2.7 電磁場解析コード ADVENTURE_Magnetic 性能評価

名古屋大学情報基盤センター
荻野　正雄

2.7.1 はじめに

ADVENTURE とは、設計用大規模計算力学システム開発プロジェクト(通称 ADVENTURE プロジェクト)において開発されているオープンソースCAEシステムである。大規模メッシュを用いて自然物や人工物を丸ごと詳細にモデル化し、多様な並列分散計算機環境のもとで固体の変形や熱・流体の流れ等の力学解析から可視化、設計最適化までを行える特徴がある。無料公開されていることもあり、教育・研究から産業応用まで幅広く利用されているソフトウェアである。ここでは、電磁場解析モジュール ADVENTURE_Magnetic の測定を行う。ADVENTURE_Magnetic は、非線形静磁場問題、時間調和渦電流問題、さらに非定常渦電流問題の有限要素解析機能を有しており、磁気ベクトルポテンシャルを未知数とするA法または$A-\phi$法の定式化、ベクトル関数を要素辺上で評価する Nédélec 要素を用いている。また、非構造格子を用いた有限要素解析では、大規模で疎な行列を係数に持つ連立一次方程式を解く必要があり、マルチコア・メニーコア上で高い並列効率や演算効率を出すことが困難な分野とされている。これに対し、ADVENTURE_Magnetic は領域分割法(Domain Decomposition Method: DDM)を採用することで、分散メモリ型並列環境において高い並列効率が得られ、将来の計算機においても高性能な計算が期待できるアプリケーションである。

2.7.2 プログラム概要

今回は時間調和渦電流問題と非定常渦電流問題解析を対象とする。$A-\phi$法による定式化は時間調和渦電流問題が、

$$\left(\nu \operatorname{rot} A_h,\ \operatorname{rot} A_h^*\right) - \left(i\omega\sigma\, A_h, A_h^*\right) + \left(\sigma \operatorname{grad} \phi_h,\ A_h^*\right) = \left(\tilde{J}_h,\ A_h^*\right) \tag{1a}$$

$$\left(\sigma \operatorname{grad} \phi_h, \operatorname{grad} \phi_h^*\right) - \left(i\omega\sigma\, A_h, \operatorname{grad} \phi_h^*\right) = 0 \tag{1b}$$

非定常渦電流問題が、

$$\left(\nu \operatorname{rot} A_h^{n+1},\ \operatorname{rot} A_h^*\right) + \left(\frac{\sigma}{\Delta t}\ A_h^{n+1}, A_h^*\right) + \left(\sigma \operatorname{grad} \phi_h^{n+1},\ A_h^*\right) = \left(\tilde{J}_h^{n+1},\ A_h^*\right) + \left(\frac{\sigma}{\Delta t}\ A_h^{n}, A_h^*\right) \tag{2a}$$

$$\left(\sigma\, A_h^{n+1}, \operatorname{grad} \phi_h^*\right) + \Delta t\left(\sigma \operatorname{grad} \phi_h^{n+1}, \operatorname{grad} \phi_h^*\right) = \left(\sigma\, A_h^{n}, \operatorname{grad} \phi_h^*\right) \tag{2b}$$

となる。ここで、Aは磁気ベクトルポテンシャル、ϕは電気スカラーポテンシャル、νは磁気抵抗率、Jは電流密度、iは虚数単位、ωは角周波数、σは導電率、Δtは時間刻み幅、下付き添え字hは有限要素近似、上付き添え字$*$は任意の試験関数、上付き添え字nは時間ステップを表す。式(1)-(2)より、最終的に解くべき式は時間調和渦電流問題で複素対称連立一次方程式、非定常渦電流問題で実対称連立一次方程式となる。さらに、ADVENTURE_Magnetic における DDM では、解くべき連立一次方程式を Schur 補元方程式へと変換する方法を採用している。まず、解くべき式を次式で表す。

$$\mathbf{Kx} = \mathbf{b} \tag{3}$$

ここで、係数行列$\mathbf{K}$は大規模かつ疎な行列となる。次に、FEM のメッシュをN個の部分領域に分割し、領域間境界上自由度への静的縮約を行うことで、次式で表される Schur 補元方程式が得られる。

$$\sum_{i=1}^{N} \mathbf{R}_B^{(i)T}\left(\mathbf{K}_{BB}^{(i)} - \mathbf{K}_{BI}^{(i)}\mathbf{K}_{II}^{(i)\dagger}\mathbf{K}_{IB}^{(i)}\right)\mathbf{R}_B^{(i)}\,\mathbf{x}_B = \sum_{i=1}^{N} \mathbf{R}_B^{(i)T}\left(\mathbf{b}_B^{(i)} - \mathbf{K}_{BI}^{(i)}\mathbf{K}_{II}^{(i)-1}\mathbf{b}_I^{(i)}\right) \tag{4}$$

ここで、$\mathbf{R}$は全体から部分領域への自由度制限行列、下付き添え字Bは領域間境界上自由度に関する項、Iは部分領域内部自由度に関する項、上付き添え字(i)は部分領域iに関する項、Tは転置

行列、†は一般化逆行列を表す。この式は、非定常渦電流問題であれば共役勾配（CG）法、時間調和渦電流問題は共役直交共役勾配（COCG）法などで解くことができる。また、必要となる係数行列と修正方向ベクトルの積演算は第 3 回 WG 資料 7 ページに示すように、部分領域ごとの疎行列ベクトル積演算 2 回と連立一次方程式解析 1 回に置き換えられ、部分領域ごとの連立一次方程式は不完全コレスキー分解前処理付きの CG 法または COCG 法で解いている。疎行列の格納形式は CSR を用いている。

次に、ADVENTURE_Magnetic のコーディング指針について述べる。本プログラムは C 言語で記述されている。前述のとおり複数の解析機能があるが、それらは有限要素法や領域分割法の実装で共通化できる箇所が多く、また線形代数関係コードも実数と複素数の違い以外はほぼ共通化できる。そこで、第 3 回 WG 資料 8 ページに示すように、共通コードとマクロ定義の組み合わせで全体のプログラム行数を削減する工夫が行われている。今回はこのようなコードの性能評価結果について紹介する。

2.7.3 測定環境

性能測定には、名古屋大学情報基盤センターの FX10 と FX100、及び理化学研究所の京を使用した。それぞれの環境におけるコンパイラとコンパイラオプションは WG 資料に示す通りである。

2.7.4 測定結果

まず、ADVENTURE_Magnetic の共通化コードとマクロ定義の組み合わせによるプログラムに対するコンパイラの最適化情報について述べる。第 3 回 WG 資料 8 ページに示すように、C プログラムのソースファイルを#include 指示で取り込んでおり、定義したマクロによって例えば実数向け・複素数向けとなるように実装されている。このようなプログラムを富士通社製コンパイラにかけた場合、最適化情報を含んだソースリストが表示されず、最適化作業が行いづらいという問題が生じた。そこで今回は、ソースリストを見るためにインクルードするソースファイルを手動で取り込んだファイルを別途用意することで、その問題を回避した。

