クラウド環境での GPGPU線形ソルバーによる OpenFOAMの性能向上

株式会社 EQN

松村　茂

2015.11.28

自己紹介

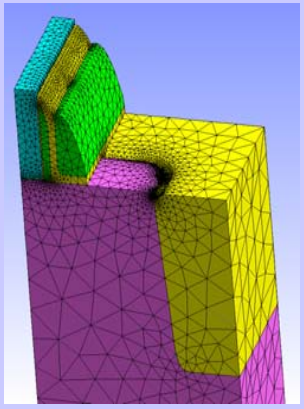
名前　松村 茂

シミュレーター設計開発コンサルティング

株式会社EQN　代表取締役

株式会社EQN 業務内容

- 直交ベース半導体シミュレーターの四面体メッシュ化
- 半導体シミュレーターの3D自由形状作成機能開発
- 流体計算シミュレーターの新機能物理モデル開発
- 非構造メッシャー開発
- GPGPU高速ソルバー、CPU並列化ソルバー開発
- 計算環境のクラウド化



線形ソルバーとは？

並列化・線形ソルバー基礎

線形ソルバーとは

$$\mathbf{A} \times \vec{x} = \vec{b}$$ (Aは行列)

- Aと$\vec{b}$が既知の時$\vec{x}$を計算する。
- 陰解法を用いるシミュレーターはこれに依存する。
- 計算時間のほとんど全てがこの計算に消費される。

行列の種類

密行列

- 行列全体のほとんどが0以外の要素からなる。

疎行列

- 行列要素のほとんどが0からなる。
- 隣接する要素間の関係のみ必要とする計算行列は疎行列
- **OpenFOAMの流体・構造計算は疎行列計算**

並列化・線形ソルバー基礎

- 線形行列ソルバーの種類

直接法(LU分解)

- 必ず何らかの解が得られる。
- 精度は保証されない。
- メモリ量O(n^2) (密行列)
- 計算時間O(n^3) (密行列)

  行列サイズn=1万(1万x1万の行列)の
  計算に1秒かかる場合、
  n=1億を計算するのに3万年以上かかる。

⇒ 大規模計算に向かない

反復法

- 発散等、解が得られる保証がない
- 解の精度は保証される
- 膨大なアルゴリズムがある

  CG、BiCG、BiCGStab、GMRES、AMG等

  ⇒ OpenFOAMではCG, BiCG, GAMG等
- 前処理を用いて収束性を上げる

  ヤコビ、SOR、IC、ILU前処理等

  ⇒OpenFOAMではDILU,DIC等

並列化・線形ソルバー基礎

線形ソルバーの例

PCG(Preconditioned Conjugate Gradient)

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{z}_0 := \mathbf{M}^{-1}\mathbf{r}_0$
$\mathbf{p}_0 := \mathbf{z}_0$
$k := 0$
repeat
$\alpha_k := \dfrac{\mathbf{r}_k^{\mathrm{T}}\mathbf{z}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small then exit loop end if
$\mathbf{z}_{k+1} := \mathbf{M}^{-1}\mathbf{r}_{k+1}$
$\beta_k := \dfrac{\mathbf{z}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{z}_k^{\mathrm{T}}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
The result is $\mathbf{x}_{k+1}$

