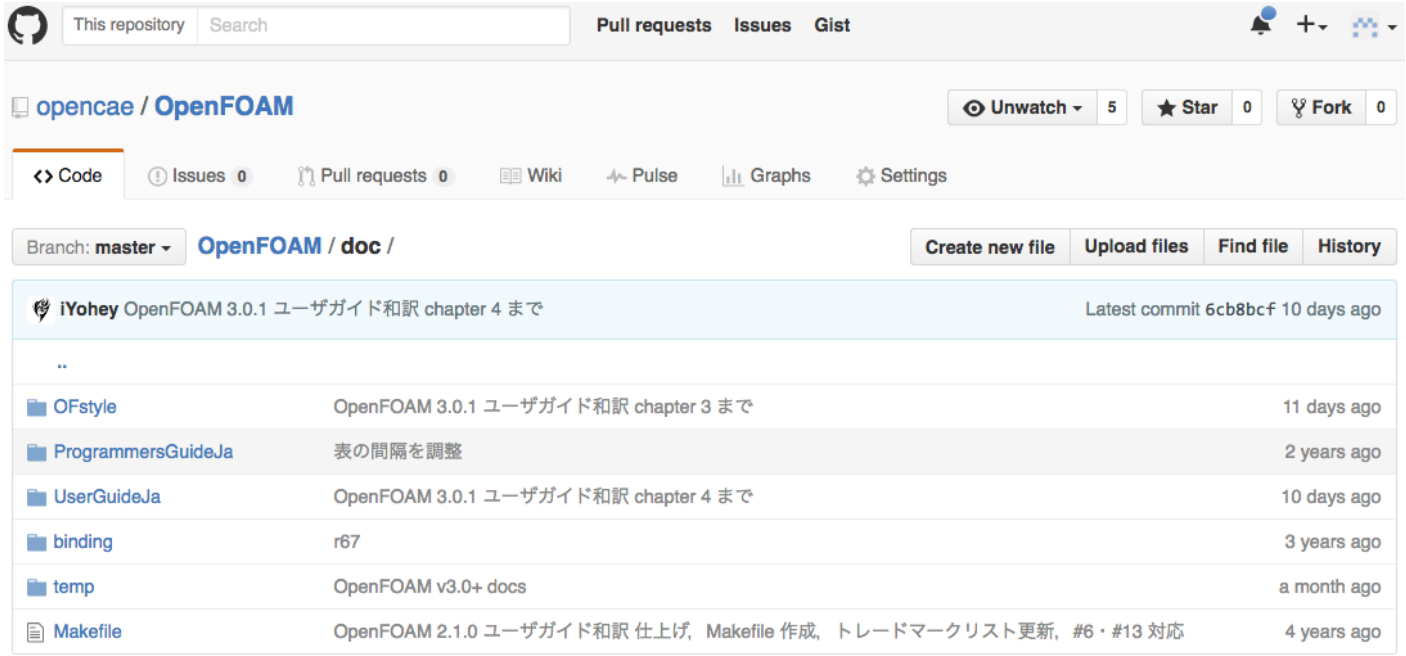
✓ **OpenFOAMデータ読み込み方法**

- **OpenFOAM附属のparaFoamコマンド使用 (非推奨)**
  - 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通常この方法
  - ログインノードでの起動は負荷が掛かり，かつX転送は遅いため，非推奨
- **通常のParaView(Ver.3.8以降)を使用**
  - ParaViewを起動するクライアント側にOpenFOAMの環境は不要
  - OpenFOAMのデータを読むには拡張子が.foamのダミーファイルが必要
    - **スタンドアローン・モード：**スパコン側の格子・解析結果をクライアント側に転送して，クライアント側のParaViewで可視化 **(今回はこの方法)**
    - **クライアント–サーバ・モード：**スパコン側で起動したpvserverと，クライアント側で起動したParaViewが通信して可視化