次に、ADVENTURE_Magnetic と同様に領域分割法を採用する構造解析コード ADVENTURE_Solid の演算性能について述べる。第 3 回 WG 資料 10 ページに示すように、FX10 においてピーク性能比 4%以上が得られていた。これは、3 次元構造解析では節点あたり 3 自由度を持つため、それを利用して手動ループアンローリングが可能であったことが要因の 1 つである。しかし、ADVENTURE_Magnetic では要素辺あたり 1 自由度のみのため、同様の最適化を行えない。これにより、ピーク性能比は構造解析に比べて低下することが予想される。実測した結果を第 7 回 WG 資料に示す。6 ページに as is コード、コンパイラへの最適化指示行追加、一部のループ入れ替え、realloc 関数呼び出し削減を行ったコードでの FX100 におけるピーク性能比を示す。非定常渦電流問題でピーク性能比 0.7～0.9%、時間調和渦電流問題で 1.6%～2.4 程度であり、実数問題における性能の低さが際立つ結果となった。一方、複素数の乗算は実数に比べて演算強度が大きくなるため、実行効率が高くなったと言える。この複素数問題の場合について、京、FX10、FX100 で比較した結果を 7 ページに示す。これより、京や FX10 に比べて FX100 での実行効率が大きく下がっていることが分かる。この点について考察する。FX100 はノードあたり 32 コアで京の 8 コアや FX10 の 16 コアに比べて増加しているが、FX100 では Core Memory Group（CMG）あたり MPI プロセスを 1 つ起動させているため、プロセスあたりのコア数は FX10 と同数であり、8 ページに示すようにスレッド並列効率の結果も FX10 と同程度

であった。そこで SIMD 率（11 ページ）を調べたところ、FX10 に比べて半分以下になっており、FX100 の 4way SIMD を十分に活かせていないことが分かった。この点は今後改善する必要がある。

また、三吉委員より、9 ページの高コスト関数リストで「__lll_lock_wait」や「__pthread_mutex_unlock_usercnt」が上位にきており、メモリ獲得をマスタースレッド以外でも頻繁に行っているプログラムであれば、「XOS_MMM_L_ARENA_LOCK_TYPE=0」の環境変数を指定するよう助言を頂いた。その結果を次表に示す。表より、同環境変数の指定が有効であることが分かった。

表 1 時間調和渦電流問題における実行効率[%]の比較

	FX100
as is	1.64
+realloc 呼び出し削減	2.43
環境変数指定	2.95

2.7.5 まとめ

今回の測定で得られた知見は以下の通りである。

a) #include 指示行で取り込むファイルに対しては、コンパイラによる最適化情報付きソースリストは出力されない。

b) 複素数の疎行列ベクトル積が主な計算となる領域分割法において、京や FX10 に対し FX100 では実行効率が半分以下に低下した。SIMD 率も半分以下に低下しており、SIMD 幅増加に対して複素数四則演算の効率的な実装が必要となる。

c) スレッド並列処理時にマスタースレッド以外でもメモリ確保・解放を繰り返す領域分割法コードでは、”XOS_MMM_L_ARENA_LOCK_TYPE=0”の指定が有効である。

アプリ性能評価
電磁場解析「ADVENTURE_Magnetic」

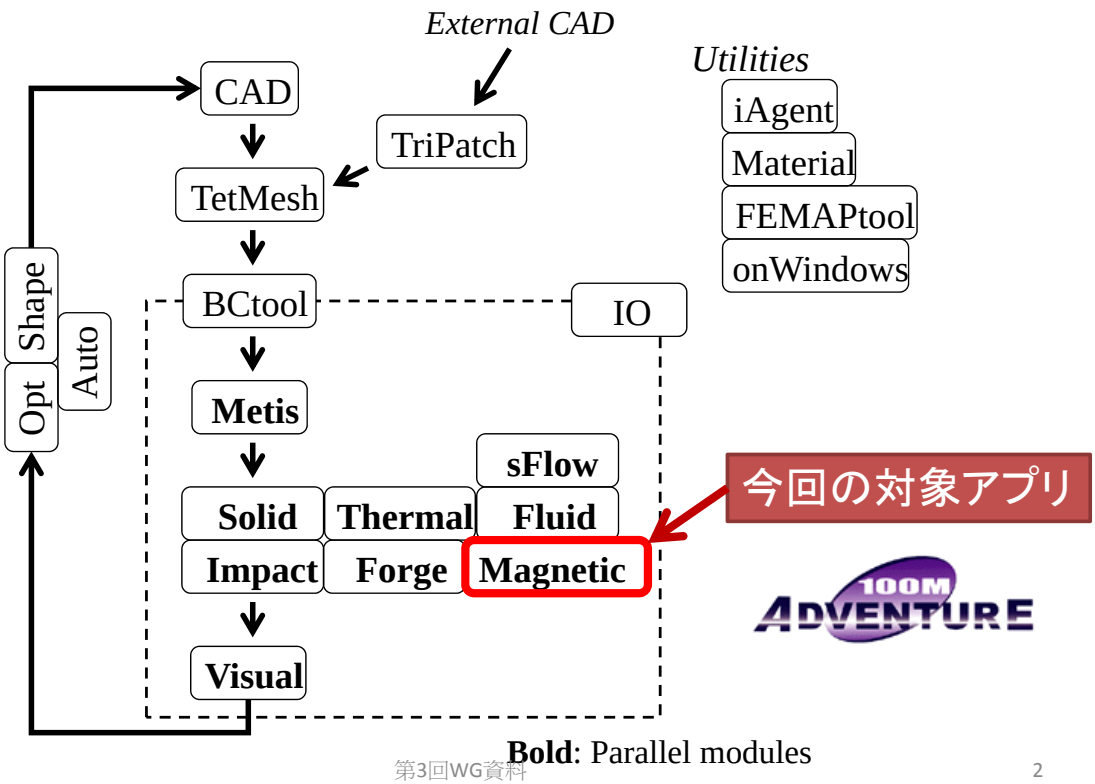
荻野正雄 (名古屋大学)

2014年10月1日

第3回SS研ポストペタアプリ性能WG

ADVENTUREシステム

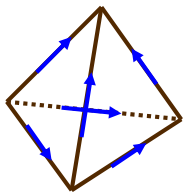
http://adventure.sys.t.u-tokyo.ac.jp/



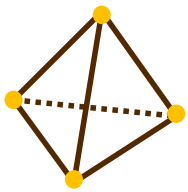
ADVENTURE_Magneticの概要

- 電磁場解析ADVENTURE_Magnetic
  - 最新バージョン: 1.5 (2014年3月19日公開)
  - 有限要素法(FEM)による電磁場解析
    - 非線形静磁場問題, 時間調和渦電流問題, 非定常渦電流問題
    - *A*法または*A*-ϕ法による定式化
    - 陰解法
    - Nédélec四面体1次要素 (ベクトル関数を要素辺上で評価)
  - 領域分割法(DDM)
    - Glowinski '84に基づく定式化
    - 領域分割数N >= プロセス数N_p
    - 計算ノード内は部分領域ループをスレッド並列処理
  - 言語
    - C, OpenMP, MPI

定式化



*A*はNédélec四面体1次



ϕは四面体1次

- 静磁場問題 実数, 対称

$$(\nu \operatorname{rot} A_h, \operatorname{rot} A_h^*) = (\tilde{J}_h, A_h^*)$$

