AMAZON EC2 GPUクラウドによるOPENFOAM流体解析の性能評価

西條晶彦, 井口寧, 松澤照男
北陸先端科学技術大学院大学

背景

- クラウド計算機によるHPC資源の利用
  - 払ったぶんだけの計算機資源 (pay-as-you-go) を確保
  - 大学・研究機関等に所属していなくても誰でもHPCが使える
- Webサイトやデータ解析の分野では使用例がある
- 科学技術計算の分野における使用例は未だ少ない

研究目的

- クラウドHPCであるAmazon EC2サービスにおいて流体解析計算の性能測定

- 流体ソルバOpenFOAMのGPU対応版ソルバを開発EC2上での高速化

手法

- クラウドHPCはMPI通信性能が非常に低い
- MPI数の増加は通信コスト増大，Courant数の増加，コスト増
- 1ノードあたりの性能を高めるためのGPGPUとMPIを組み合わせたソルバの開発が必要

AMAZON EC2 GPU CLUSTER COMPUTE INSTANCE

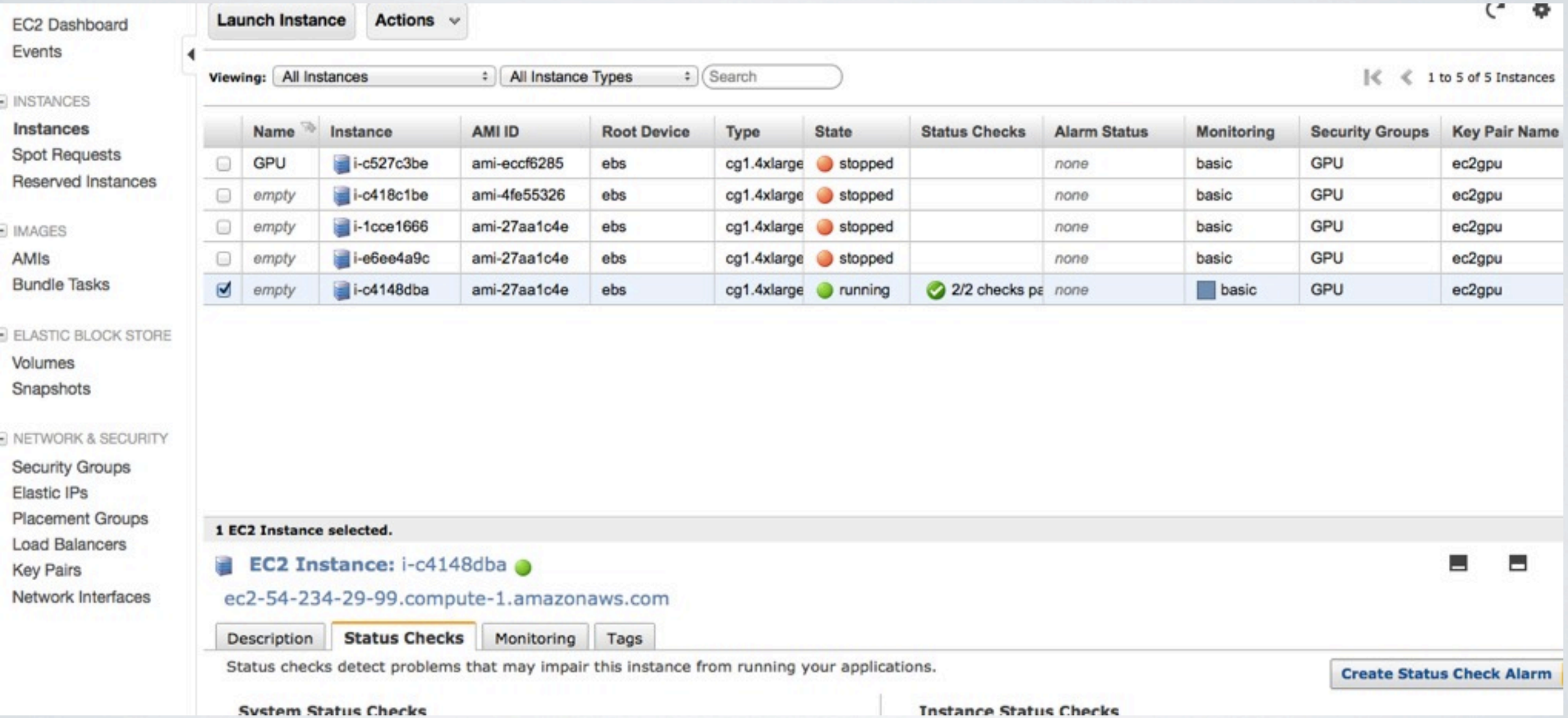
- EC2サービスによって提供されるGPUクラウドマシン (cg1.4xlarge)をオンデマンド契約形態で利用 (米東海岸N.Virgina地域)

- Quadcore Intel Xeon 5570 2.93 Ghz x2 (8cores)
- 22GB Memory
- NVIDIA M2050 (2687MB) x 2
- 10 GbEtherNet
- Amazon Linux AMI 2012.03 (RHEL base)
- $2.10 /hour / node

EC2計算環境構築

- Youtubeの公式チュートリアル動画を参考:“Building a Cluster in Less Than Ten Minutes”
- 見かけは普通の遠隔マシン，sudo権限があるので自由度が高い
- CUDA SDK, OpenFOAM, GPUソルバ, 解析ケースを持つMachine Imageを作成，これをWebコンソールからデプロイする
- 今回の計算ではインスタンスの共有ドライブは作らず，それぞれのインスタンスが自分自身のルートノードをストレージとする．並列計算の場合は分割ケースを全てコピー
  - $ 0.10 / GB / month