OpenFOAMユーザーガイド和訳

- 和訳のLaTeXソースや図など全てオープンCAE学会のレポジトリで公開
- オープンCAE学会による和訳はOpenFOAM FoundationのWEBに掲載
- 現在，3.0.1版まで公開．学会で最新plus版の和訳を計画中

https://gitlab.com/OpenCAE/OpenFOAM

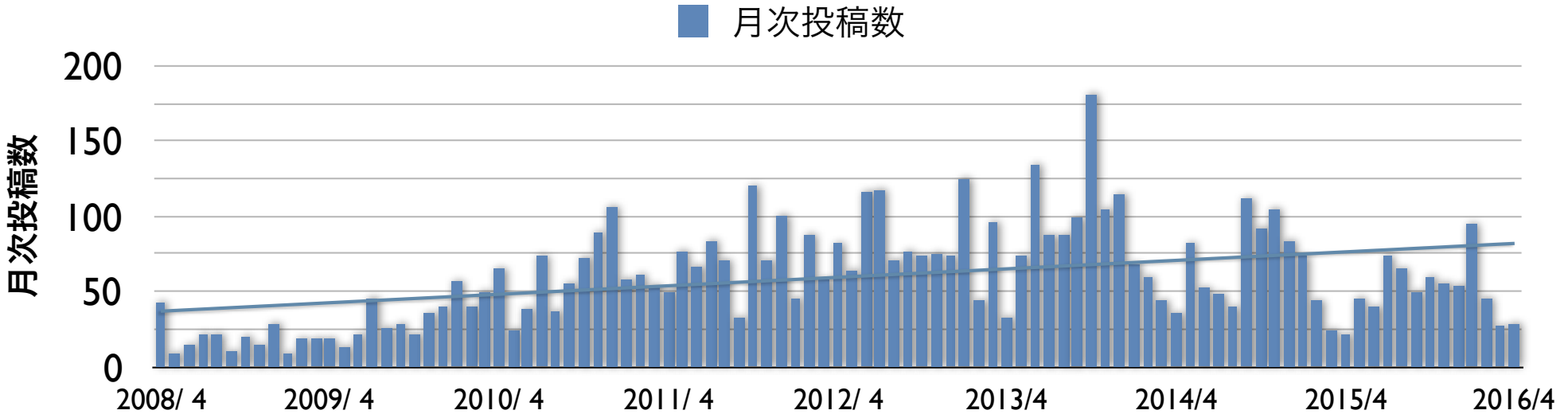


日本版OpenFOAM掲示板

OpenFOAM Googleグループ

- 設立：2008年3月
- 登録者：959名 (2019/1/7時点)
- 質問，情報交換，イベント告知
- 匿名で質問可能
- 初心者の駆け込み寺

第４７回オープンＣＡＥ勉強会@岐阜のご案内 （1）	1 件の投稿	5月9日
OpenFOAM-v3.0+のバイナリインストールについて （4）	4	5月6日
チュートリアルcircuitBoardCoolingの設定について （3）	3	5月6日
オープンCAE講習会のご案内 （1）	1	5月5日
移動メッシュの破たんについて （3）	3	5月3日
【関西】(4/24) 第48回オープンCAE勉強会@関西 開催のご案内 …	3	4月23日
interDymFoamでの質問 （3）	3	4月23日
convectiveOutletの対流速度につきまして （2）	2	4月20日
第30回オープンCAE勉強会@広島 4月16日開催 （2）	2	4月10日
メッシュの結合について（enGridとblockMesh） （12）	12	4月9日
【勉強会@富山】4月16日(土)オープンCEA勉強会@富山(第42回)開…	1	4月8日