行列・ベクトル積計算

前処理計算

計算コスト≒高速化の余地
行列ベクトル積・前処理 ＞ 内積 ＞ ベクトル加減・定数倍
特にILU前処理などは高コスト

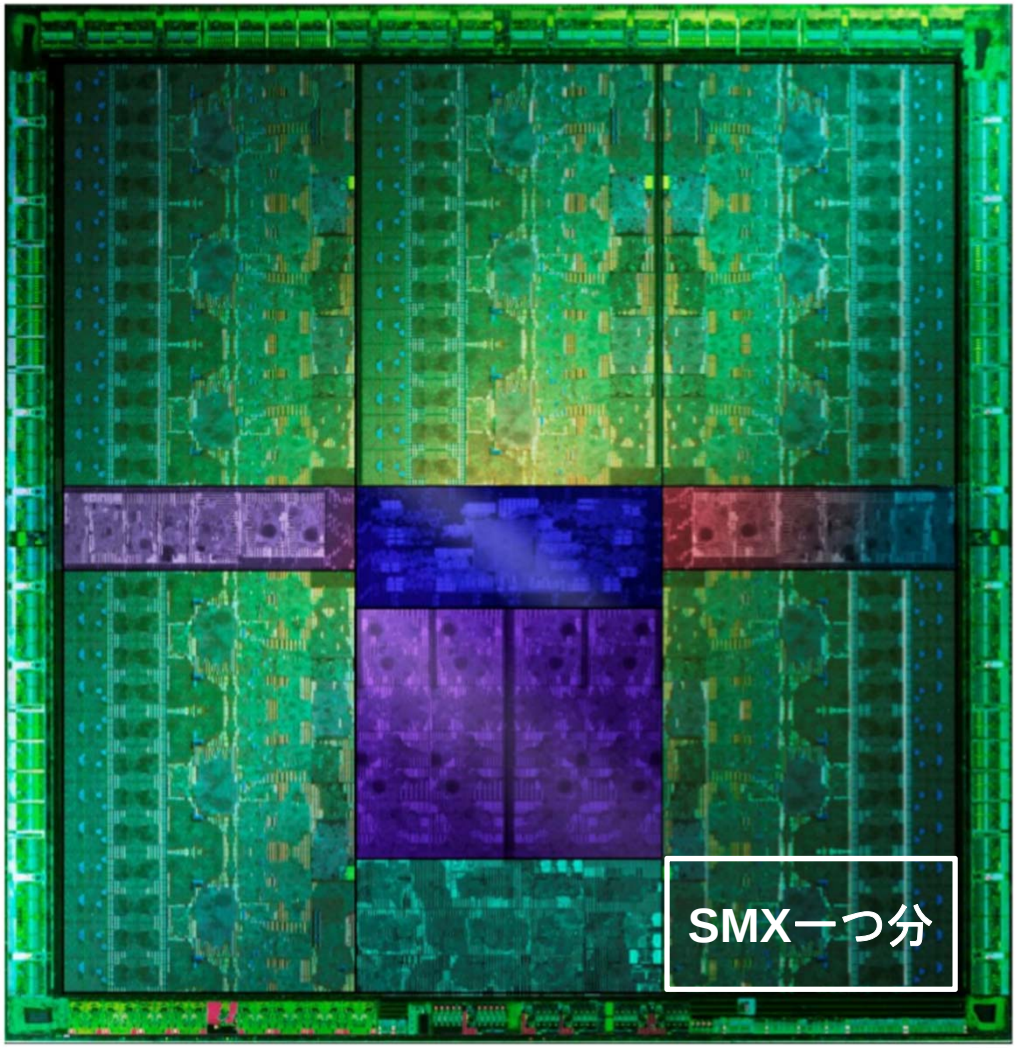
GPGPUを実行するGPUデバイスとは？

GPGPUデバイスの例

10万円強で買える
スーパーコンピューター

NVIDIA GeForce GTX Titan は11.12cm×26.67cmの
フルハイトPCIexのグラフィックカード

GK110 (Kepler)世代のチップ概要

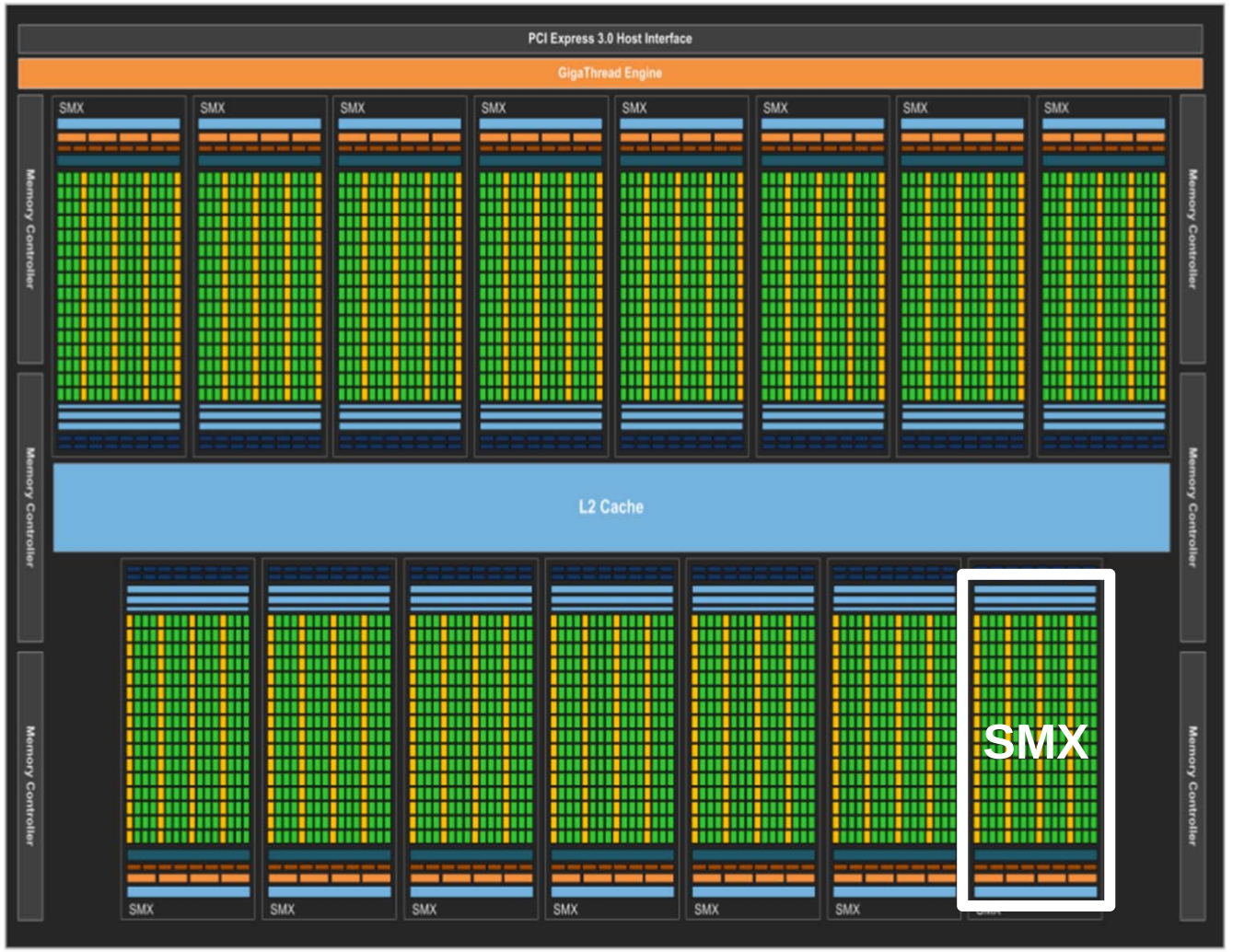


NVIDIA

©4gamer.net

- トランジスタ数 71億個
  (Xeon 56xx系で12億)
- 28nmプロセス
- ダイサイズは
  23.5×24.3mm
  (500円玉より大きい)