EC2 WEB CONSOLE



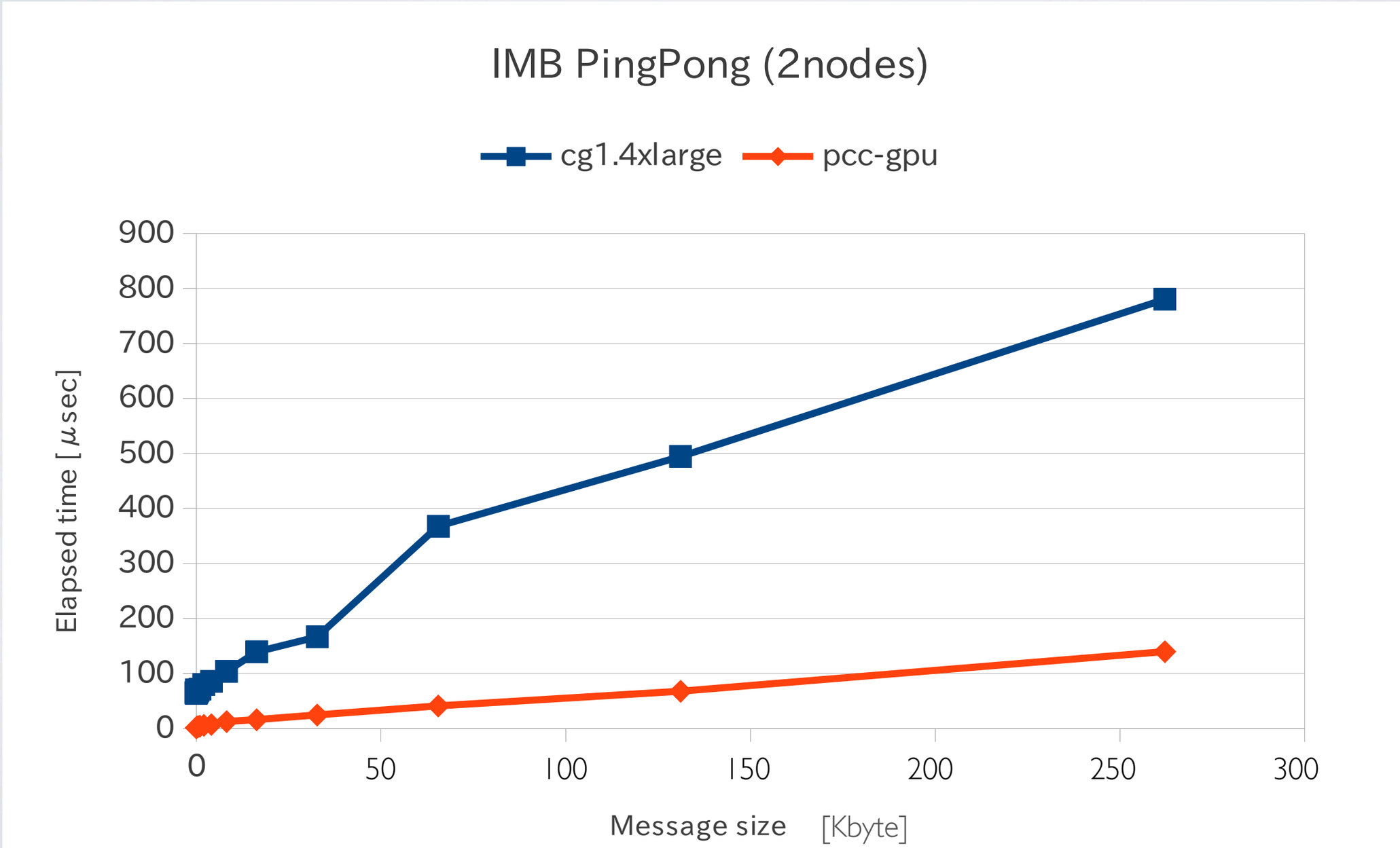
PCC-GPU: APPRO GPU CLUSTER

- 比較として北陸先端大情報センターから提供される内部の(in-house)型のGPUクラスタ(pcc-gpu)を用いる
  - Octocore AMD Opteron 6136 @ 2.4 GHz x 2 (16 cores)
  - 32 GB Memory
  - NVIDIA M2050 (2687MB) x 2
  - Infiniband QDR
  - CentOS 6.2
- 8 ノード使用 (最大9ノード)

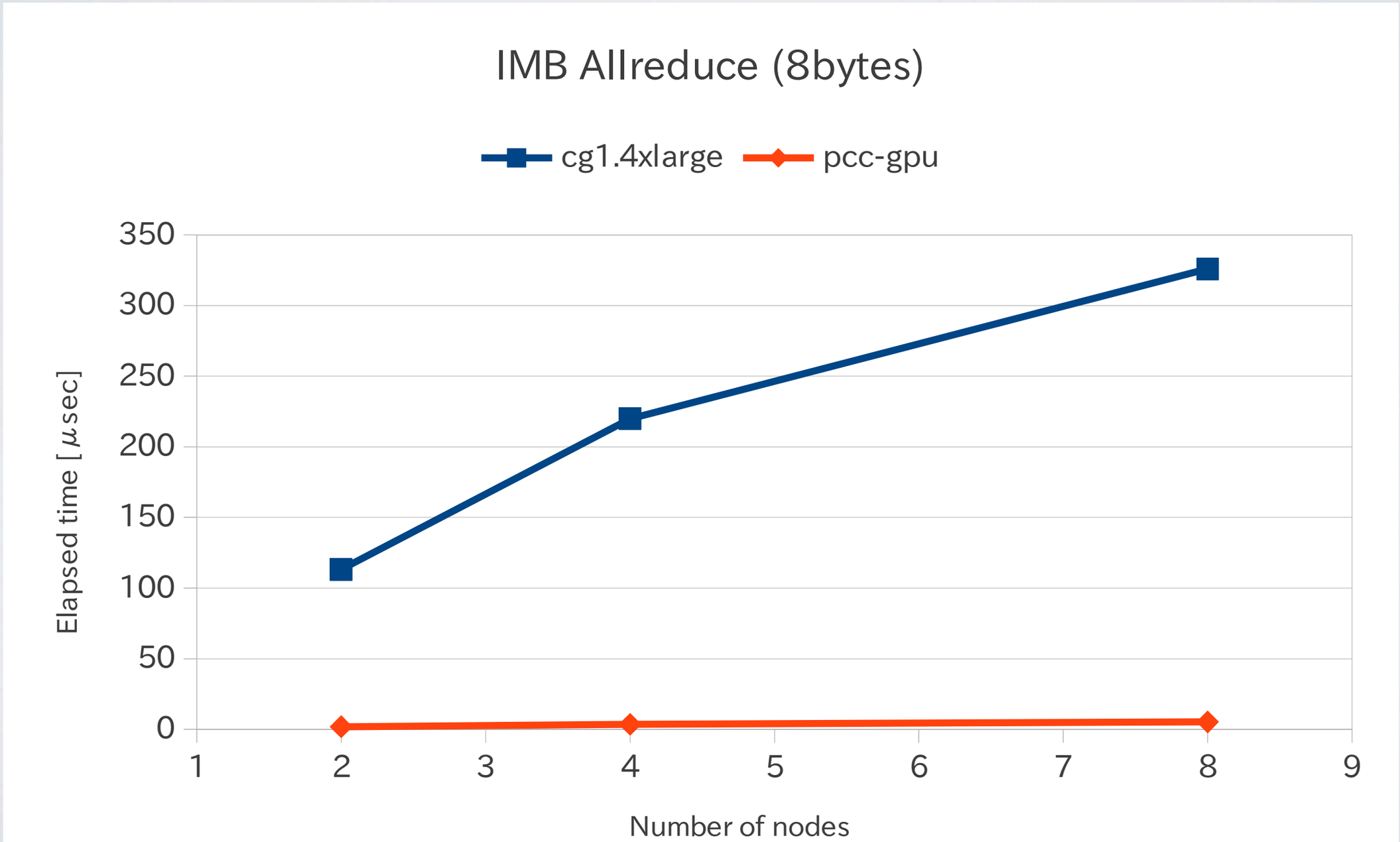
通信性能ベンチマーク

- Intel MPI Benchmarks (IMB) 3.2.3
- OpenFOAMで利用する2つの通信方式のベンチマークを同一ワークグループ内のノード間で行う
  - 双方向通信: PingPong 2ノード間メッセージサイズに対する経過時間
  - 縮約演算通信: Allreduce (MPI_SUM, 8bytes) ノード数に対する経過時間

IMB: PINGPONG (2 NODES)



IMB: ALLREDUCE (SUM, 8BYTES)



