CMSI計算科学技術 特論B

第5回
アプリケーションの性能最適化の実例2

2014年5月15日

独立行政法人理化学研究所
計算科学研究機構 運用技術部門
ソフトウェア技術チーム チームヘッド

南 一生
minami_kaz@riken.jp

K computer

RIKEN ADVANCED INSTITUTE FOR COMPUTATIONAL SCIENCE

講義の概要

- スーパーコンピュータとアプリケーションの性能
- アプリケーションの性能最適化1（高並列性能最適化）
- アプリケーションの性能最適化2（CPU単体性能最適化）
- アプリケーションの性能最適化の実例1
- アプリケーションの性能最適化の実例2

内容

- 理研で進めた性能最適化
- Seism3Dの性能最適化
- FrontFlow/blueの性能最適化
- NPB MGのチューニング事例

- 本資料は，理化学研究所AICS運用技術部門ソフトウェア技術チーム，井上俊介氏(現所属富士通)，熊畑清氏の発表データを使用して作成しています．

理研で進めた性能最適化

理研で進めた性能最適化

6本のターゲットアプリ

プログラム名	分野	アプリケーション概要	期待される成果	手法
NICAM	地球科学	全球雲解像大気大循環モデル	大気大循環のエンジンとなる熱帯積雲対流活動を精緻に表現することでシミュレーションを飛躍的に進化させ，現時点では再現が難しい大気現象の解明が可能となる．(開発　東京大学,JAMSTEC,RIKEN AICS)	FDM (大気)
Seism3D	地球科学	地震波伝播・強震動シミュレーション	既存の計算機では不可能な短い周期の地震波動の解析・予測が可能となり，木造建築およびコンクリート構造物の耐震評価などに応用できる．(開発　東京大学地震研究所)	FDM (波動)
PHASE	ナノ	平面波展開第一原理電子状態解析	第一原理計算により，ポスト35nm世代ナノデバイス，非シリコン系デバイスの探索を行う．(開発　物質・材料研究機構)	平面波 DFT
FrontFlow/Blue	工学	Large Eddy Simulation (LES)に基づく非定常流体解析	LES解析により，エンジニアリング上重要な乱流境界層の挙動予測を含めた高精度な流れの予測が実現できる．(開発　東京大学生産技術研究所)	FEM (流体)
RSDFT	ナノ	実空間第一原理電子状態解析	大規模第一原理計算により，10nm以下の基本ナノ素子(量子細線，分子，電極，ゲート，基盤など)の特性解析およびデバイス開発を行う．(開発　東京大学)	実空間 DFT
LatticeQCD	物理	格子QCDシミュレーションによる素粒子・原子核研究	モンテカルロ法およびCG法により，物質と宇宙の起源を解明する．(開発　筑波大)	QCD

理研で進めた性能最適化

コラボレーション

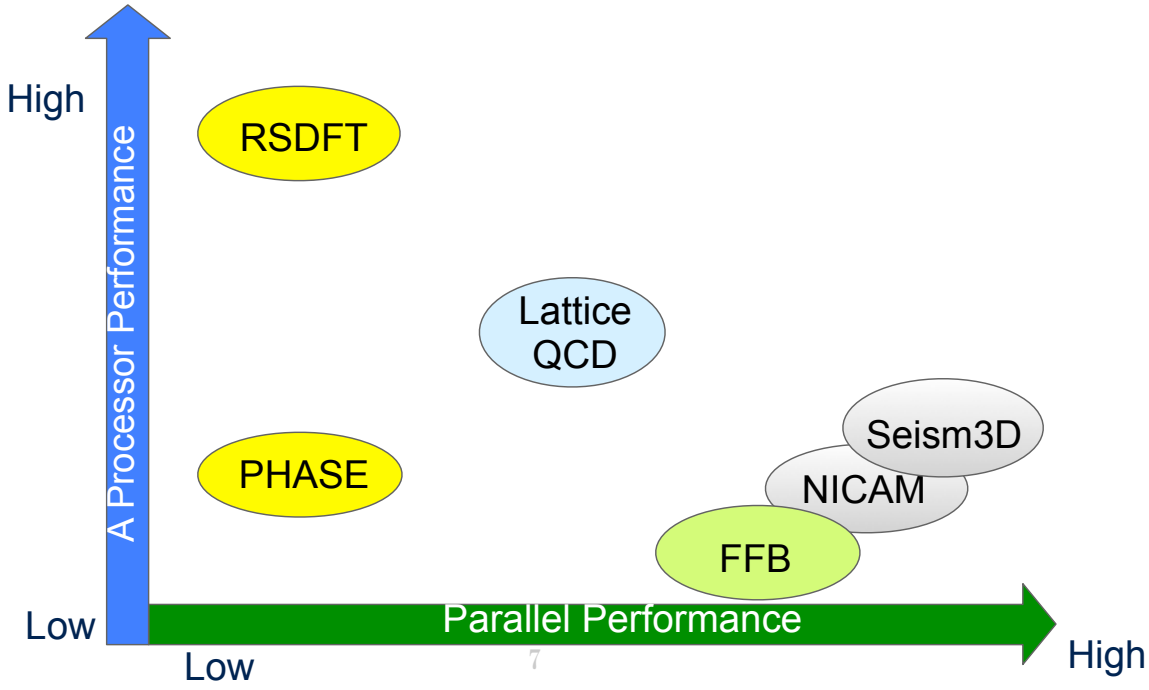
計算科学
(コード開発者)