- 時間調和渦電流問題 複素数, 対称

$$(\nu \operatorname{rot} A_h, \operatorname{rot} A_h^*) - (i\omega\sigma\, A_h, A_h^*) + (\sigma \operatorname{grad} \phi_h, A_h^*) = (\tilde{J}_h, A_h^*)$$
$$(\sigma \operatorname{grad} \phi_h, \operatorname{grad} \phi_h^*) - (i\omega\sigma\, A_h, \operatorname{grad} \phi_h^*) = 0$$

- 非定常渦電流問題 実数, 対称

$$(\nu \operatorname{rot} A_h^{n+1}, \operatorname{rot} A_h^*) + \left(\frac{\sigma}{\Delta t} A_h^{n+1}, A_h^*\right) + (\sigma \operatorname{grad} \phi_h^{n+1}, A_h^*) = (\tilde{J}_h^{n+1}, A_h^*) + \left(\frac{\sigma}{\Delta t} A_h^n, A_h^*\right)$$
$$(\sigma A_h^{n+1}, \operatorname{grad} \phi_h^*) + \Delta t(\sigma \operatorname{grad} \phi_h^{n+1}, \operatorname{grad} \phi_h^*) = (\sigma A_h^n, \operatorname{grad} \phi_h^*)$$

A: 磁気ベクトルポテンシャル[Wb/m], ϕ: 電気スカラーポテンシャル[V],
ν: 磁気抵抗率[m/H], J: 電流密度[A/m^2], i: 虚数単位, ω: 角周波数[rad/s],
σ: 導電率[S/m] (不導体領域では$\sigma = 0$), Δt: 時間刻み幅[s]

領域分割法(DDM)

- 線形化された方程式

$$\mathbf{Kx} = \mathbf{b}$$

- Non-overlapping領域分割

$$\begin{bmatrix} \mathbf{K}_{\mathrm{II}} & \mathbf{K}_{\mathrm{IB}} \\ \mathbf{K}_{\mathrm{BI}} & \mathbf{K}_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathrm{I}} \\ \mathbf{x}_{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathrm{I}} \\ \mathbf{b}_{\mathrm{B}} \end{bmatrix}$$

- Schur補元方程式

$$\mathbf{Sx}_{\mathrm{B}} = \mathbf{g}$$

Krylov部分空間法 (CG, COCG)で解く

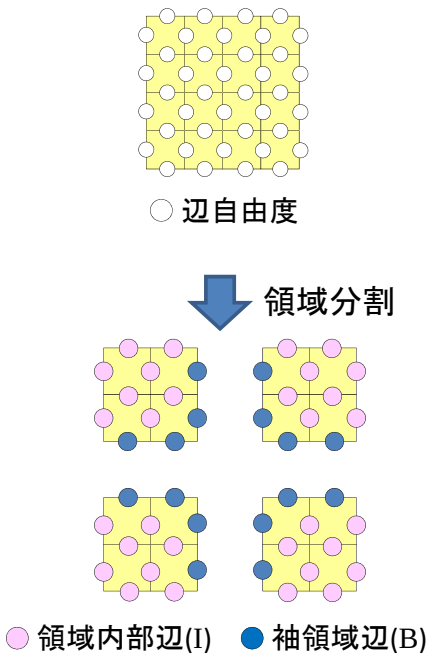
ただし,

$$\mathbf{S} = \mathbf{K}_{\mathrm{BB}} - \mathbf{K}_{\mathrm{BI}}(\mathbf{K}_{\mathrm{II}})^{\dagger}\mathbf{K}_{\mathrm{IB}},$$
$$\mathbf{g} = \mathbf{b}_{\mathrm{B}} - \mathbf{K}_{\mathrm{BI}}(\mathbf{K}_{\mathrm{II}})^{\dagger}\mathbf{b}_{\mathrm{I}}$$

または部分領域ごとの項を用いて,

$$\mathbf{S} = \sum_{\mathrm{i}=1}^{N} \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})\mathrm{T}}\mathbf{S}^{(\mathrm{i})}\mathbf{R}_{\mathrm{B}}^{(\mathrm{i})}, \qquad \mathbf{g} = \sum_{\mathrm{i}=1}^{N} \mathbf{R}_{\mathrm{B}}^{(\mathrm{i})\mathrm{T}}\mathbf{g}^{(\mathrm{i})},$$

$$\mathbf{S}^{(\mathrm{i})} = \mathbf{K}_{\mathrm{BB}}^{(\mathrm{i})} - \mathbf{K}_{\mathrm{BI}}^{(\mathrm{i})}\left(\mathbf{K}_{\mathrm{II}}^{(\mathrm{i})}\right)^{\dagger}\mathbf{K}_{\mathrm{IB}}^{(\mathrm{i})}, \quad \mathbf{g}^{(\mathrm{i})} = \mathbf{b}_{\mathrm{B}}^{(\mathrm{i})} - \mathbf{K}_{\mathrm{BI}}^{(\mathrm{i})}\left(\mathbf{K}_{\mathrm{II}}^{(\mathrm{i})}\right)^{\dagger}\mathbf{b}_{\mathrm{I}}^{(\mathrm{i})}$$



DDMの実装例

- 一般的な実装 (ADVENTUREなど)
  - Schur補行列を陰に構築

    行列ベクトル積演算 $\mathbf{q} \leftarrow \left(\mathbf{K}_{\mathrm{BB}} - \mathbf{K}_{\mathrm{BI}}(\mathbf{K}_{\mathrm{II}})^{\dagger}\mathbf{K}_{\mathrm{IB}}\right)\mathbf{p}$

- 河合らによる実装 (ADV-K, '12)
  - 部分領域(i)のローカルSchur補行列を陽に構築

    行列ベクトル積演算 $\mathbf{q} \leftarrow \sum_{\mathrm{i}=1}^{N} \mathbf{R}^{(\mathrm{i})\mathrm{T}}\mathbf{S}^{(\mathrm{i})}\mathbf{R}^{(\mathrm{i})}\,\mathbf{p}$

全体行列の成分が不明のため, 評価できる反復法・前処理法が限定される

- 屋・荻野による実装 (LexADV_TryDDM, '13)
  - 全体行列を陽に構築

    行列ベクトル積演算 $\mathbf{q} \leftarrow \mathbf{Sp}$

ほとんどの反復法・前処理法が評価可能で, 新しい前処理の開発コストも低い

ADVENTUREにおけるコア部分

$$
\begin{aligned}
&\textbf{for } p = 1 \ to \ N_p \textbf{ do} \\
&\quad \textbf{for } i = 1 \ to \ N_{dom}^{(p)} \textbf{ do} \\
&\quad\quad \mathbf{s}_{\mathrm{I}}^{(\mathrm{p,i})} \leftarrow -\mathbf{K}_{\mathrm{IB}}^{(\mathrm{p,i})}\mathbf{p}_{\mathrm{B}}^{(\mathrm{p,i})} \quad ① \\
&\quad\quad \text{solve } \mathbf{K}_{\mathrm{II}}^{(\mathrm{p,i})}\mathbf{t}_{\mathrm{I}}^{(\mathrm{p,i})} = \mathbf{s}_{\mathrm{I}}^{(\mathrm{p,i})} \quad ② \\
&\quad\quad \mathbf{q}_{\mathrm{B}}^{(\mathrm{p,i})} \leftarrow \mathbf{K}_{\mathrm{BB}}^{(\mathrm{p,i})}\mathbf{p}_{\mathrm{B}}^{(\mathrm{p,i})} + \mathbf{K}_{\mathrm{BI}}^{(\mathrm{p,i})}\mathbf{t}_{\mathrm{I}}^{(\mathrm{p,i})} \quad ③ \\
&\quad\quad \mathbf{q}_{\mathrm{B}}^{(\mathrm{p})} \leftarrow \mathbf{q}_{\mathrm{B}}^{(\mathrm{p})} + \mathbf{R}_{\mathrm{B}}^{(\mathrm{p,i})}\mathbf{q}_{\mathrm{B}}^{(\mathrm{p,i})} \\
&\quad \textbf{end for} \\
&\quad \mathbf{q}_{\mathrm{B}} \leftarrow \mathbf{q}_{\mathrm{B}} + \mathbf{q}_{\mathrm{B}}^{(\mathrm{p})} \\
&\textbf{end for}
\end{aligned}
$$