定常NS方程式

$$\begin{cases} \nabla \cdot (\rho \mathbf{U}) = 0, \\ (\mathbf{U} \cdot \nabla)\,\mathbf{U} - \nabla \cdot (\nu \nabla \mathbf{U}) = -\nabla P \end{cases}$$

$$\mathbf{U}_f = \left(\frac{H(\mathbf{U})}{a_p} \right)_f - \frac{(\nabla P)_f}{(a_p)_f}$$

$$\nabla \cdot \left(\frac{1}{a_p} \nabla P \right) = \nabla \cdot \left(\frac{H(\mathbf{U})}{a_p} \right) = \sum_f \mathbf{S} \left(\frac{H(\mathbf{U})}{a_p} \right)_f$$

- 定常NS方程式の運動量方程式をセルpとその界面fで差分化し差分化係数apと圧力勾配∇Pと中間的な速度場 Ufの関係を導出

- 連続の式の離散化を代入して，圧力のPoisson方程式を解く

SIMPLE法

Algorithm 1 SIMPLE 法

1: 境界条件の設定
2: **repeat**
3: 離散化された運動量方程式を解いて中間的な速度場を算出
4: セル界面の質量流束の計算
5: PCG 法で圧力方程式を解き，不足緩和を適用
6: セル界面での質量流束を修正
7: 新しい圧力場から速度場を修正
8: 境界条件の更新
9: **until** 圧力場と速度場が閾値以下に収束するまで

PRECONDITIONED CG法

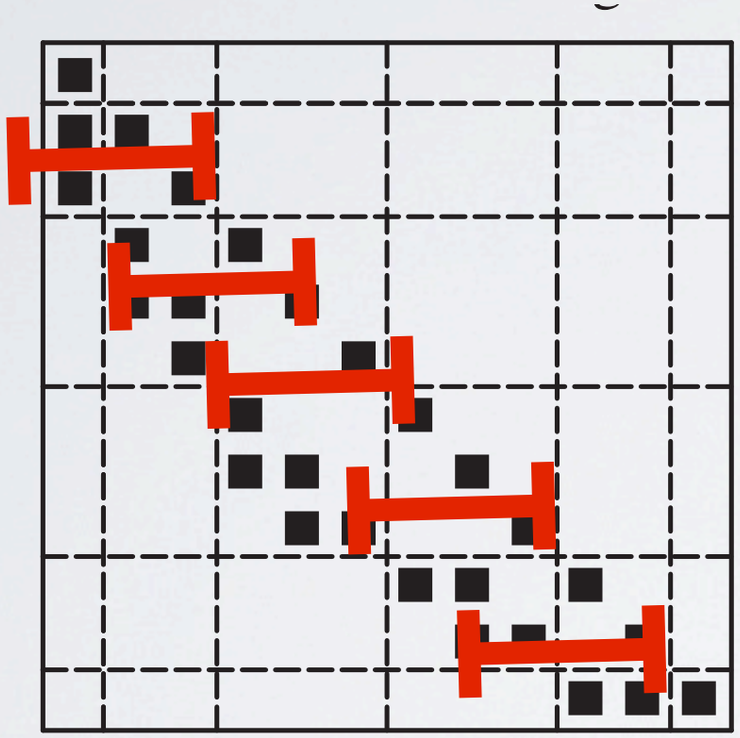
- 3回のMPI縮約通信 - CUBLAS
- SpMV (sparse Matrix Vector) - CUDA ITSOL (Li and Saad, 2012) JAD形式
  - MPI双方向通信: 袖領域の交換
- 前処理 - CUDA ITSOL, NVIDIA CUSP

GPU線形ソルバ

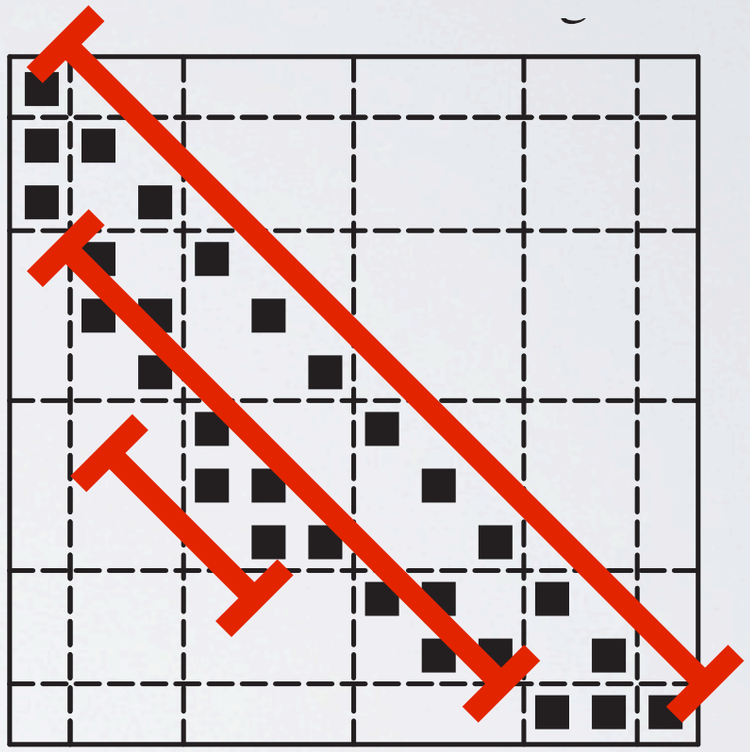
- CUDA ITSOL (Li and Saad, 2011)
  - CUDA用の線形ソルバパッケージ
    - JAD形式 SpMV (Sparse Matrix Vector product)
    - GPU用の様々な前処理
- NVIDIA CUSP: 線形ソルバパッケージ AMG前処理
- これを用いてMPI通信を統合したOpenFOAM用線形ソルバを開発