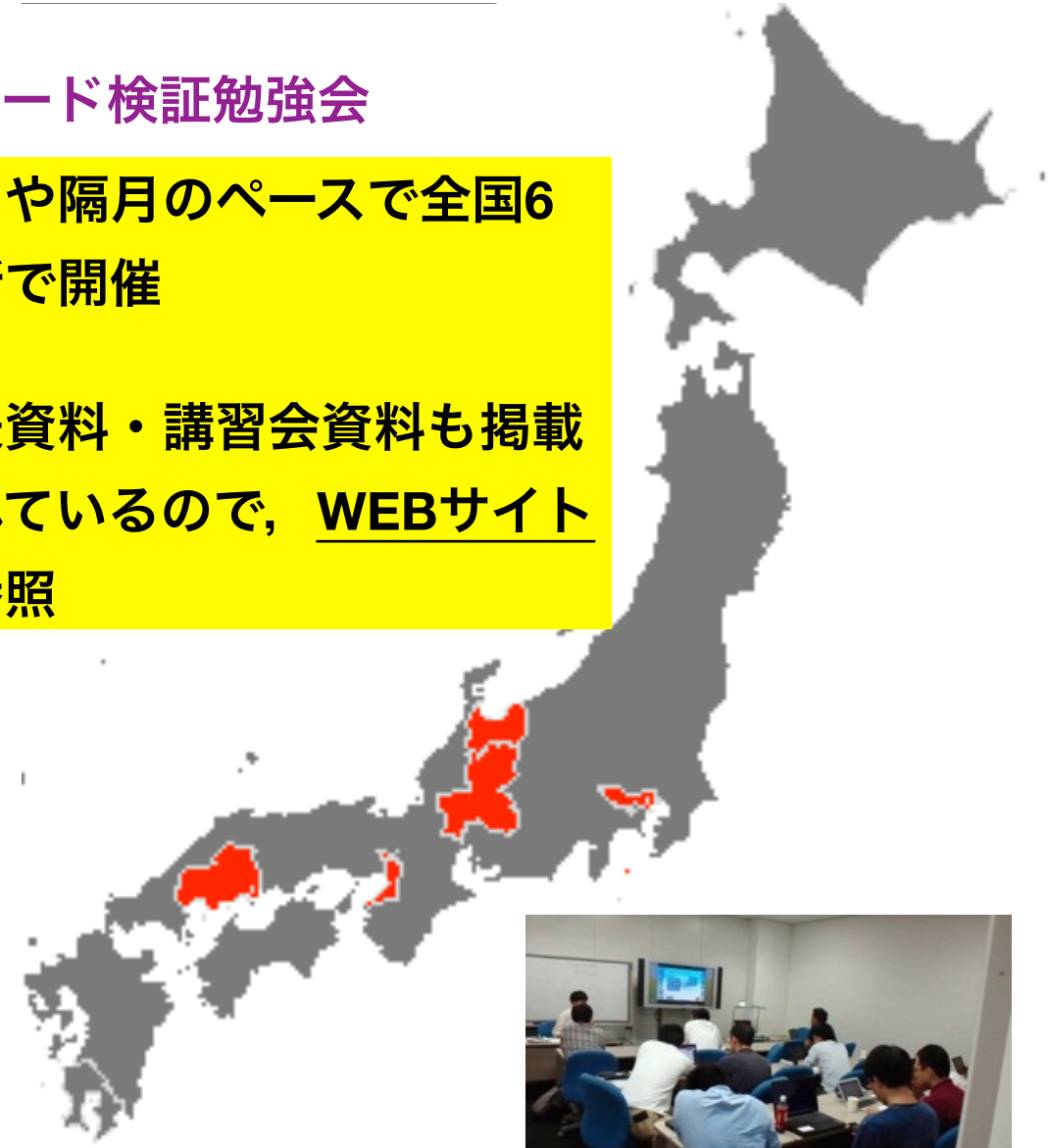
https://groups.google.com/forum/#!forum/openfoam

オープンCAE勉強会

1. @関東(流体など)(2010年6月～)
   Ustream配信(初期)，OpenFOAMコード検証勉強会
2. @関西(2010年12月～)
   自作風洞実験，ブックレビュー
3. @岐阜(2011年1月～)
   初心者のみ質問会・夏合宿
4. @富山(2012年5月～)
   ミニ講習会・鱒寿司制覇懇親会
5. @広島(2012年7月～)
   ミニ講習会
6. @関東(構造など)(2014年10月～)
   構造解析に特化した勉強会
7. 合同勉強会