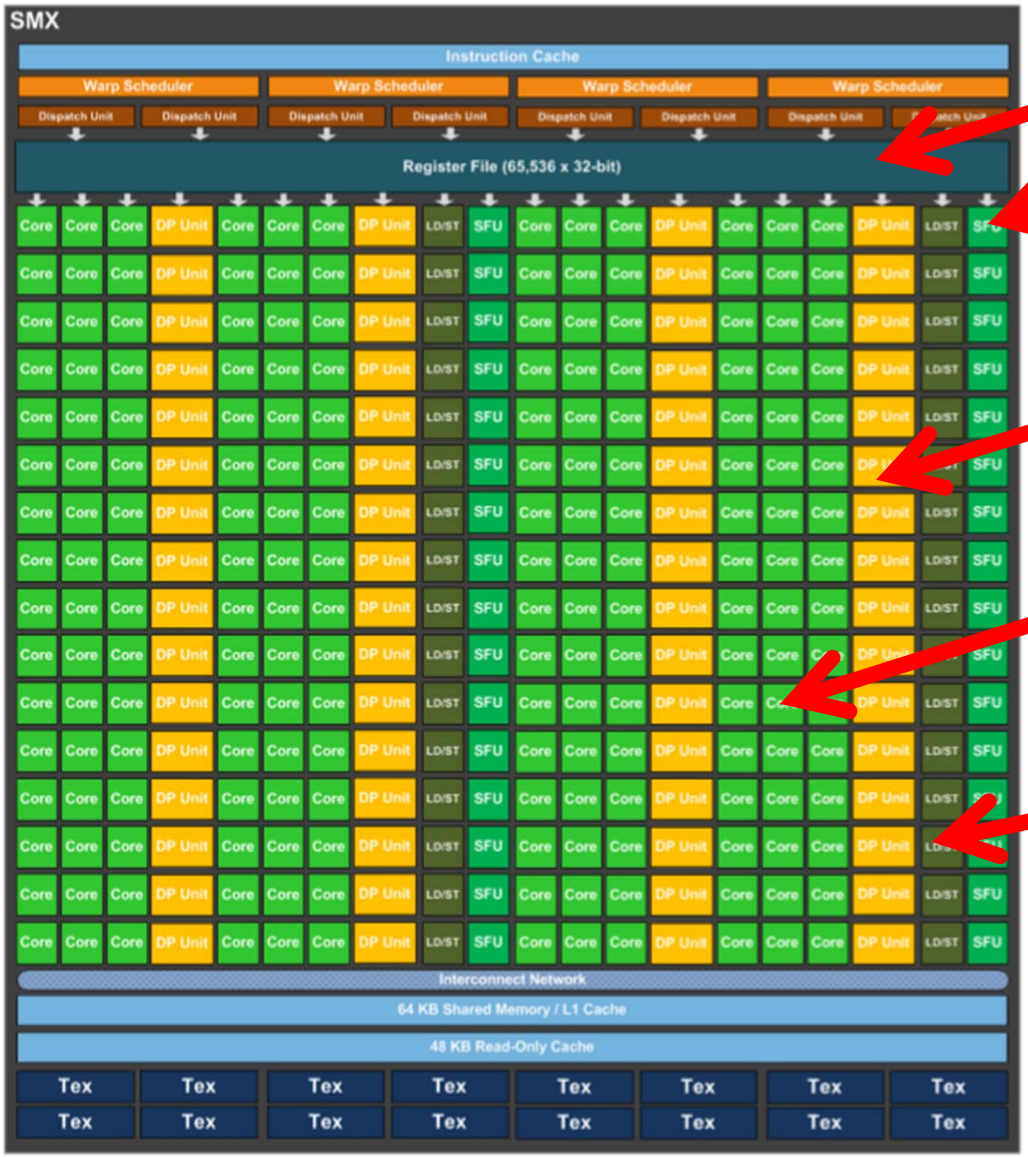
GK110 (Kepler)世代 ブロックダイアグラム



NVIDIA

SMX(ストリーミングマルチプロセッサー)が15個分入っている
Titan の場合は14個

SMXブロックダイアグラム



レジスタ　65536個

特殊関数(sin,log等)ユニット32個

倍精度ユニット64個

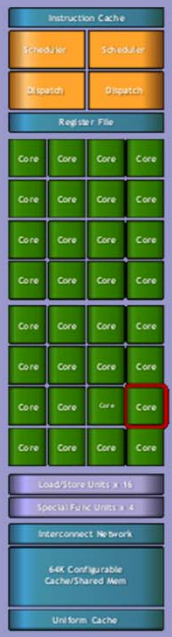
単精度CUDAコア192個

ロード/ストアユニット32個

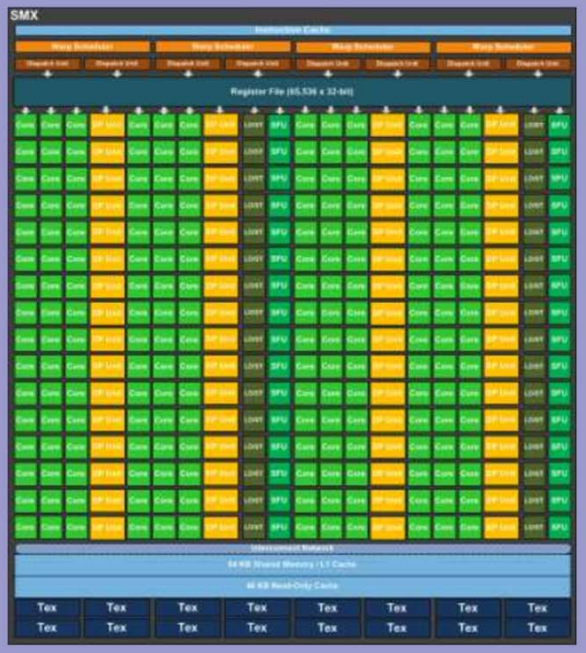
SMX：単精度 CUDA コア 192 個、倍精度ユニット 64 個、特殊関数ユニット（SFU）32 個、ロード/ストア・ユニット（LD/ST）32 個

NVIDIA

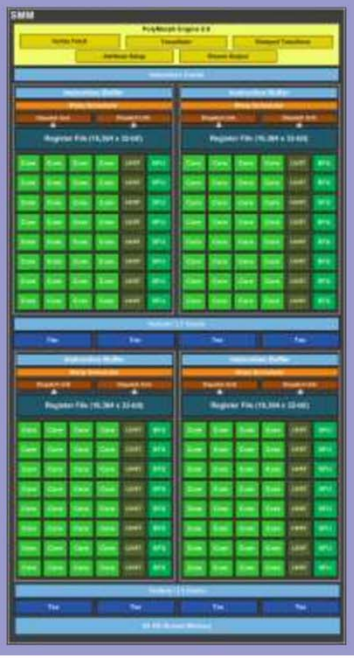
世代毎のストリームマルチプロセッサ



Fermi
CC 2.0 : 32 cores / SM



Kepler
CC 3.5 : 192 cores / SMX



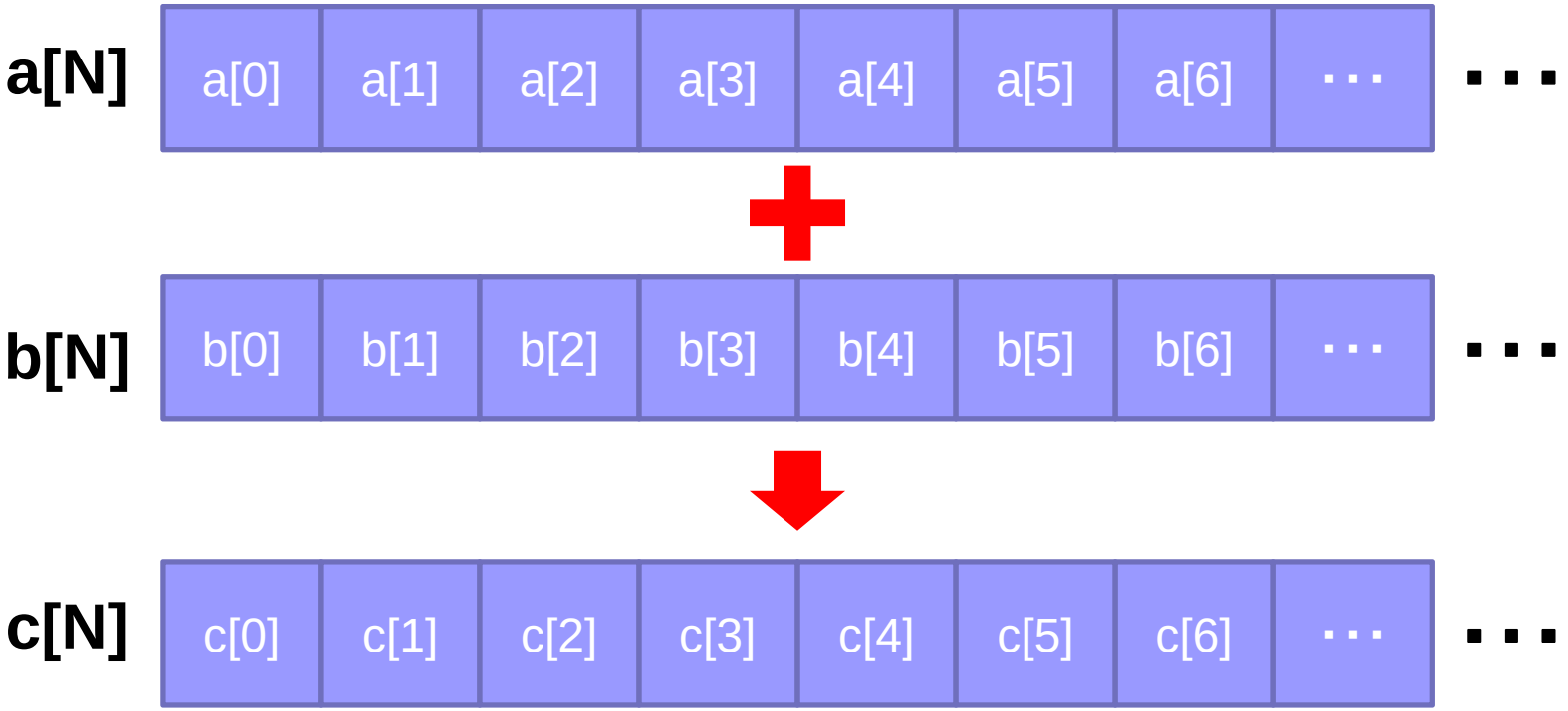
Maxwell (1st gen)
CC 5.0 : 128 cores / SMM

NVIDIA

並列化・線形ソルバー基礎

- cudaによる並列化の例

  ベクトル和の計算の例



並列化・線形ソルバー基礎

- 各手法によるベクトル和の計算の違い

シングルCPU

```
void add( int *a, int *b, int *c ) {
        for ( int tid = 0; tid < N; tid++ ) {
                c[tid] = a[tid] + b[tid];
        }
}
```

c配列=a配列+b配列

OpenMP

```
void add( int *a, int *b, int *c ) {
        #pragma omp parallel for
        for ( int tid = 0; tid < N; tid++ ) {
                c[tid] = a[tid] + b[tid];
        }
}
```