JAD: SPARSE MATRIX STORAGE

Compressed Row Storage

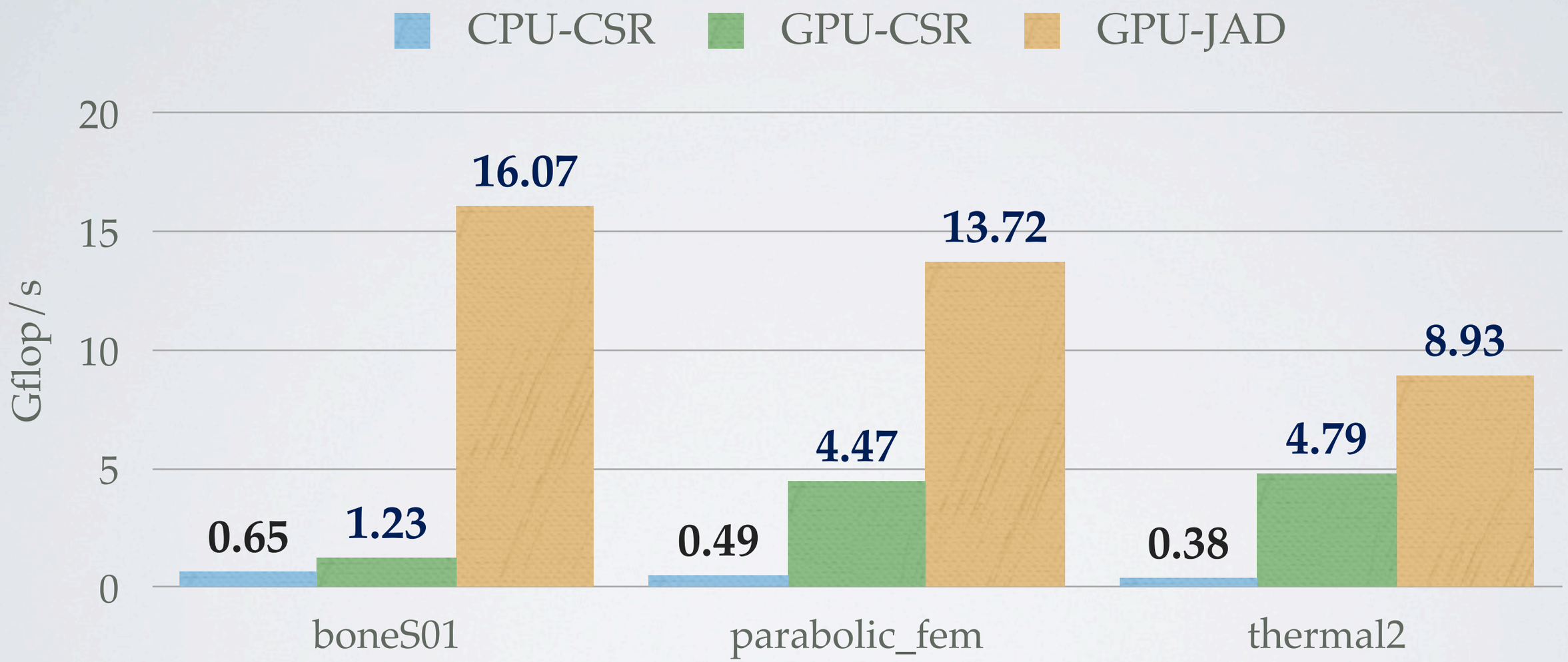


JAgged Diagonal storage



- JAD : 疎行列を対角線方向に取り行方向からのCUDAスレッドで計算していく

JAD SPMVパフォーマンス



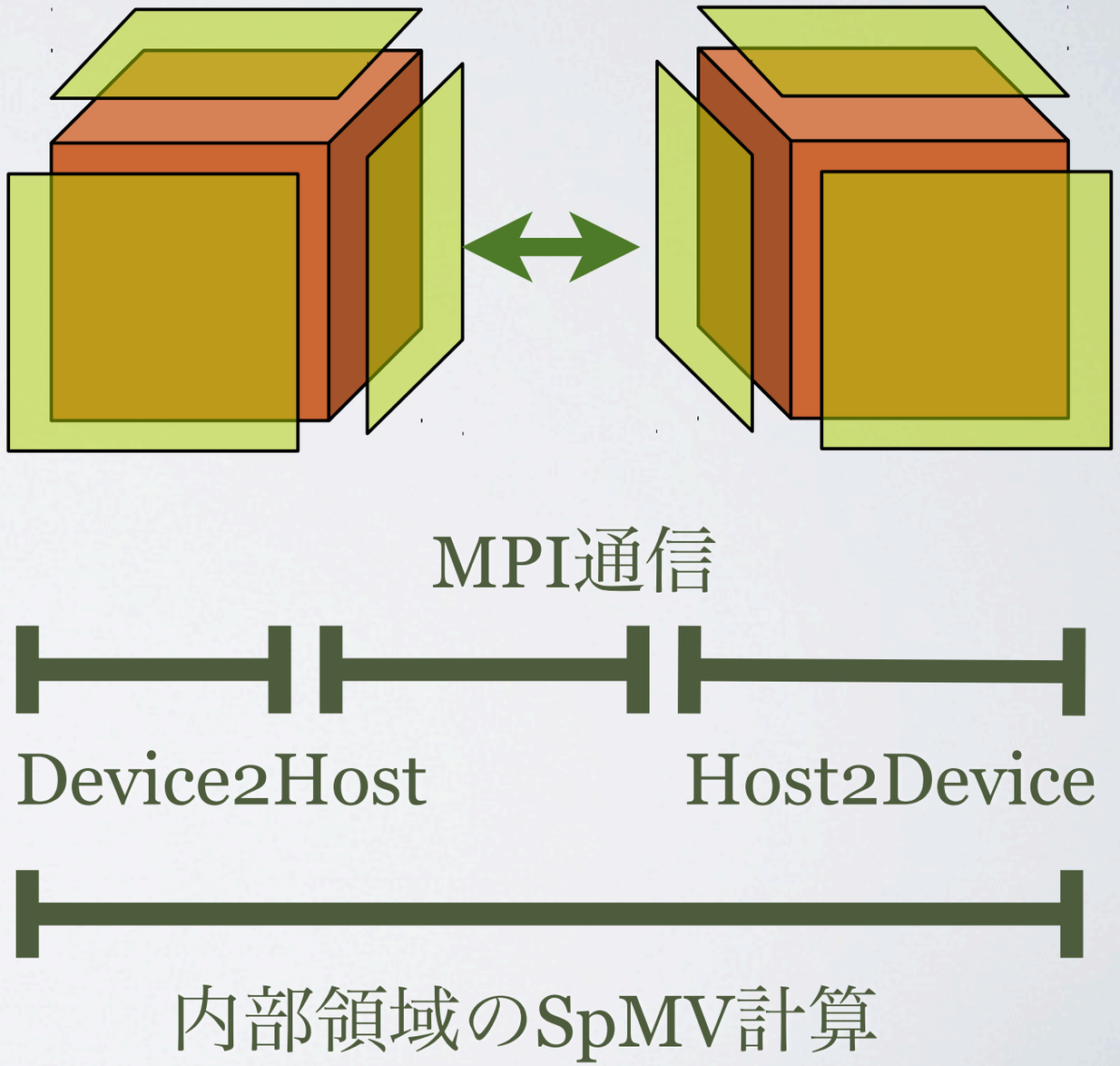
行数	127,224	525,825	1,228,045
非零要素	6,715,152	3,674,625	8,580,313

データ変換のキャッシュ

- 行列のOpenFOAM形式からGPU用JAD形式への変換コスト高
- JAD形式は行列の非ゼロ要素の位置で決まる
- 最初の1回のループの際に並び替え順を記憶
- 以降のループでは行列値のみを転送し，並び替え

SPMV: MPI双方向通信

- 通信が行われる袖領域(Ghost cell)の配列番号を取得
- 袖領域のデータをCPUホスト側に引き上げ(D2H)，MPI通信，GPUデバイス側に戻す(H2D)
- 通信とSpMVはCUDAスレッドで同時に行なわれる