Schur補行列-ベクトル積演算

- 部分領域ループ内
  - ① 疎行列ベクトル積
  - ② ICCG/ICCOCG法で解く
    - ②－1 疎行列ベクトル積
    - ②－2 前進後退代入
  - ③ 疎行列ベクトル積

ADVENTURE_Magneticの特徴

- コーディング指針
  - 静磁場・渦電流(2種)でFEMやDDM等のコードを共通化
  - 実数・複素数で線形代数関係のコードを共通化

共通コード＋異なるマクロ

```
#define _real_
#include "common.c"
```

```
#define _complex_
#include "common.c"
```

コンパイラによる最適化情報付きソースリストが出力されない箇所が多い

コード例(DAXPY)

```
3017          for( i=0 ; i<nf ; i++ ) {
3018            MTYPE  dd ;
3019    8       MMULTI( dd, alpha, aq[i] ) ;
3020    8       MPLUS( r[i], r[i], dd ) ;
3021    8     }
```

SIMD化されない原因が分からないことが多い…

数値実験

- テスト問題
  - 時間調和／非定常渦電流
  - 無限長ソレノイドコイル
    → 長さ0.1m, 中心角20°モデル
  - 100万(複素)自由度メッシュ
- 使用計算機
  - 名古屋大学Fujitsu PRIMEHPC FX10
    12ノード
  - コンパイラオプション
    "-Kfast,ocl,restp=arg,simd=2,openmp"
    (名大標準) "-Kfast -g -Ntl_trt -Xa -NRtrap"

"-ipo"を指定しようとしたが,名大標準で"-g"があるので無効になってしまった

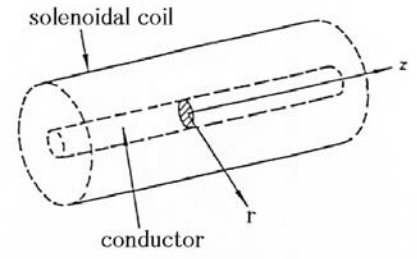


図 22 無限長ソレノイドコイル

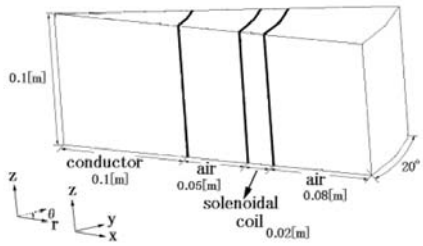
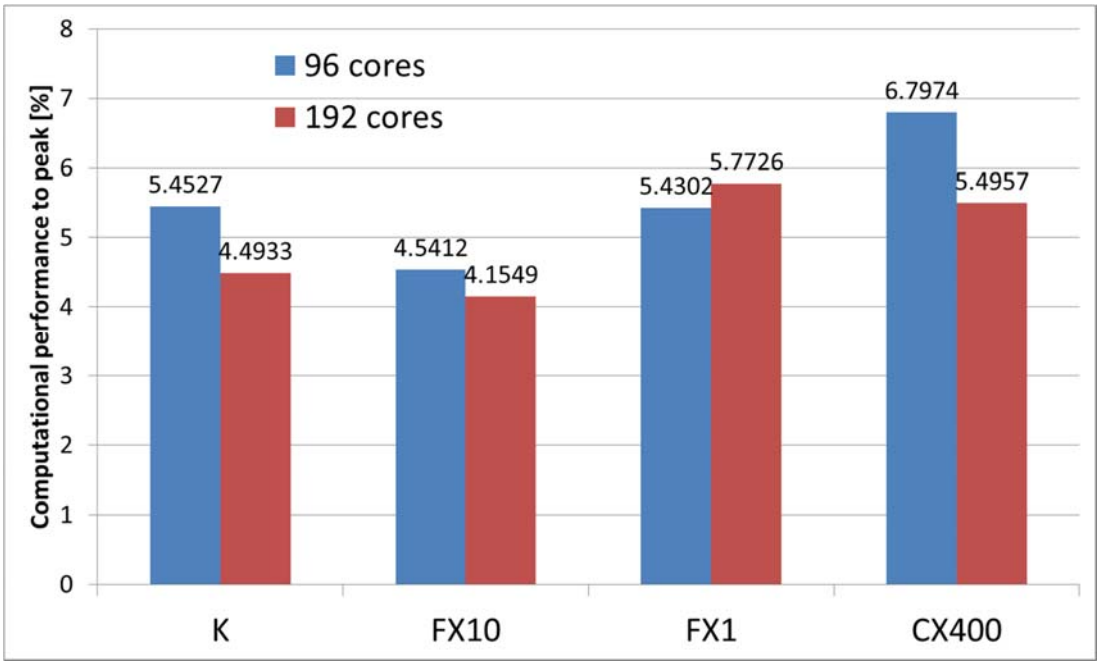


図 23 ケーキモデル

ADVENTURE_Solidのピーク性能比

第9回マルチコアWG時の資料



(# Nodes) (12) (24) (6) (12) (24) (48) (6) (12)

"AS IS"コードのピーク性能比

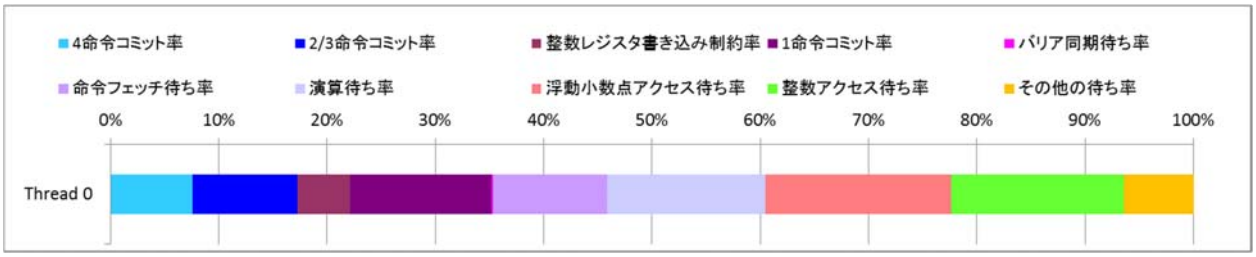
処理	OpenMP並列化ループ	Time-Harmonic (Complex) [%]	Non-Steady (Real) [%]
all	-	2.17	0.43
SolveDom	内	3.02	0.67
LoopPart	外	3.61	0.72
SolverCG ②式のCG法計算	内	3.55	0.87
mymtMxV 疎行列ベクトル積	内	3.46	1.12
mymtSetBC 境界条件処理	内	8.54	0.20
mymtMakeICC, IC分解	内	0.32	0.27
mymtSetICC, IC前処理	内	3.68	2.01
RealInner, ドット積	内外	0.00	0.45