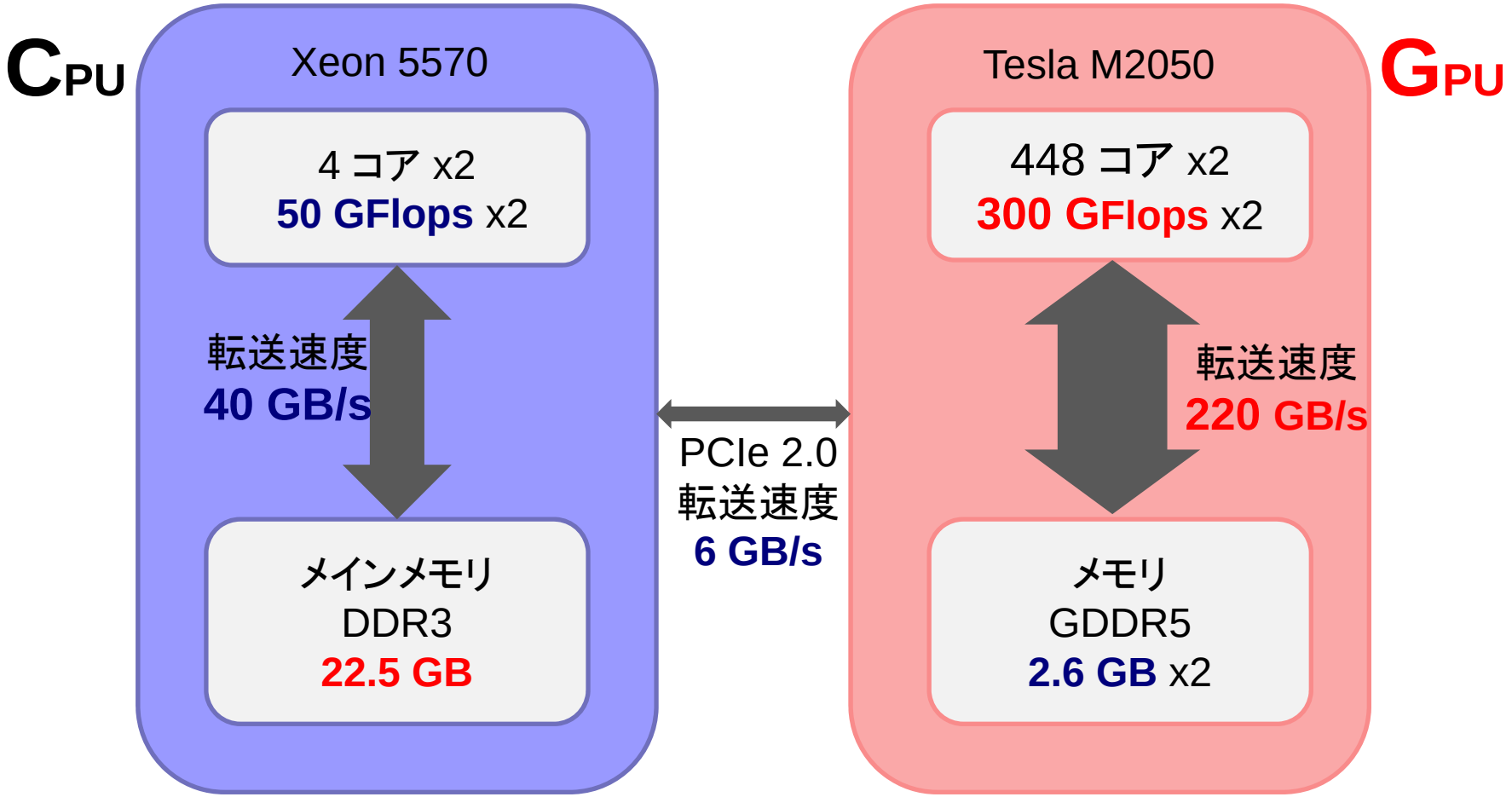
このループがCPU数に同時に展開される

CUDA

```
__global__ void add( int *a, int *b, int *c) {
        int tid = threadIdx.x + blockIdx.x * blockim.x;
        for( ; tid < N; tid += blockDim.x * gridDim.x ) {
                c[tid] = a[tid] + b[tid];
        }
}
```

この関数がGPUコアに同時に展開される

GPUシステムの概略・特徴(cg1.4xlarge)



(速度数値は全て概略値)

転送が極めて遅い
いかに転送を減らすかに高速化はかかっている

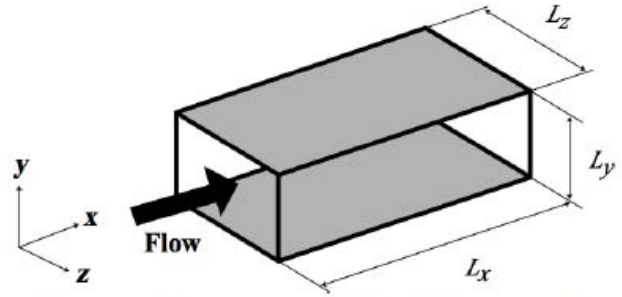
Amazon EC2のGPGPUインスタンスでのベンチマーク

OpenCAE学会 OpenFOAMベンチマーク

(OpenCAE学会今野氏 資料)

チャンネル流れケース

チャンネル流れケースchannelReTau110は，Re_τ=110のチャン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である．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，結果検証(Validation)が容易である．