計算機科学
(理研)

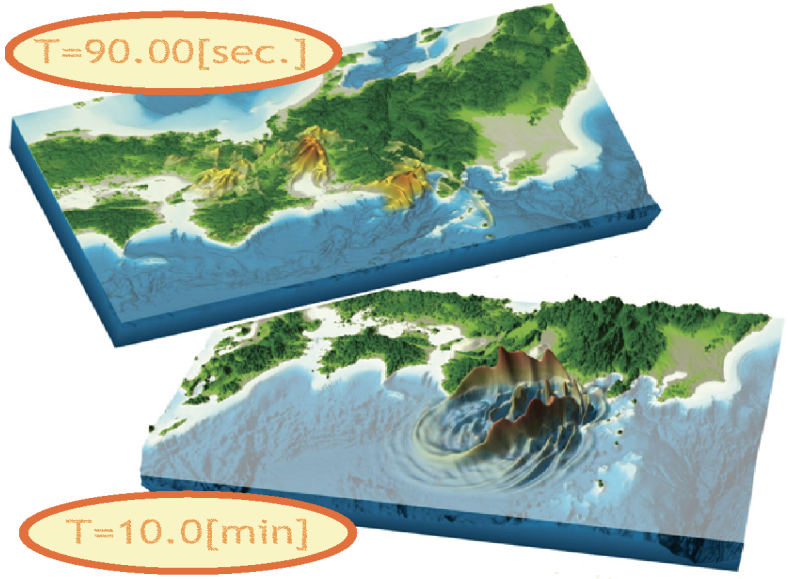
東京大学,JAMSTEC
東京大学地震研究所
物質・材料研究機構
東京大学生産技術研究所
筑波大　RIKEN AICS

理研で進めた性能最適化

6本のターゲットアプリの計算機科学的な位置づけ

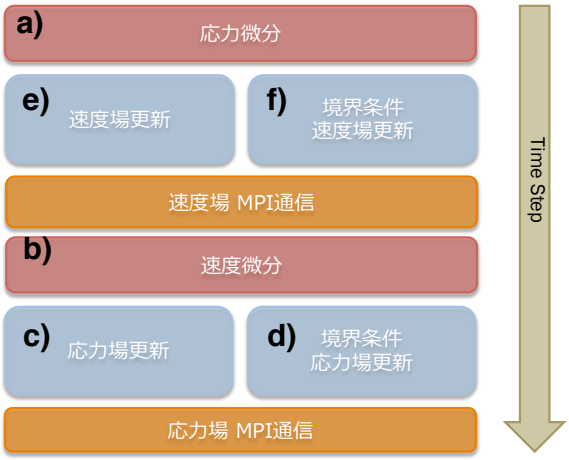


Seism3Dの性能最適化



Seism3D

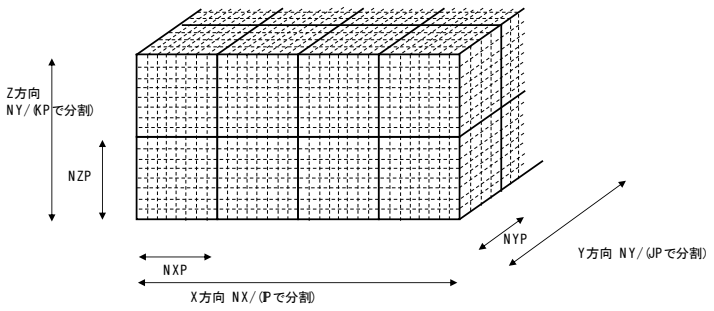
- 地震伝播を解くアプリケーション
- 有限差分法により数値的に粘弾性方程式を時間発展させる
- 現在は地震伝播と津波を連動して解く事が可能
- 大規模な並列化に対応しているアプリケーション
- 以下の6つの計算部分より構成される