簡単な改良後のピーク性能比

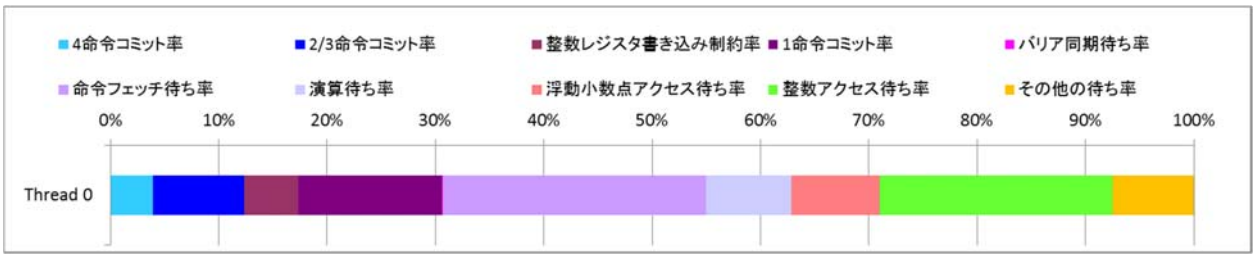
処理	Time-Harmonic (Complex) [%]		Non-Steady (Real) [%]	
	AS IS	TUNED	AS IS	TUNED
all	2.17	2.60	0.43	0.50
SolveDom	3.02	4.09	0.67	0.77
LoopPart	3.61	5.07	0.72	0.83
SolverCG	3.55	3.71	0.87	0.90
mymtMxV	3.46	3.77	1.12	1.19
mymtSetBC	8.54	8.40	0.20	0.21
mymtMakeICC	0.32	0.23	0.27	0.26
mymtSetICC	3.68	3.88	2.01	2.10
RealInner	0.00	0.00	0.45	0.46