Re_τ=110

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

主流方向(X)は一定の圧力勾配
その他の方向(Y，Z)は周期境界

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.35-40,2 003

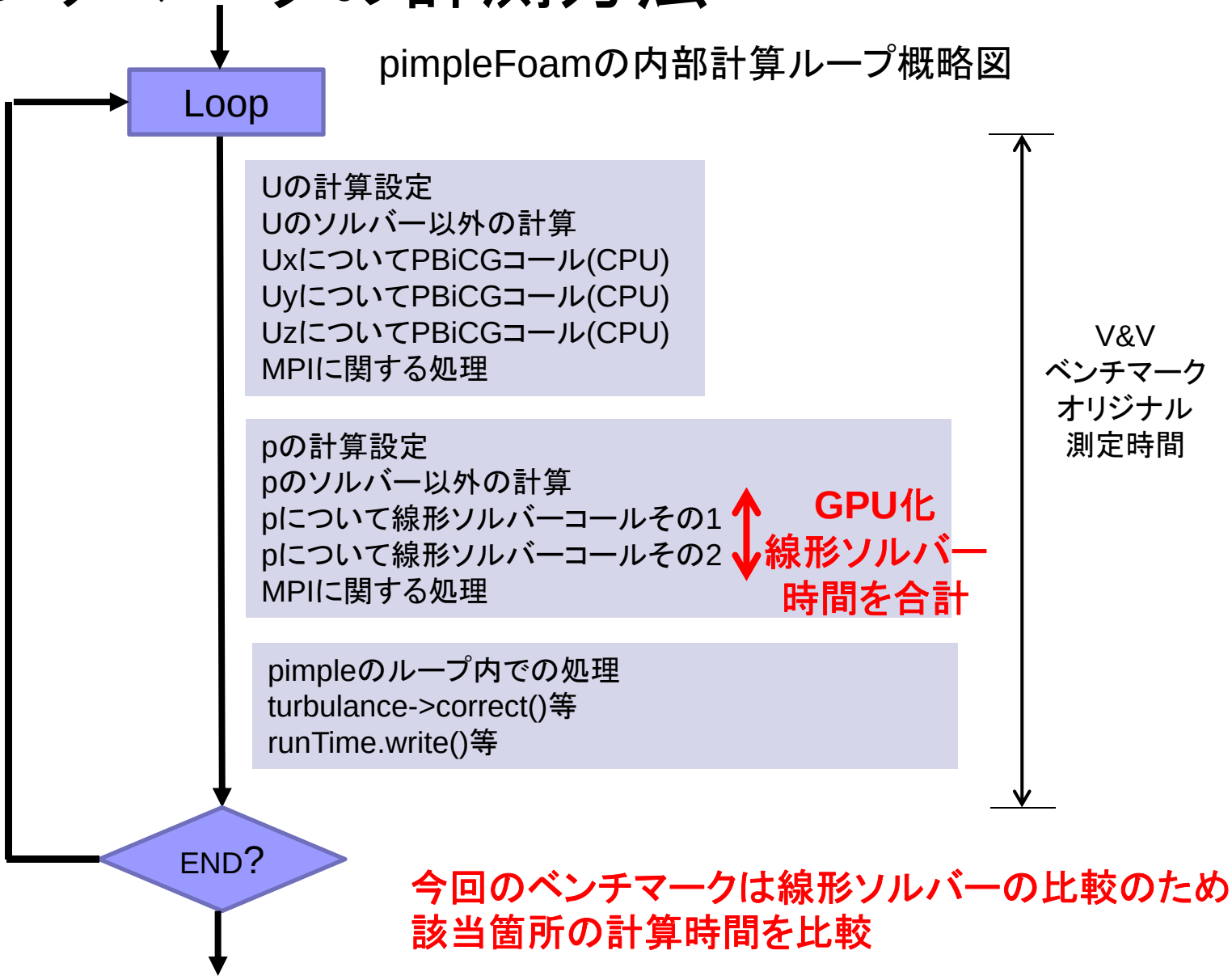
ここでは，channelReTau110のケースを対象にして，格子数，MPI並列数および圧力線型ソルバを変化させたベンチマークテストを行い，FX10の並列化効率や実行性能を簡易に調べる．channelReTau110ケースの設定詳細は省略する．

[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

OpenCAE学会のベンチマーク題材を用いてパフォーマンスを確認

ベンチマークの計測方法

pimpleFoamの内部計算ループ概略図



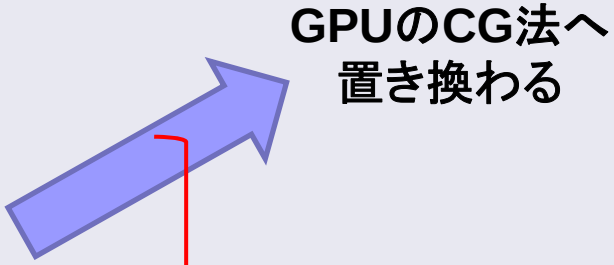
今回のベンチマークは線形ソルバーの比較のため
該当箇所の計算時間を比較

ベンチマークの計測方法

src/OpenFOAM/matrices/lduMatrix/solvers/PCG/PCG.Cの例

```
Foam::solverPerformance Foam::PCG::solve
(
    scalarField& psi,
    const scalarField& source,
    const direction cmpt
) const
{
        timer_start(); // 追加

        timer_end(); // 追加
        return solverPerf;
}
```



solve関数本体
PCG法でA･x = b を解く

計測時間

GPGPUソルバー比較

	Symscape ofgpu v1.1	EQN Solver	RapidCFD
形式	**全体変更(patch)**	**ライブラリ**	**全体変更**
GPU化部分	**線形ソルバー**	**線形ソルバー**	**フルGPU**
マルチGPU	× 各ホスト1GPU並列のみ	○	○
CG/BiCG	○	○	○
PCG/PBiCG diagonal	○	○	○
PCG/PBiCG DIC/DILU	○	○	×
PCG/PBiCG smoothed aggregation	○	○	×
GAMG	×	△	○
導入難易度	× **v2.3は導入不可** (弊社で勝手に移植改良)	◎ **ファイル置くだけ**	△ **make難易度高**

GPGPUソルバー比較

	Symscape ofgpu v1.1	EQN Solver	RapidCFD
形式	全体変更(patch)	ライブラリ	全体変更
GPU化部分	線形ソルバー	線形ソルバー	フルGPU
マルチGPU	× 各ホスト1GPU並列のみ	○	○
CG/BiCG	○	○	○
PCG/PBiCG diagonal	○	○	○
PCG/PBiCG DIC/DILU	○	○	×
前処理がAMG PCG/PBiCG smoothed aggregation	○	○	×
ソルバーがAMG GAMG	×	△	○
導入難易度	× v2.3は導入不可	◎ ファイル置くだけ	△ make難易度高

一般的手法で比較

AMG系で比較

(弊社で勝手に移植)