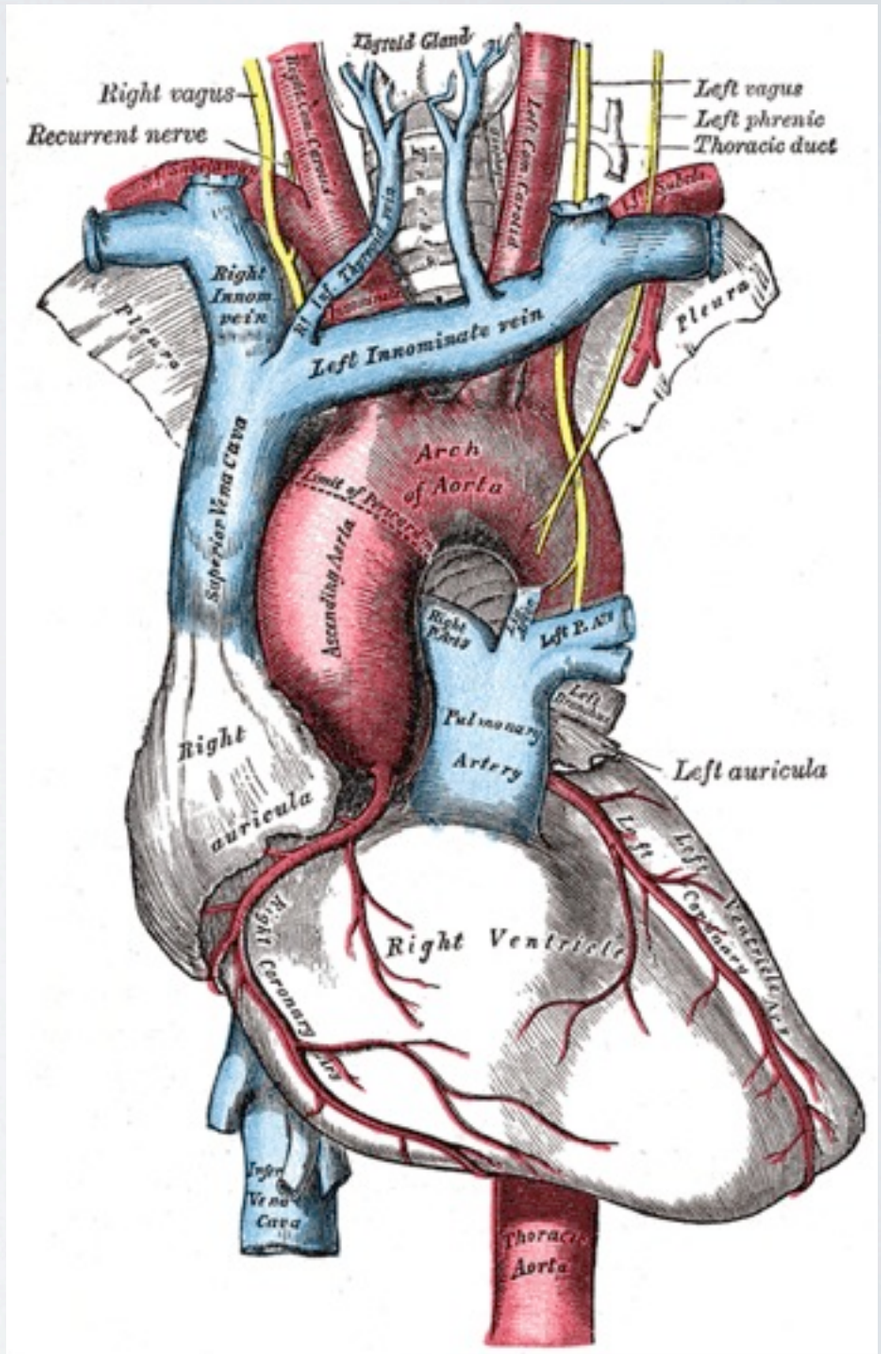
AMG前処理

- 代数的マルチグリッド前処理 (Algebraic MultiGrid Preconditioner)
  - 対象行列に対して粗い行列をいくつか生成し，粗行列での解を元の密行列に還元して解く
- 非常に強力な収束性（反復数を減らす）があるがメモリ消費が激しい
- NVIDIA CUSP LIBRARYの smoothed_aggregation コードを利用