毎月や隔月のペースで全国6箇所で開催

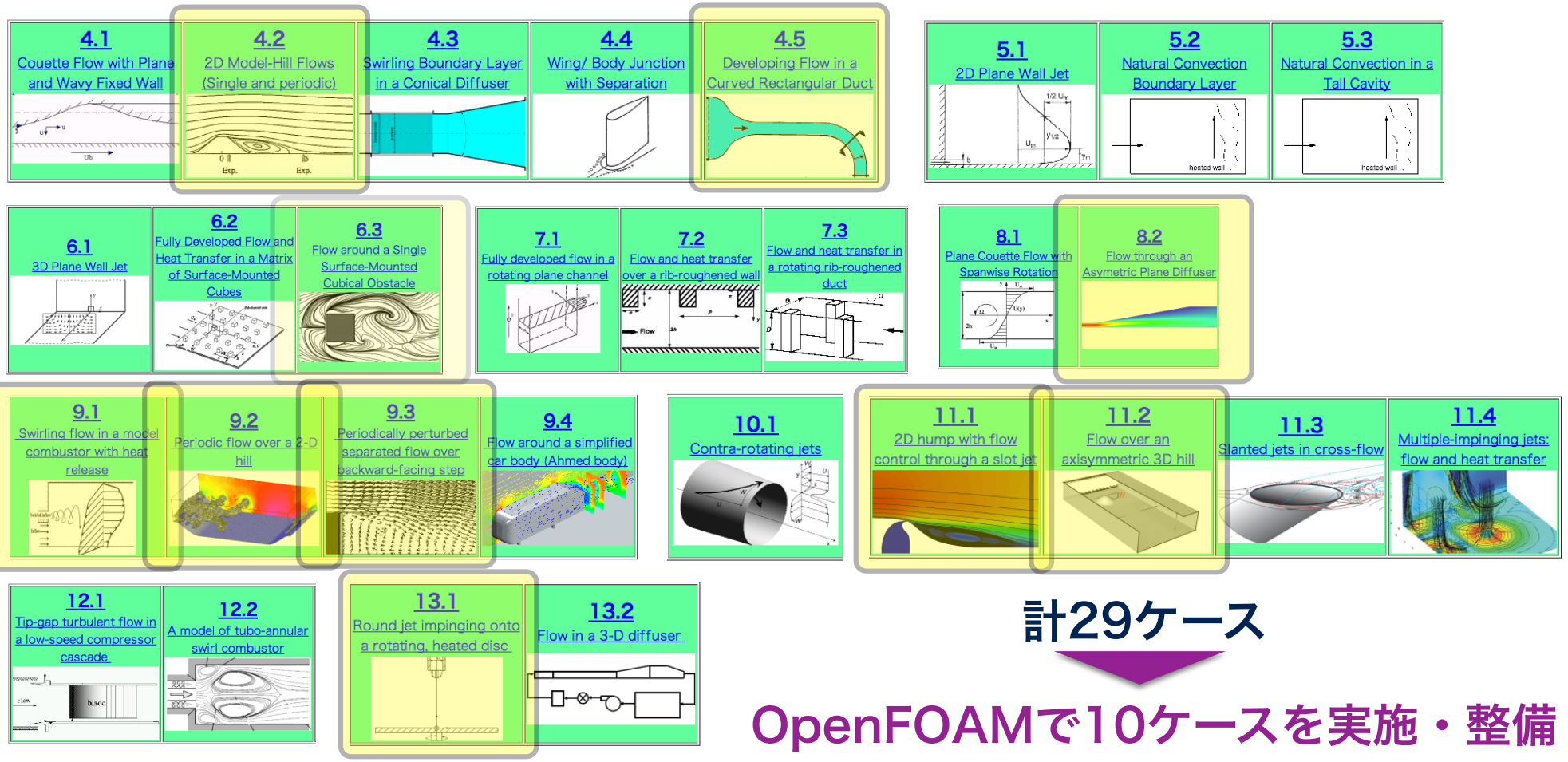
発表資料・講習会資料も掲載されているので，WEBサイトを参照



ERCOFTAC SIG15ベンチマーク

オープンCAE学会のレポジトリで実験値との検証ケースを公開．実験値との比較も可能

ERCOFTAC(European Research Community on Flow, Turbulence And Combustion)内の乱流モデリンググループのワークショップ('95-'01)で実施したベンチマーク