Amazon EC2 GPUクラウド

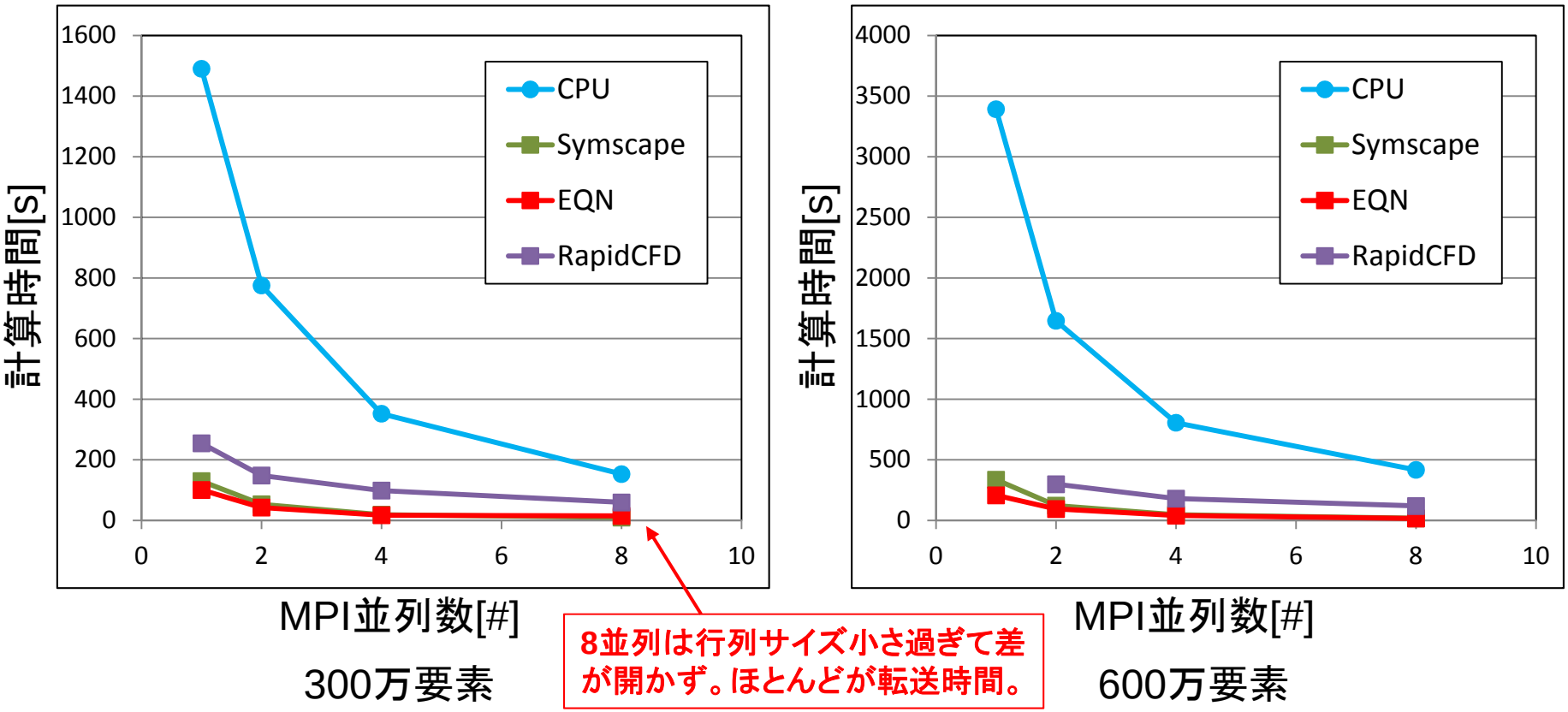
Amazon EC2 GPGPUインスタンススペック一覧

インスタンスタイプ	g2.2xlarge	cg1.4xlarge
CPU	Xeon E5-2670	Xeon X5570
メモリ	15GiB	22.5GiB
ECUs	26	33.5
vCPUs	8	16
ComputeCapability	3.0	2.0
GPU	Grid K520*1	Tesla M2050*2
GDDR5	4096 MB	2687 MB
CUDA Core	1536	448
GPU Clock	797 MHz	1147 MHz
Memory Clock	2.5 GHz	1546 MHz
単精度演算性能x1	1128 GFlops	585 GFlops
単精度演算性能x2		1155 GFlops
倍精度演算性能x1	**83 GFlops**	**286 GFlops**
倍精度演算性能x2		585 GFlops
料金(Virginia, RHEL)	$0.65 / 1時間	$2.10 / 1時間

まずはこちら

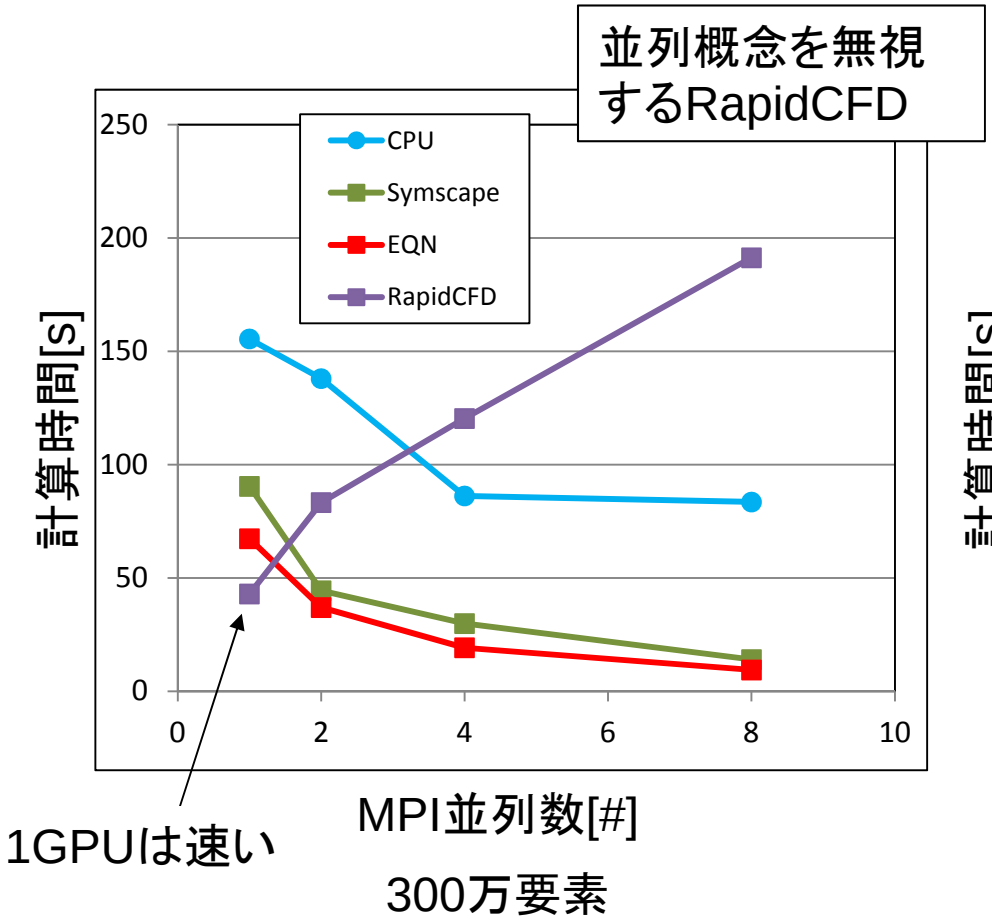
ベンチマーク結果

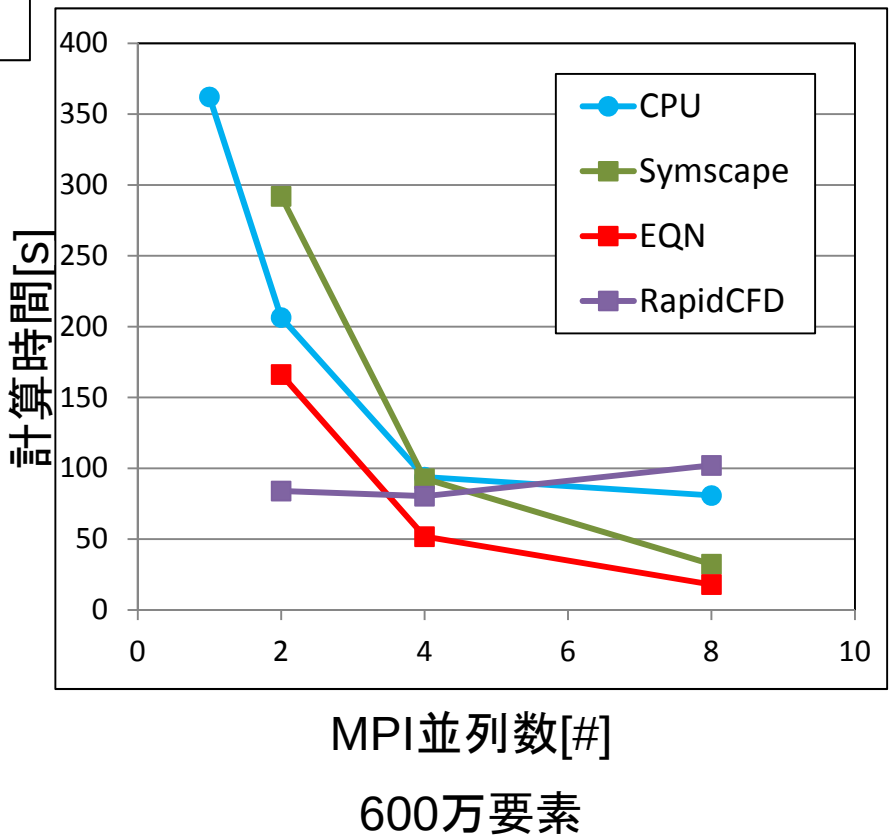
※プロットの無いところはGPUメモリ不足のため計算できず



PCG-diagonal計算(g2インスタンス、Smagorinskyモデル)

ベンチマーク結果





AMG系計算(g2インスタンス、Smagorinskyモデル)