対象計算:胸部大動脈血流

- 心臓から全身に血液を送る血管（図の赤い部分）

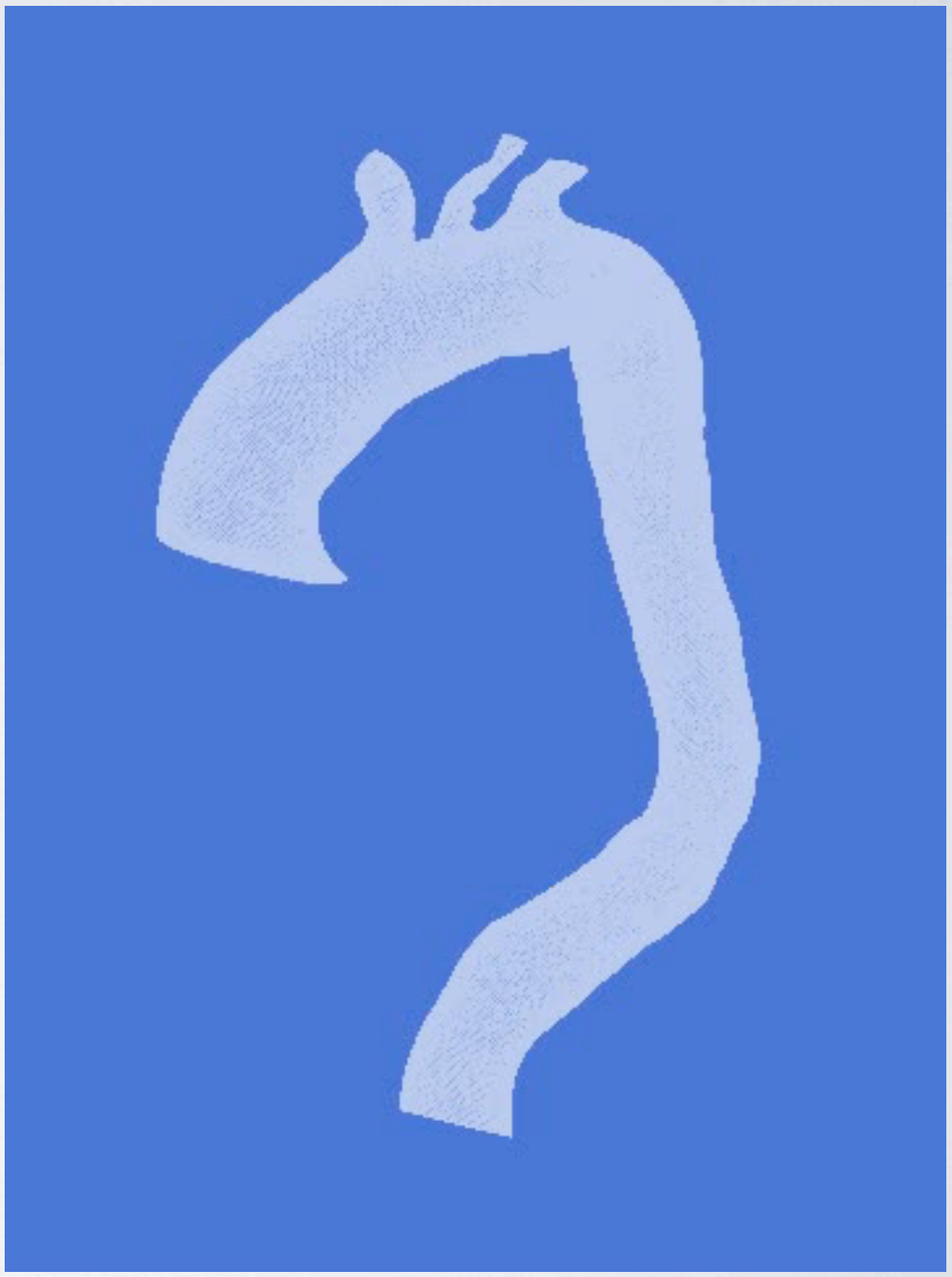
- 病理の原因となりうる，医療シミュレーションに重要な部位



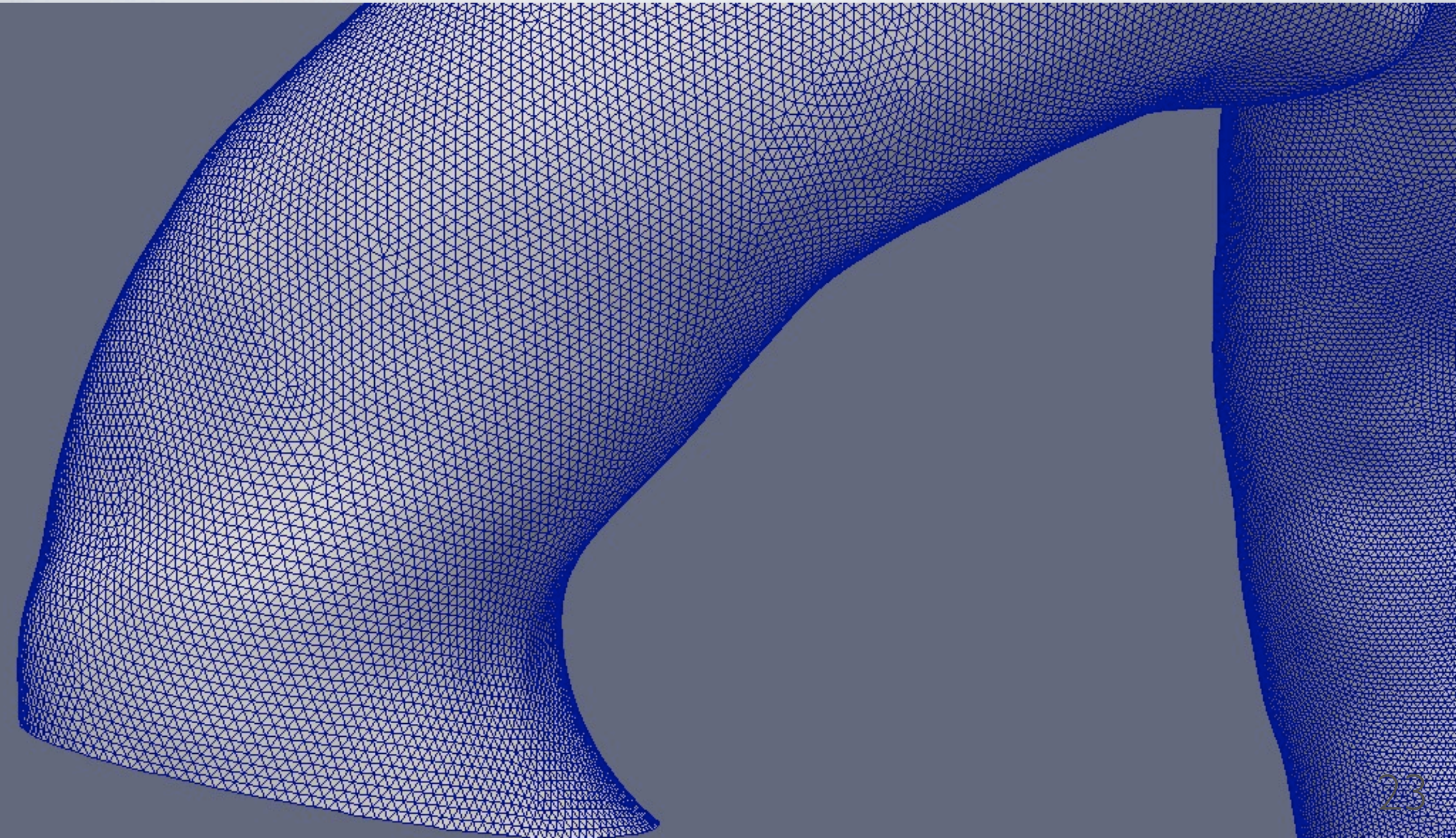
https://commons.wikimedia.org/wiki/File:Gray505.png