精密PA結果: SolveDom, 実行時間内訳

時間調和渦電流(複素数)の実行時間内訳

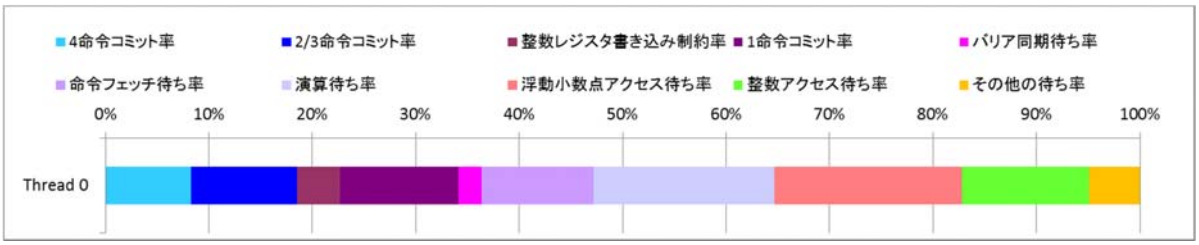


非定常渦電流(実数)の実行時間内訳

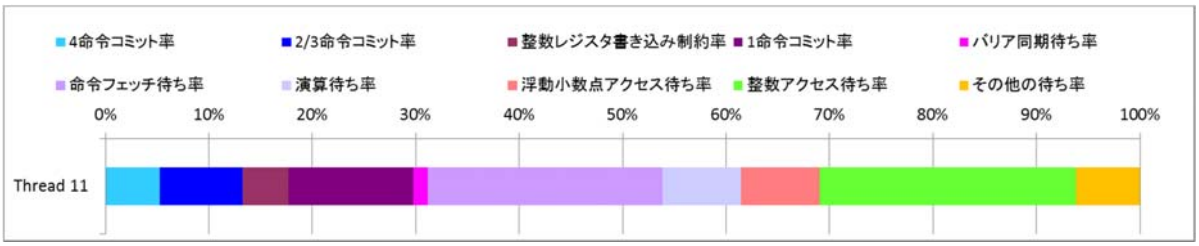


精密PA結果: LoopPart, 実行時間内訳

時間調和渦電流(複素数)の実行時間内訳

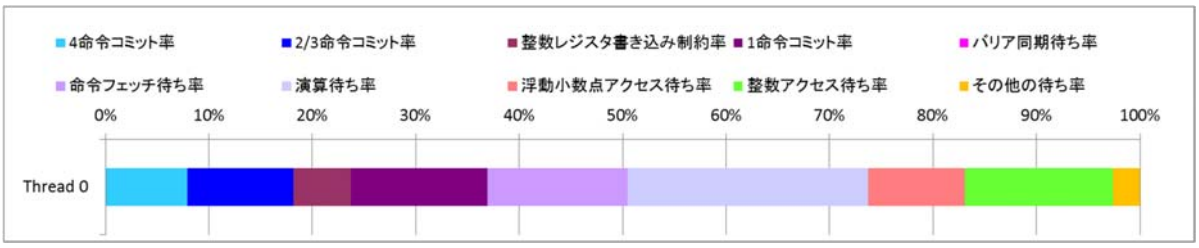


非定常渦電流(実数)の実行時間内訳

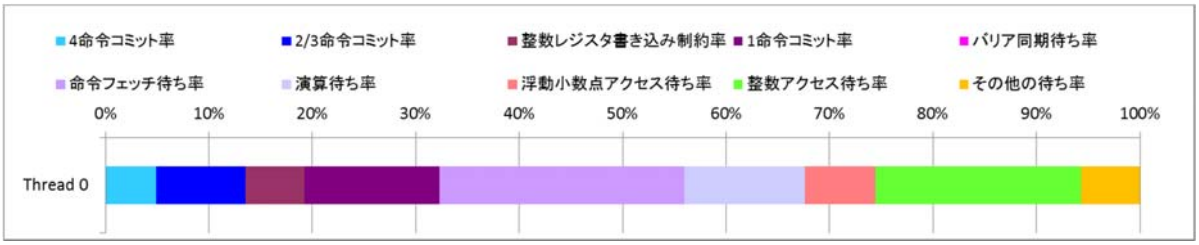


精密PA結果: SolverCG, 実行時間内訳

時間調和渦電流(複素数)の実行時間内訳

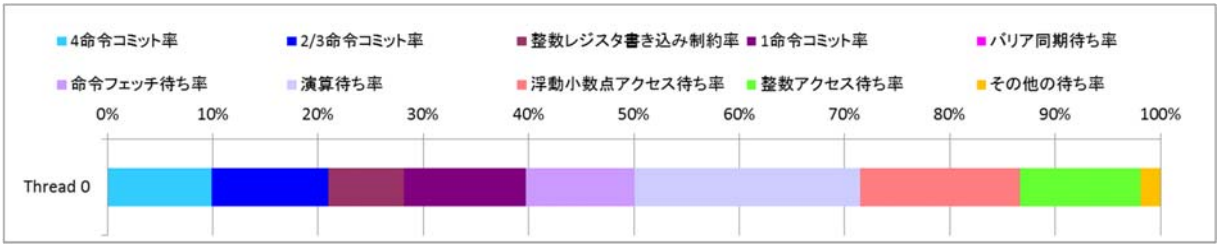


非定常渦電流(実数)の実行時間内訳

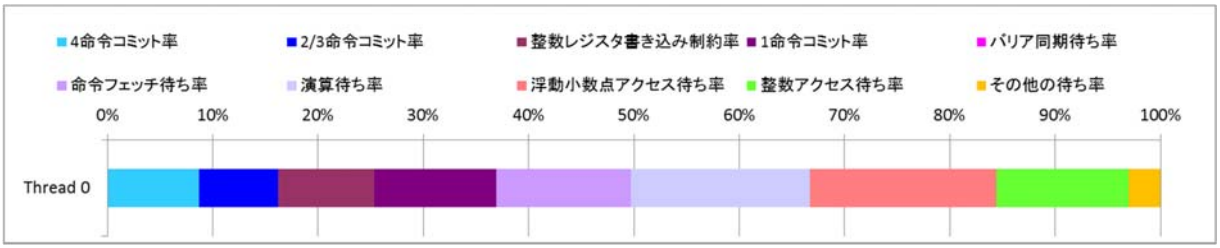


精密PA結果: mymtMxV, 実行時間内訳

時間調和渦電流(複素数)の実行時間内訳

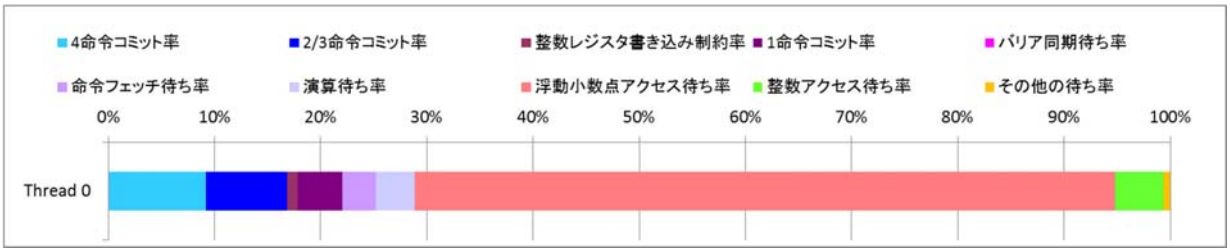


非定常渦電流(実数)の実行時間内訳

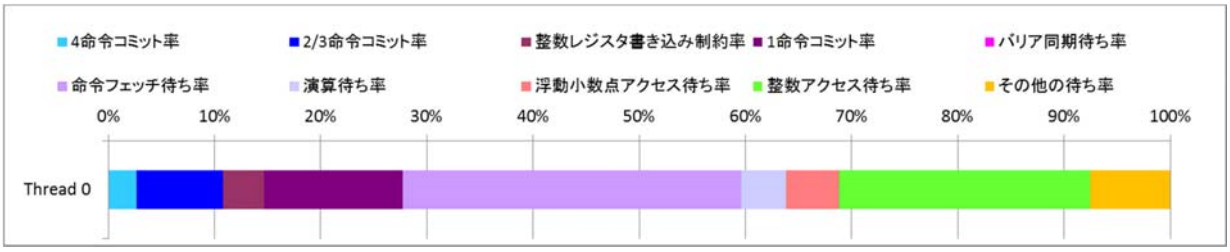


精密PA結果: mymtSetBC, 実行時間内訳

時間調和渦電流(複素数)の実行時間内訳

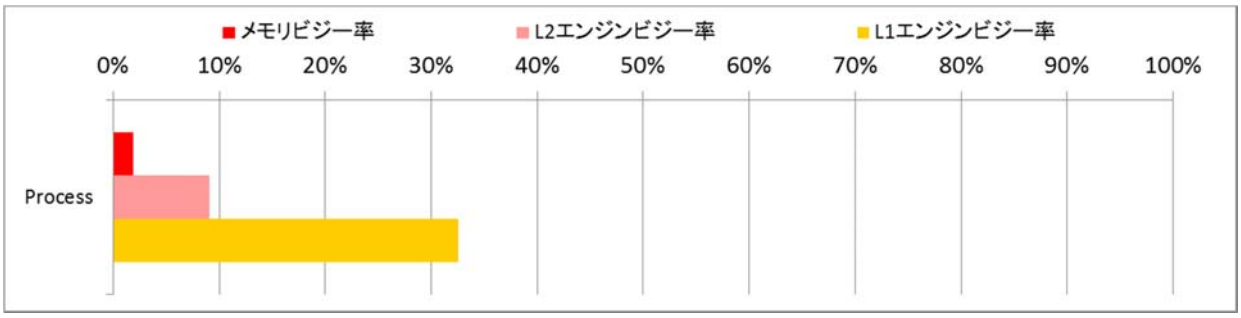


非定常渦電流(実数)の実行時間内訳

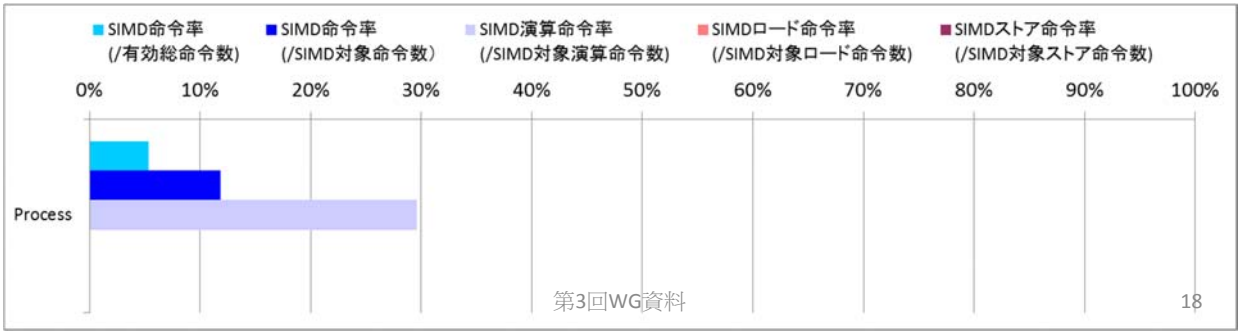


精密PA結果: mymtMxV (1)

メッシュ・キャッシュエンジン　ビジー率



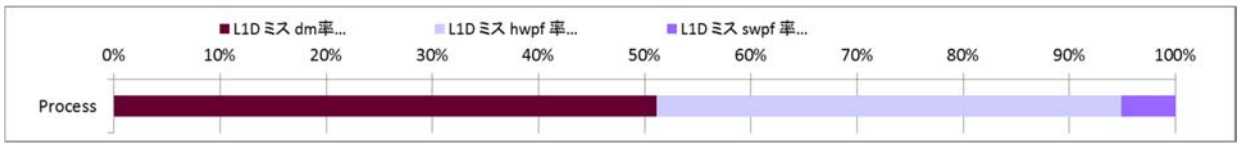
SIMD命令率



精密PA結果: mymtMxV (2)

キャッシュミス率(/ロード・ストア数)