CPU、RapidCFDはGAMG

Symscape、EQNはPCG-Smoothed_Aggregation

Amazon EC2 GPUクラウド

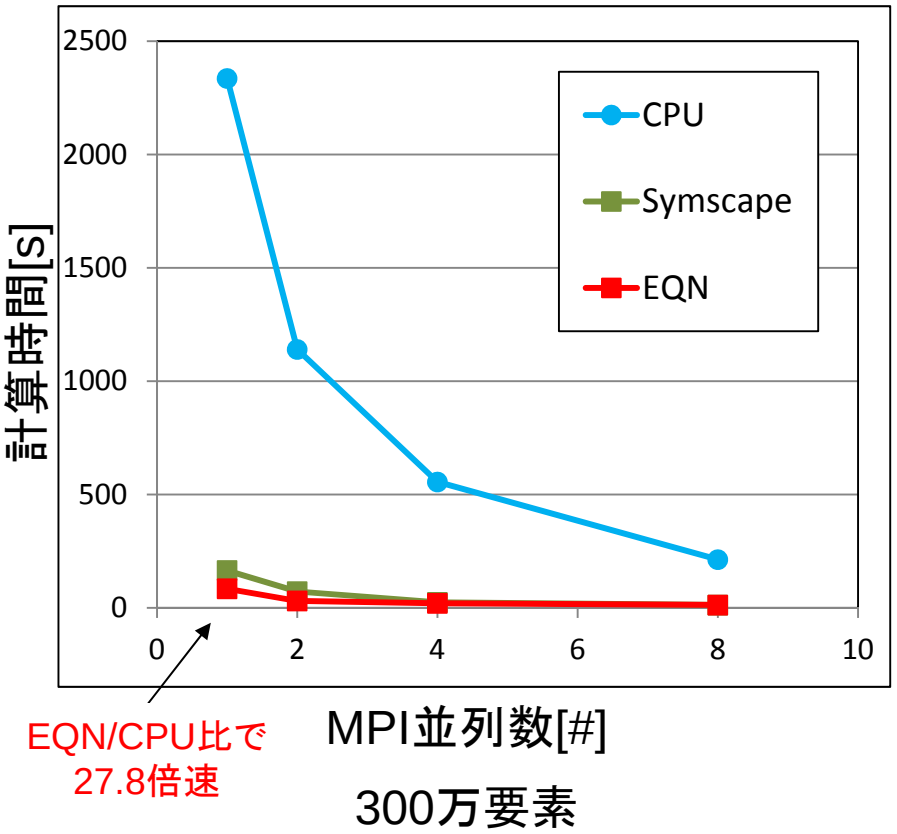
Amazon EC2 GPGPUインスタンススペック一覧

インスタンスタイプ	g2.2xlarge	cg1.4xlarge
CPU	Xeon E5-2670	Xeon X5570
メモリ	15GiB	22.5GiB
ECUs	26	33.5
vCPUs	8	16
ComputeCapability	3.0	2.0
GPU	Grid K520*1	Tesla M2050*2
GDDR5	4096 MB	2687 MB
CUDA Core	1536	448
GPU Clock	797 MHz	1147 MHz
Memory Clock	2.5 GHz	1546 MHz
単精度演算性能x1	1128 GFlops	585 GFlops
単精度演算性能x2		1155 GFlops
倍精度演算性能x1	**83 GFlops**	**286 GFlops**
倍精度演算性能x2		585 GFlops
料金(Virginia, RHEL)	$0.65 / 1時間	$2.10 / 1時間