計29ケース

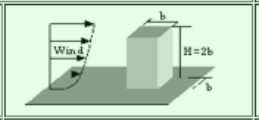
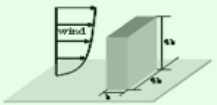
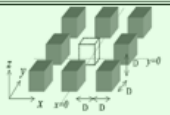
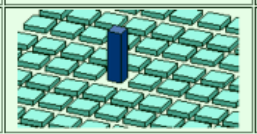
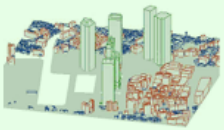
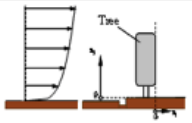
OpenFOAMで10ケースを実施・整備

市街地風環境ベンチマーク

日本建築学会 流体数値計算による風環境評価ガイドライン作成WG が整備

書籍

WEBサイト(実験データや標準計算条件)

	test case		dataset	Ref.
A	2:1:1 shape building model		Data file :CaseA(1_1_2).xls	[1]
B	4:4:1 shape building model		Data file : CaseB(4_4_1).xls	[2][3]
C	Simple Building blocks		Data file : CaseC(City_blocks).xls	-
D	A high-rise building in city blocks		Data file : CaseD(Highrise+Blocks).xls CAD File(DXF) : CaseD_dxf.zip CAD File(MCD) : CaseD_mcd.zip	[5]
E	Building complexes with simple building shape in actual urban area (Niigata)		Data file : CaseE(Niigata).xls CAD File(DXF) :CaseE_dxf.zip CAD File(MCD) : CaseE_mcd.zip	[6]
F	Building complexes with complicated building shape in actual urban area (Shinjuku)		Data file : CaseF(Shinjuku).xls CAD File(DXF) :CaseF_dxf.zip CAD File(MCD) : CaseF_mcd.zip CAD File(STL) : CaseF_stl.zip	[6]
G	Two-dimensional pine tree		Data file : CaseG(Tree).xls	[7]

OpenFOAMで6ケースを実施・整備

工学ナビ

東京大学生産技術研究所・革新的シミュレーション研究センター(センター長：加藤千幸先生)が制作運営しているWEBサイト

- 複合材料強度信頼性評価 (10)
  - FrontCOMP (10)
- 音響解析 (2)
  - FFB-Acoustics (2)

ナノテクノロジー (65)

- 第一原理電子状態解析 (65)
  - PHASE (65)

ライフサイエンス (36)

- フラグメント分子軌道法 (16)
  - ABINIT-MP (16)
- 標準密度汎関数法 (20)
  - ProteinDF (20)

- OpenFOAM (16)
  - OpenFOAM Version 2.2.2 (16)
- 最適化設計 (3)
  - CHEETAH (3)
- 構造解析 (37)
  - ADVENTURECluster Solver (4)
  - Front ISTR (31)
- 流体構造連成解析 (1)
- 流体解析 (54)
  - FFR (4)
  - FFV (16)
  - FrontFlow/blue (28)
  - GKV (2)
  - GT5D (2)
  - LANS3D (3)

解析格子を以下に示す．計算格子生成にはOpenFOAM付属の自動格子生成ユーティリティsnappyHexMeshを使用し，学会提供のCADデータをSTL形式に変換したデータを使用して約530万要素の格子を自動生成させた．

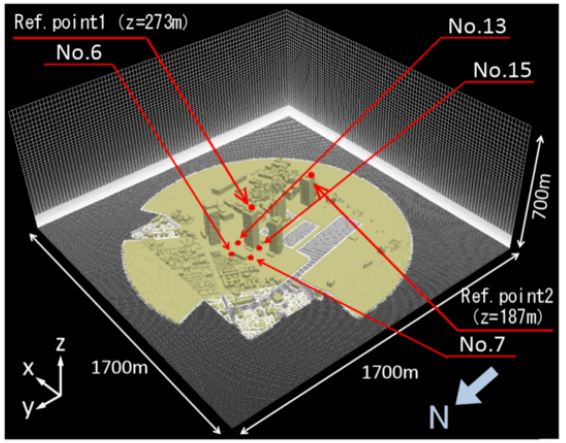
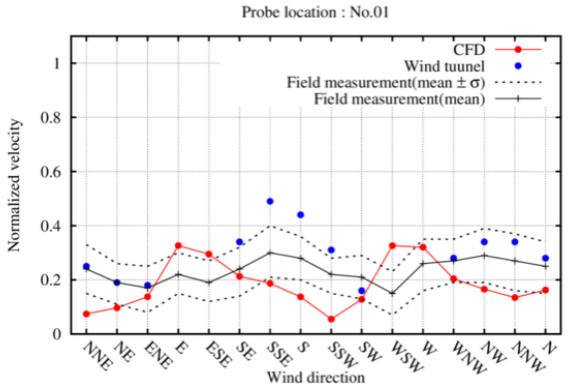