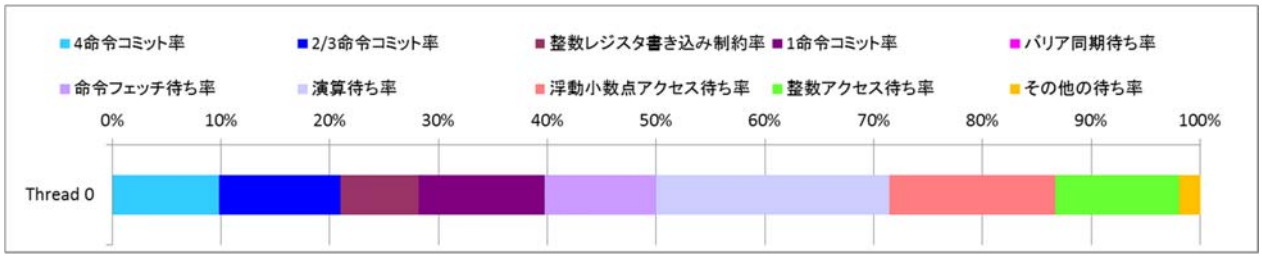
L1Dキャッシュミス内訳(/L1Dキャッシュミス数)



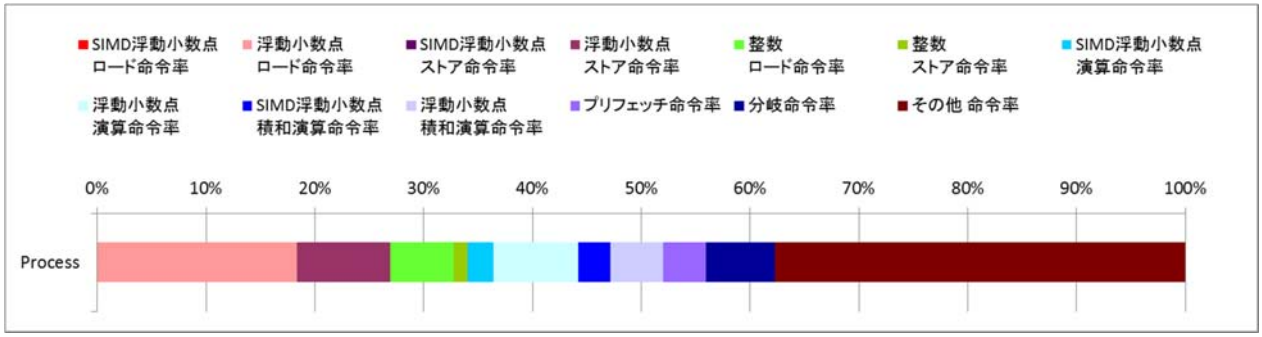
L2キャッシュミス内訳(/L2キャッシュミス数)

精密PA結果: mymtMxV (3)

実行時間内訳 [概略] (/実行時間)



命令内訳 (/有効総命令数)



電磁場解析ADVENTURE_MagneticのFX100性能評価

荻野正雄 (名古屋大学)

2015年10月8日

第7回SS研ポストペタアプリ性能WG

AdvMagneticにおける実装の特徴

- コーディング指針
  - 静磁場・渦電流(2種)でFEMやDDM等のコードを共通化
  - 実数・複素数で線形代数関係のコードを共通化

マクロ定義+共通コード

hddm_solver_rs.c

```
#define __ADVMAG_MATRIX_SYSTEM_COMPLEX__

#include "mtype.h"
#include "hddm_solver.c"
```

hddm_solver_cs.c

```
#define __ADVMAG_MATRIX_SYSTEM_REAL__

#include "mtype.h"
#include "hddm_solver.c"
```

プログラム実体であるhddm_solver.cをincldue文で取り込んでおり, それらはコンパイラによる最適化情報付きソースリストが出力されない

必要に応じて, 展開してコンパイルしてみて最適化情報を確認

四則演算のコーディング方法

for j = ai[i] to ii **do**
  x[i] += vij[j] × vec[aij[j]]
  x[aij[j]] += vij[j] × vec[i]
endfor

```
for( j=ai[i] ; j<ii ; j++ ) {
   MTYPE  ddd;
   MMULTI( ddd, vij[j], vec[aij[j]] );
   MPLUS( x[i], x[i], ddd );
   MMULTI( ddd, vij[j], vec[i] );
   MPLUS( x[aij[j]], x[aij[j]], ddd );
}
```

- AdvMagneticの実装
  - 実数
    - 関数形式マクロ
  - 複素数
    - 構造体(AOS) + 関数
    - 構造体(AOS) + 関数形式マクロ
    - _Complex型

```
 v for( j=ai[i] ; j<ii ; j++ ) {
 v   MTYPE  ddd;
6v   MMULTI( ddd, vij[j], vec[aij[j]] );
6v   MPLUS( x[i], x[i], ddd );
6v   MMULTI( ddd, vij[j], vec[i] );
6v   MPLUS( x[aij[j]], x[aij[j]], ddd );
6v }
```

```
 v for( j=ai[i] ; j<ii ; j++ ) {
 v   MTYPE  ddd;
2v   MMULTI( ddd, vij[j], vec[aij[j]] );
2v   MPLUS( x[i], x[i], ddd );
2v   MMULTI( ddd, vij[j], vec[i] );
2v   MPLUS( x[aij[j]], x[aij[j]], ddd );
2v }
```

#pragma loop norecurrenceを指定する(第6回富士通報告より)ことで, 実数/複素数ともにSIMD化が適用された

並列計算方法と行列サイズ

for $p = 1\ to\ N_p$ **do**

for $k = 1\ to\ N_{dom}^{(p)}$ **do**

$\mathbf{s}_{\mathrm{I}}^{(p,k)} \leftarrow -\mathbf{K}_{\mathrm{IB}}^{(p,k)}\mathbf{p}_{\mathrm{B}}^{(p,k)}$

solve $\mathbf{K}_{\mathrm{II}}^{(p,k)}\mathbf{t}_{\mathrm{I}}^{(p,k)} = \mathbf{s}_{\mathrm{I}}^{(p,k)}$

$\mathbf{q}_{\mathrm{B}}^{(p,k)} \leftarrow \mathbf{K}_{\mathrm{BB}}^{(p,k)}\mathbf{p}_{\mathrm{B}}^{(p,k)} + \mathbf{K}_{\mathrm{BI}}^{(p,k)}\mathbf{t}_{\mathrm{I}}^{(p,k)}$

$\mathbf{q}_{\mathrm{B}}^{(p)} \leftarrow \mathbf{q}_{\mathrm{B}}^{(p)} + \mathbf{R}_{\mathrm{B}}^{(p,k)}\mathbf{q}_{\mathrm{B}}^{(p,k)}$

end for

$\mathbf{q}_{\mathrm{B}} \leftarrow \mathbf{q}_{\mathrm{B}} + \mathbf{q}_{\mathrm{B}}^{(p)}$

end for

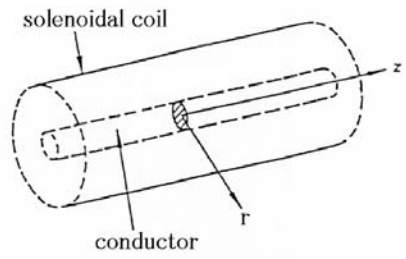
MPI並列化 (MPIプロセスに1つずつ割当)

OpenMP並列化 (スレッドに複数領域を割当)