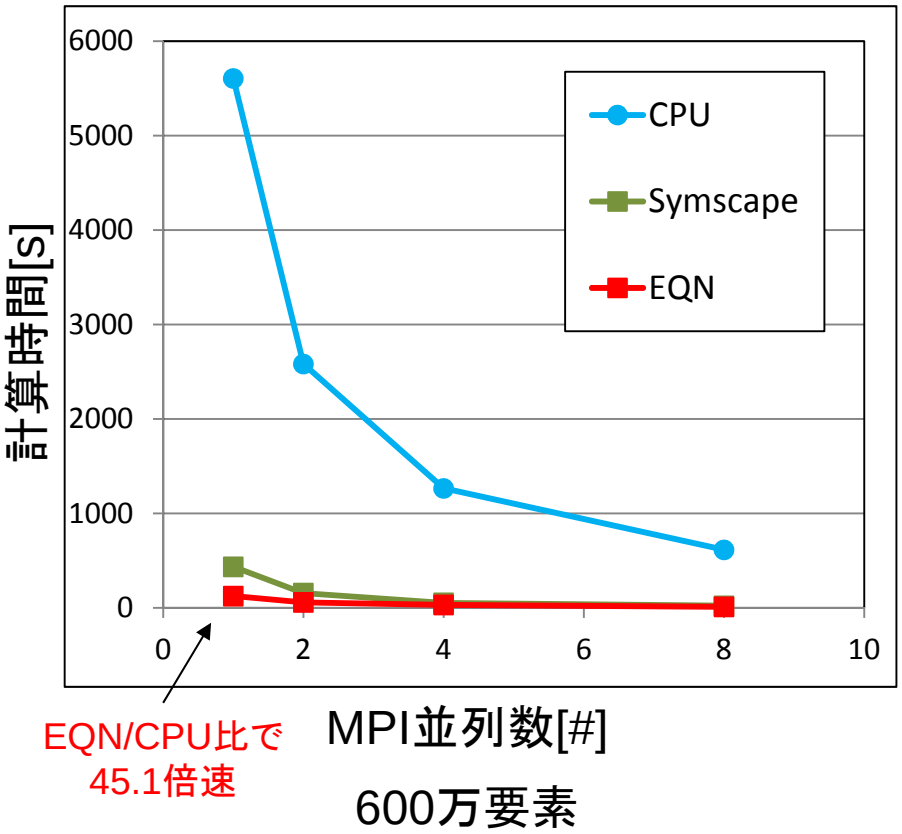
次はこちら

ベンチマーク結果

※RapidCFDはComputeCapability3.0以上でないと計算不能のため起動せず



300万要素

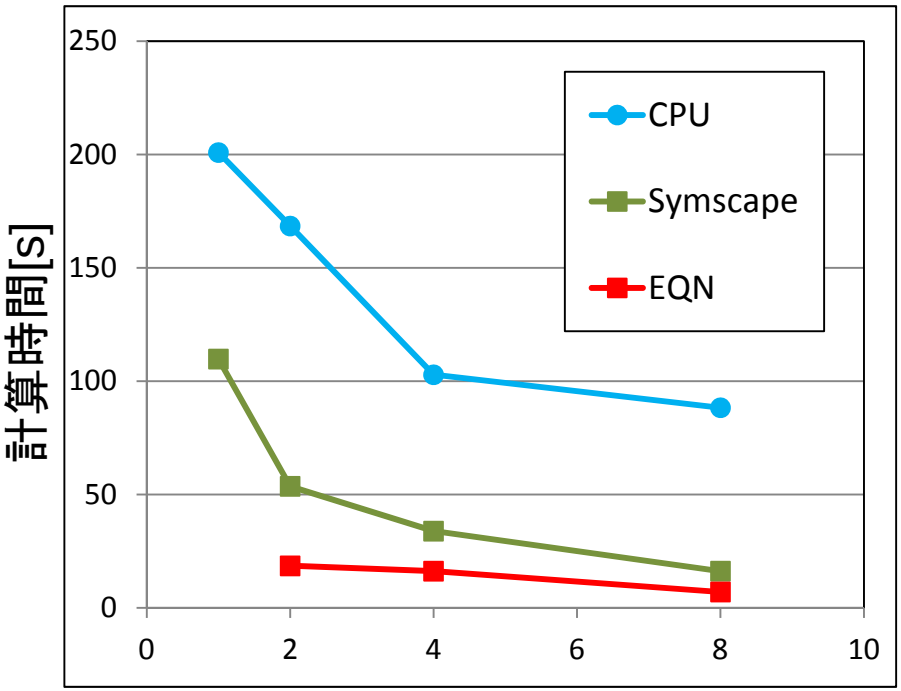


600万要素

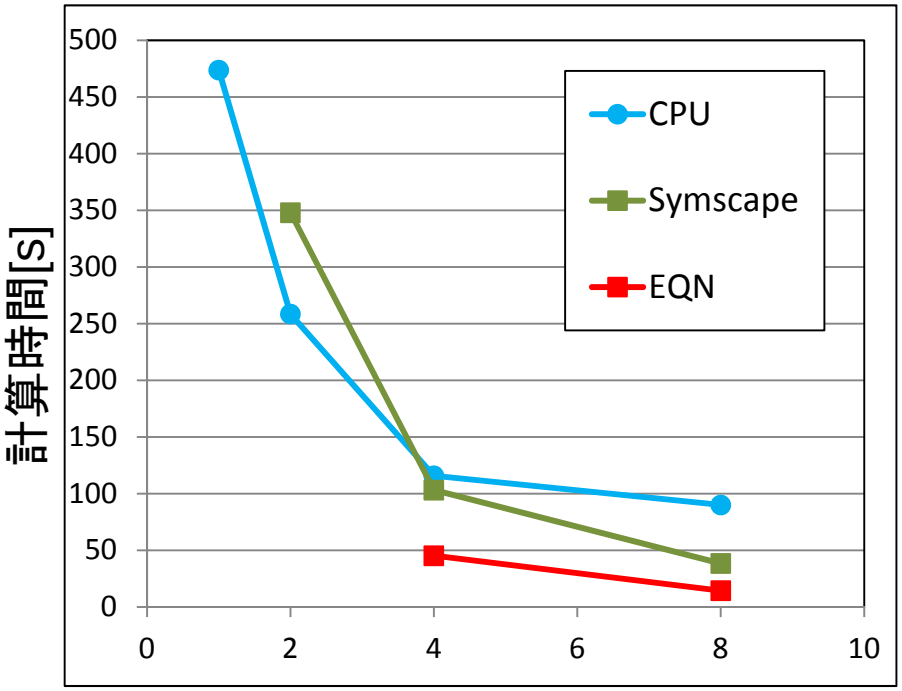
PCG diagonal計算(cg1インスタンス、Smagorinskyモデル)

ベンチマーク結果

※RapidCFDはComputeCapability3.0以上でないと計算不能のため起動せず



300万要素



600万要素

AMG系計算(g2インスタンス、Smagorinskyモデル)

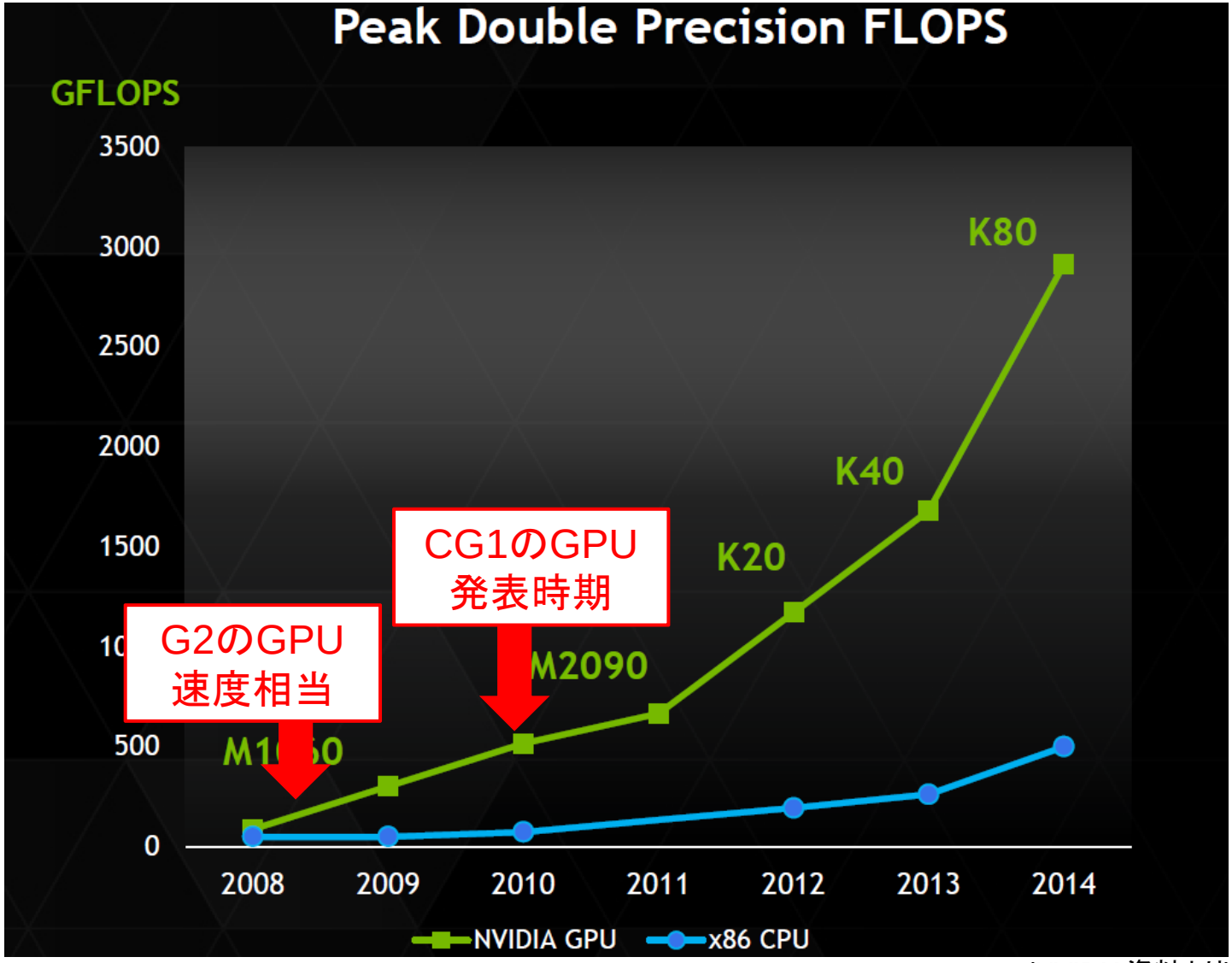
CPUはGAMG

Symscape、EQNはPCG-Smoothed_Aggregation

高速化状況について

一見、GPUによって十分に高速化されて
いるように思えますが…

NVIDIA GPU 計算速度



(NVIDIA資料より)

最新のGPUを用いた場合、更に圧倒的な高速化が可能！

まとめ・コメント

- EQNソルバーはじめSymscape、RapidCFD各社GPUソルバーでCPUよりも大幅な高速化を実現できた
- 計算量が小さすぎて、一回あたり1秒以内等の計算ではGPUの性能がまだまだ発揮しきれていない
- g2インスタンスは8円/時間程度のスポット料金なので積極的に高速化に利用しましょう
- 次回、K80での大幅な高速化の成果発表にご期待ください！