血管メッシュの作成

- MRI画像より血管部を抽出（金沢大学医学部との共同研究）
- Gambitによるメッシュ作成
- OF付属のツールでOFメッシュに変換



血管メッシュの作成

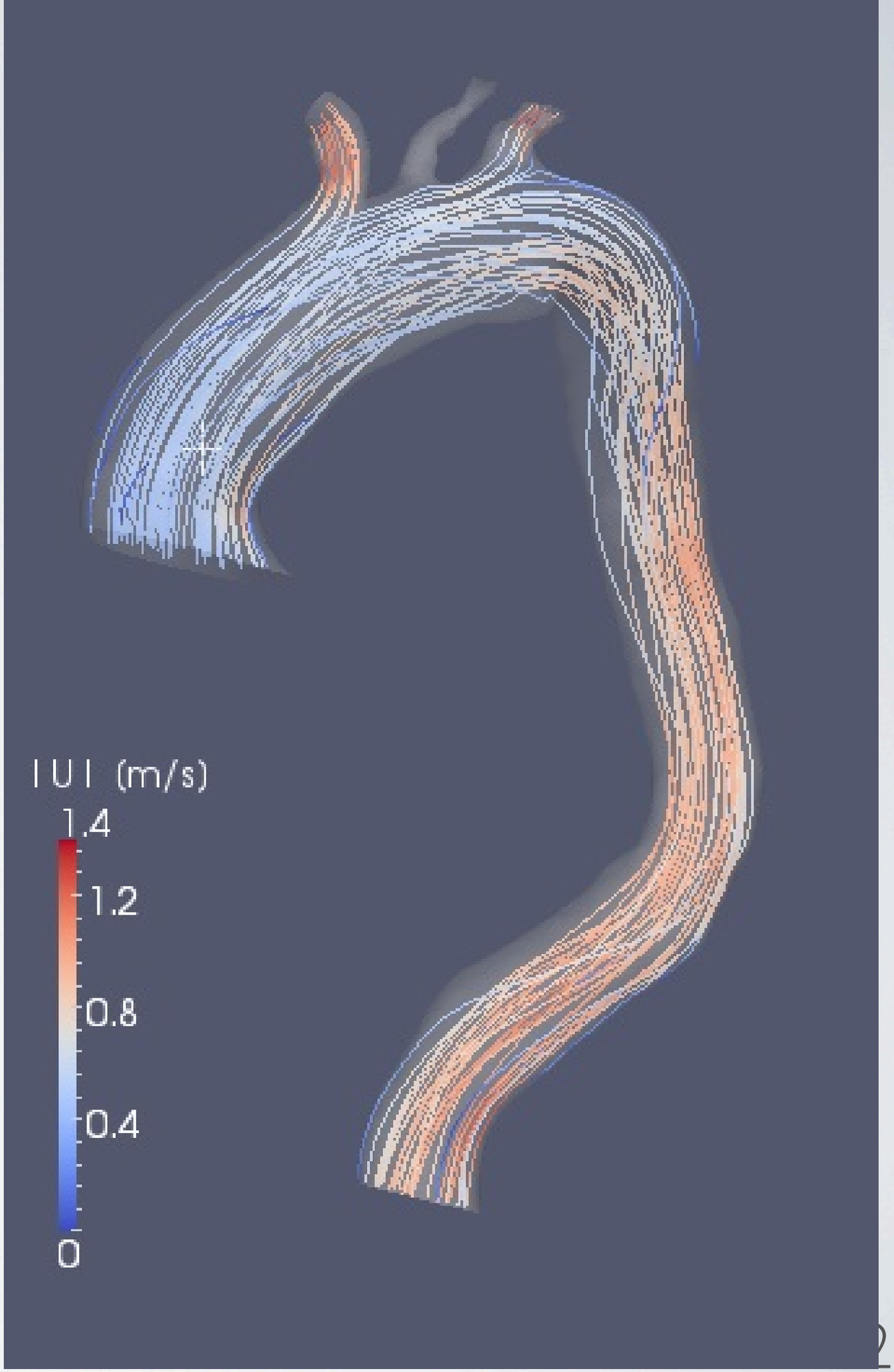


流体条件

ソルバ	simpleFoam (OpenFOAM-2.1.1)
物性値	動粘性係数 $\nu = 3.33 \times 10^{-6}$ [m^2/s](血液)
乱流モデル	層流モデル
流入側境界条件	流入速度 $V = 0.461$ [m/s] (Re $= 6500$)
流出側境界条件	分岐管 $P = 76$ [Pa], 腹部 $P = 0$ [Pa]
外部収束緩和係数	圧力・速度ともに 0.6
外部反復収束条件	$\|\delta P\|_1 \leq 1.0 \times 10^{-6}$ and $\|\delta V\|_1 \leq 1.0 \times 10^{-6}$
線形ソルバ	圧力 GPU-AMG-CG 法，速度 ILU-BiCG 法
内部反復収束条件	相対残差ノルム $\|\mathbf{r}\|_1 \leq 1.0 \times 10^{-8}$