Fig. Calculation mesh (参考文献[Imano2010]から引用)

計算結果

各計測点における風向別の風速比を以下の図に示す．風洞実験値及び実測値[AIJ]も合わせて示す．何れの結果も，風向NE～N～NWでは参照点1(D)，それ以外の風向は参照点2(C)の風速で基準化してある．



例：新宿副都心高層ビル群の風環境

OpenFOAM書籍

- 発売日：2016年6月17日
- 価格：3,456円(税込)
- 編集：オープンCAE学会
- 出版社：森北出版
- 日本初のOpenFOAM本
- 書籍名はスハス V.パタンカー(森北出版)の「コンピュータによる熱移動と流れの数値解析」のオマージュ
- 第4刷はOpenFOAM-2.4.0, 3.0.x, 4.x, 5.0, 6に対応した脚注あり

OpenFOAMの主な国内会議

- **オープンCAEシンポジウム(2010年～，参加約80～150名)**
  - ✓ 主催: オープンCAE学会
  - ✓ 2～3並列×3～4コマ程度のトレーニング(構造解析・可視化・1DCAEも含む)
  - ✓ OF以外の流体解析ツールや構造解析・可視化関連の発表も有る．資料WEB公開
  - ✓ **現状OpenFOAM関連の発表件数は日本最大規模．年々増加**
- **OpenFOAM・CAEワークショップ (2013年～，参加約50～100名[参加費無料])**
  - ✓ 主催: 高度情報科学技術研究機構(RIST)
  - ✓ HPCI課題による「京」でのOFの事例．RISTによるOFのチューニング．富士通によるコンパイラの最適化などHPC向けの発表．資料WEB公開
- **ソフトウェアベンダーのユーザ会でのOpenFOAMセッション**
- **オープンCAE学会以外の学会でのOpenFOAMに関する研究発表**

学会，ユーザ会，HPC系WSなどで，年々OpenFOAMの発表が増加している

OpenFOAMの主な国際会議

- **OpenFOAM Workshop(2006年～，参加約140～400名)**
  - ✓ 主催: OpenFOAM Workshop committee
  - ✓ **世界最大規模(ドイツ開催時は参加者約400人)**
  - ✓ 3並列×4コマ程度のトレーニング(参加費に含まれる)
  - ✓ Hrvoje Jasakによる基調講演(開発方針など)
  - ✓ OFのカスタマイズ例・適用例など発表多数．講習会資料も公開
- **OpenFOAM User Conference (2013年～，初回参加222名)**
  - ✓ 主催: ESI社
  - ✓ 基調講演: OF開発にファンドしているVolkswagen等
  - ✓ 開発メンバーによる今後の開発方針，ユーザ適用事例等

多くの発表資料や講習会資料が公開されているので，
興味がある分野の資料を検索してみてください

関連Webサイト

[OFF] The OpenFOAM Foundation (http://www.openfoam.org/)
[OFC] OpenCFD Ltd (ESI Group)(http://www.openfoam.com/)
[OCSJ] オープンCAE学会(http://www.opencae.or.jp/) OpenFOAMユーザガイド和訳, プログラマズガイド和訳, The ParaView Tutorial和訳, 過去のシンポジウム・ワークショップ・講習会資料
[XSIM] **XSim** OpenFOAM 付属チュートリアル一覧
(https://www.xsim.info/articles/OpenFOAM/Tutorials.html)
[OFT] 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/)
[PENGUINITIS] (http://penguinitis.g1.xrea.com/) 圧倒的な情報量
[FN365] (http://caefn.com/) 多くのOpenFOAMに関するスライド(日本語・英語)も公開
[installOpenFOAM] OpenFOAM自動ビルドスクリプト**(** https://gitlab.com/OpenCAE/installOpenFOAM/blob/master/README.md) スパコンでOpenFOAMを自動でビルドする