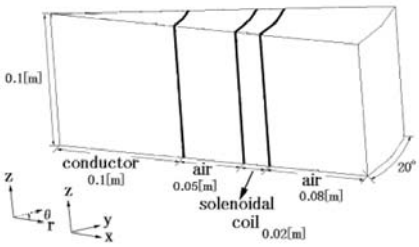
今回は行列サイズは
100～200程度となるように
領域分割

数値実験

- テスト問題
  - 時間調和／非定常渦電流
  - 無限長ソレノイドコイル
    → 長さ0.1m, 中心角20°　モデル
  - 400万(複素)自由度メッシュ
  - 領域分割数 30,720
- 使用計算機
  - 名大FX100　4ノード(4,505 Gflop/s)
    - コンパイラオプション
      "-Kfast,ocl,restp=arg,preex,array_private -Kopenmp"
  - 東大FX10　4ノード(946 Gflop/s)
  - 理研AICS京　4ノード(512 Gflop/s)



無限長ソレノイドコイル



ケーキモデル

時間調和問題(複素数)と非定常問題(実数)の比較 @ FX100

FLOPS / PEAK [%]

	Time-Harmonic (Complex)	Non-Steady (Real)
asis	1.6381	0.6962
ocl指定	1.6916	0.7256
+一部のループ入れ替え	2.3759	0.8755
+realloc呼び出し回数削減	2.4354	0.9303

複素数問題は実数問題に比べて,
例えば疎行列(CSR)ベクトル積では,
・データ読み書き量は2倍
・演算量は4倍

```
for j = ai[i] to ii do
  x[i] += vij[j] × vec[aij[j]]
  x[aij[j]] += vij[j] × vec[i]
endfor
```

時間調和問題(複素数)における K, FX10とFX100の比較

	K		FX10		FX100	
	Time [s] (ratio to FX100)	FLOPS /PEAK [%]	Time [s] (ratio to FX100)	FLOPS /PEAK [%]	Time [s]	FLOPS /PEAK [%]
asis	626 (3.35)	5.2819	424 (2.27)	4.2196	187	1.6381
ocl指定	604 (3.34)	5.4699	412 (2.28)	4.3455	181	1.6916
+一部のループ入れ替え	461 (3.60)	7.1466	320 (2.5)	5.5657	128	2.3759
+realloc呼び出し回数削減	441 (3.5)	7.4700	298 (2.37)	5.9848	126	2.4354

KやFX10に比べてFX100での実行効率が大きく下がっている

Scalability of multithreads @ FX100

# of threads / CMG	Time [s]	Speedup ratio	FLOPS/PEAK [%]
1	1,061	1.00	3.6832
2	623	1.70	3.1394
4	324	3.27	3.0199
8	188	5.64	2.5990
16	123	8.63	2.0123

CMGあたりのスレッド並列効率は54%くらいと高くはないが,
この傾向はFX10でも同じ

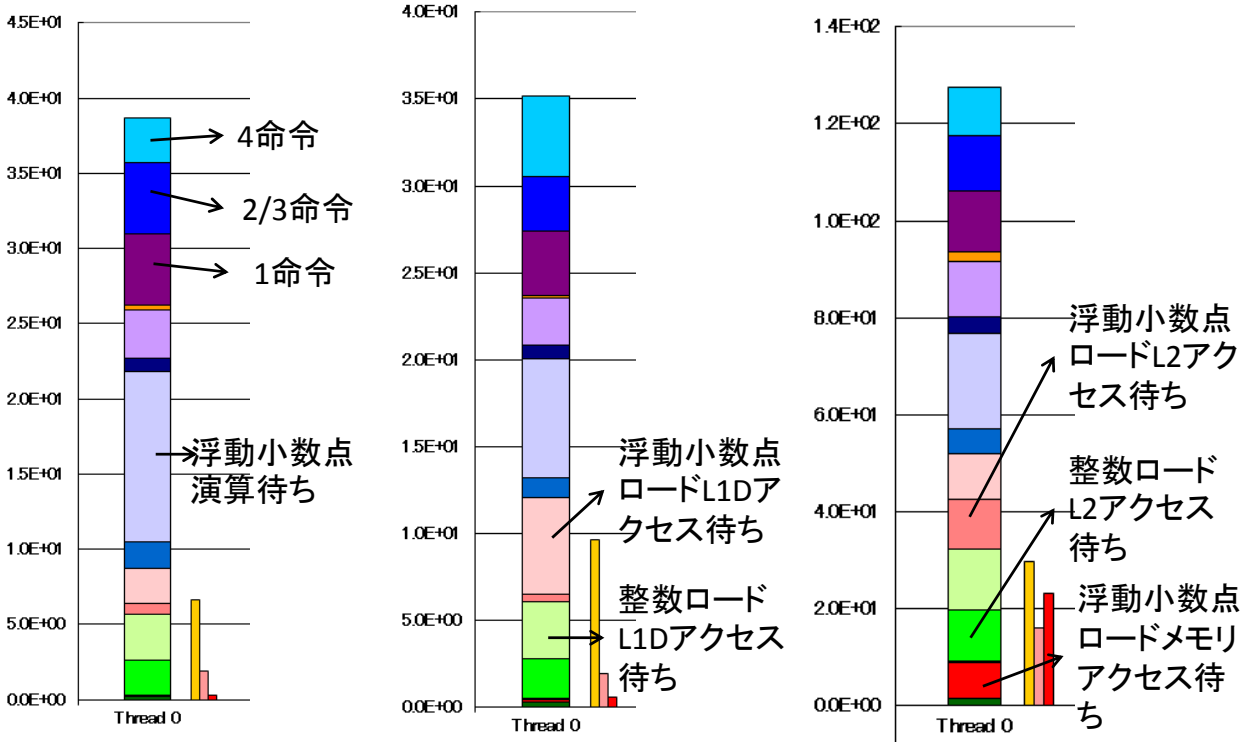
時間調和問題(複素数)における高コスト関数 @ FX100

```
*******************************************************************************
      Application - procedures
  *******************************************************************************

            Cost          %  Operation (S)
  -------------------------------------------------------------------------------
          122204   100.0000     12329.8926          Application
  -------------------------------------------------------------------------------
           32830    26.8649      3312.3438          mymt_CS_AIJ_PC_ICC_set
           27776    22.7292      2802.4683          mymt_CS_AIJ_MxV
           16841    13.7811      1699.4738          hddm_solver_CS_solve_dom
            9309     7.6176       939.2852          hddm_solver_CS_Parent_loop_part._OMP_2
            7041     5.7617       710.4250          __lll_lock_wait
            7040     5.7609       710.2708          mymt_CS_AIJ_SetBC_with_bcMat
            6136     5.0211       619.0325          __pthread_mutex_unlock_usercnt
            3686     3.0163       371.9494          hddm_solver_CS_renewal_bc_sp
            3680     3.0114       371.2768          mymt_CS_AIJ_Solver_COCG_func
             637     0.5213        64.2622          COM_WriteCpl
```

このリストはK, FX10, FX100で大差はない

高コスト関数上位3傑の精密プロファイラ情報



SetICC　　MxV　　SolveDom

全体及び高コスト関数のSIMD率

SIMD [%]

	FX10	FX100
all	38.7176	16.7654
SetICC	36.2900	15.1750
MxV	24.0306	6.6256
SolveDom	17.9075	7.5875