可視化結果

- 1778SIMPLEステップで収束



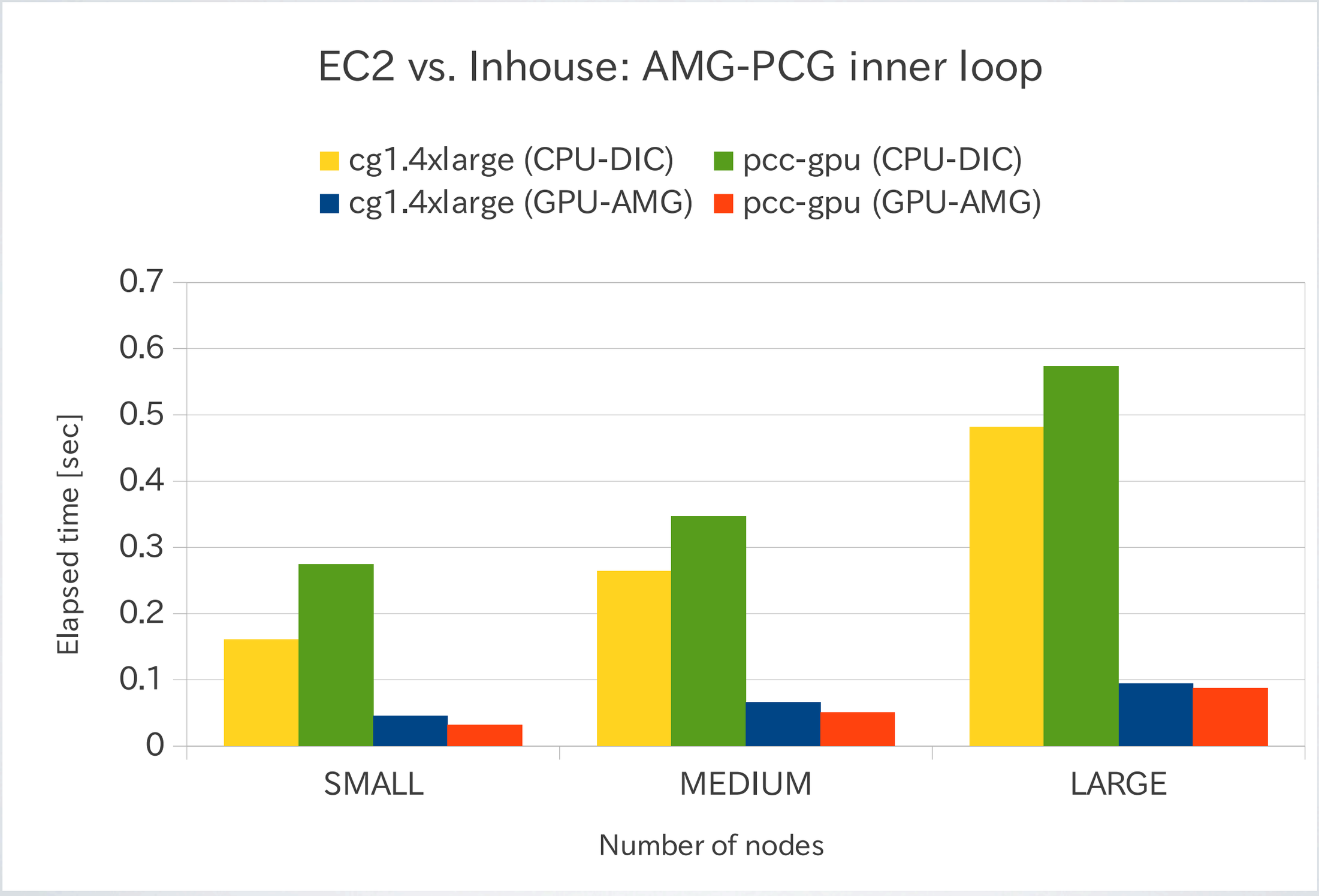
計算メッシュ

	SMALL	MEDIUM	LARGE
節点数	1,912,272	2,980,302	5,144,730
サイズ	155MB	311MB	543MB

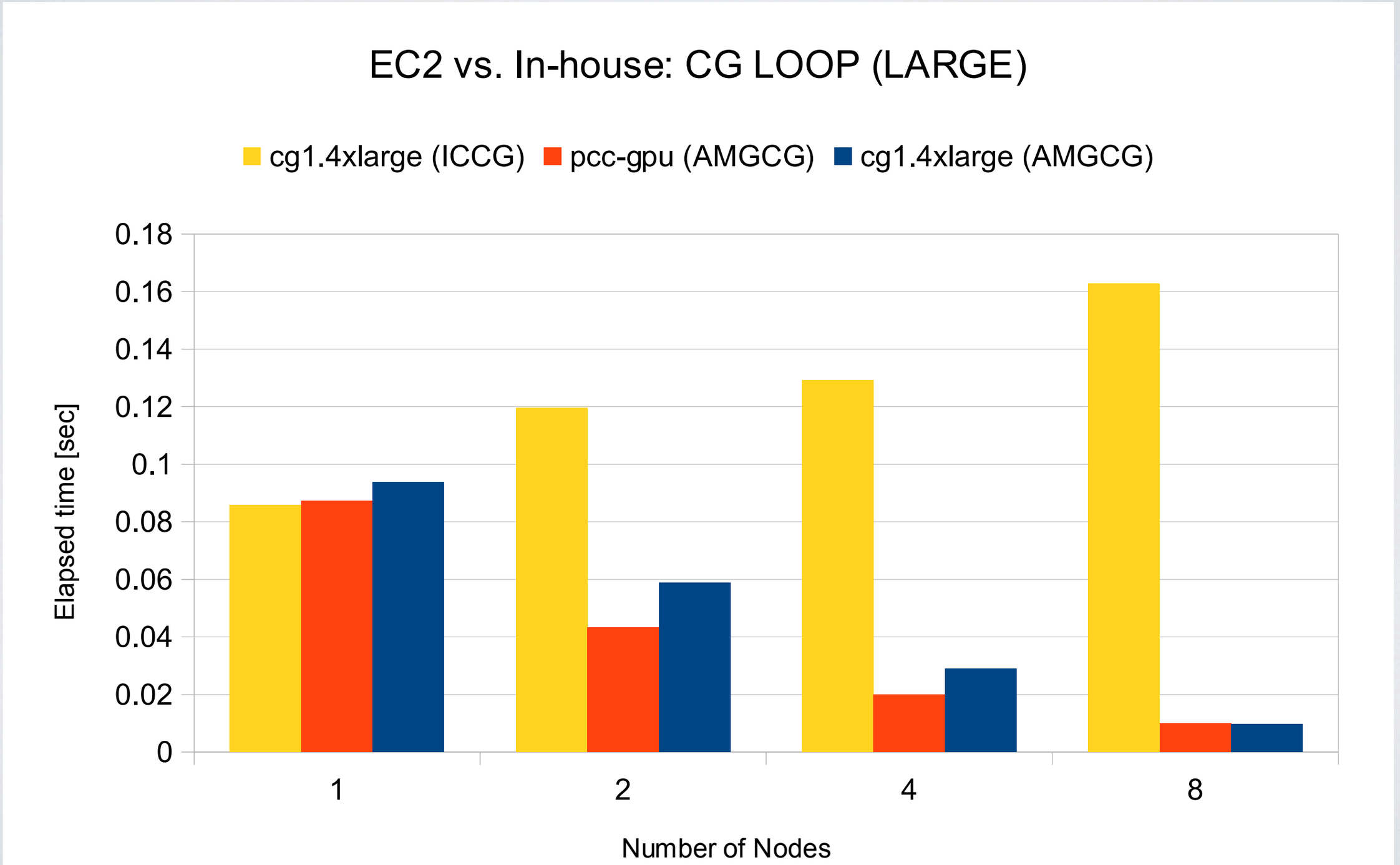
性能比較

- EC2でのCPU-ICCG法ソルバ（参考）
- EC2でのGPU-AMGCG法ソルバ
- JAIST内部 GPU ClusterでのGPU-AMGCG法ソルバ

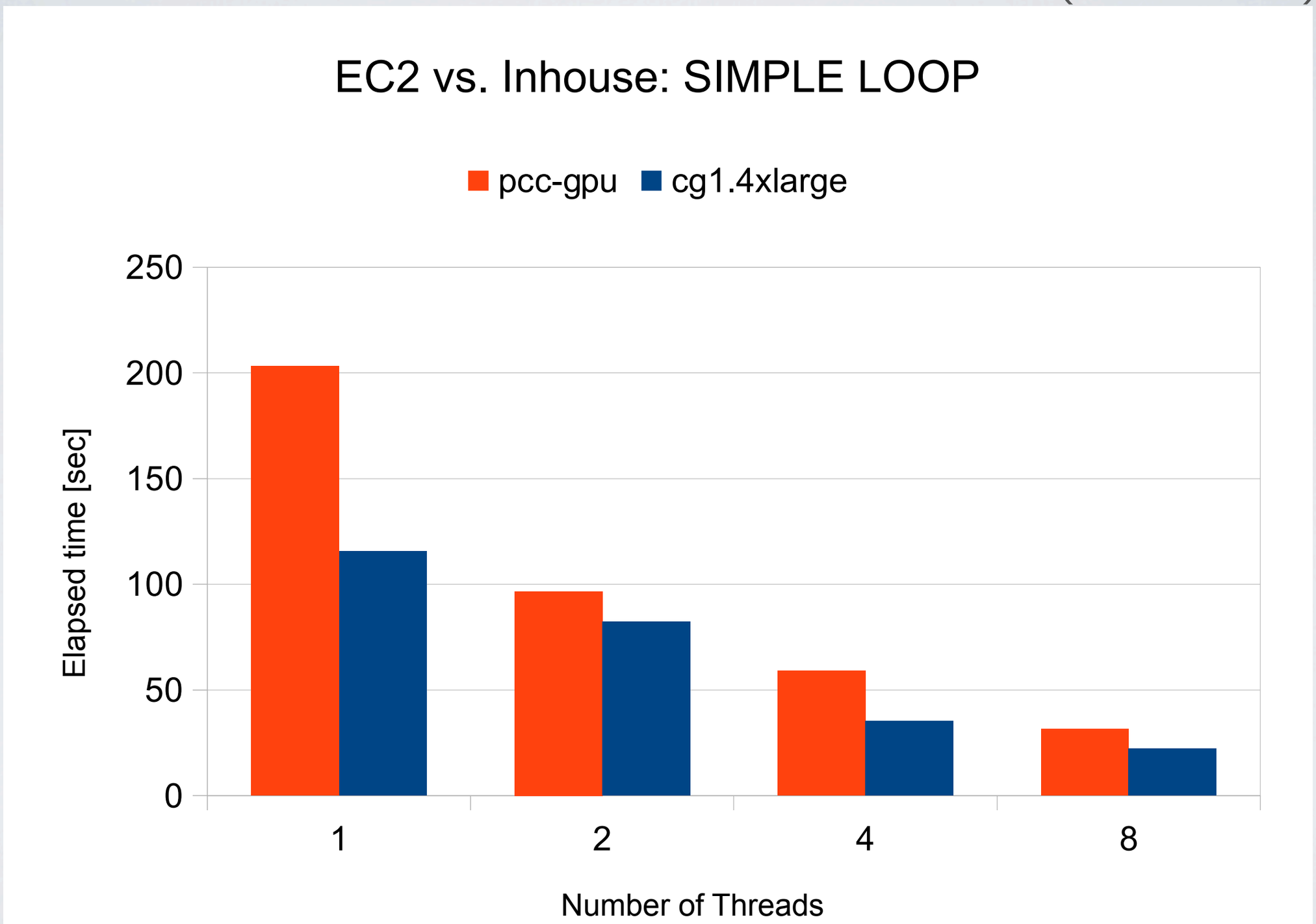
サイズに対するCG法ループ



ノード数に対するPCG法ループ



ノード数に対するSIMPLEループ(LARGE)



収束回数

- 相対残差 $|r|_1 < 1.0 \times 10^{-8} |r0|_1$

テキスト

並列数	1	2	4	8
ICCG	1005	1356	1362	1373
AMG-CG	41	94	139	198

結論

- EC2クラウド環境においてOpenFOAM流体計算の性能測定を行いCUDA ITSOLとNVIDIA CUSPを用いGPUソルバを構築した

- 大規模な血管形状メッシュに適用し，8ノードまで用いた並列計算においてスケールするコードを開発した