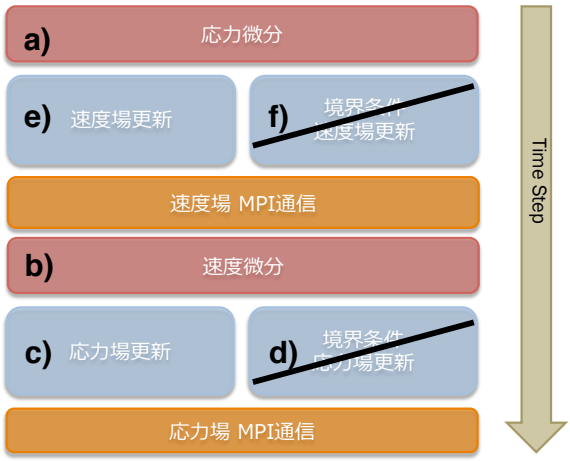
a)応力空間微分計算
b)速度空間微分計算
c)応力時間積分計算
d)応力時間積分吸収計算
e)速度時間積分計算
f) 速度時間積分吸収計算

Seism3Dの並列化

3次元の領域分割

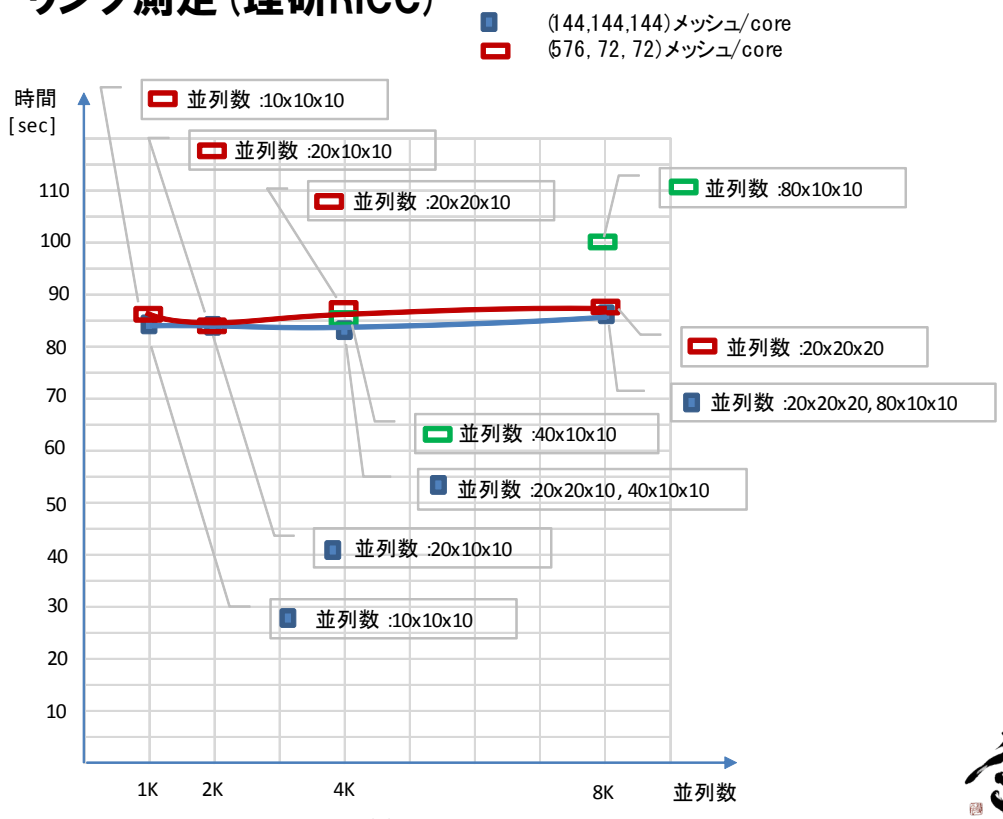


当初の評価アプリ



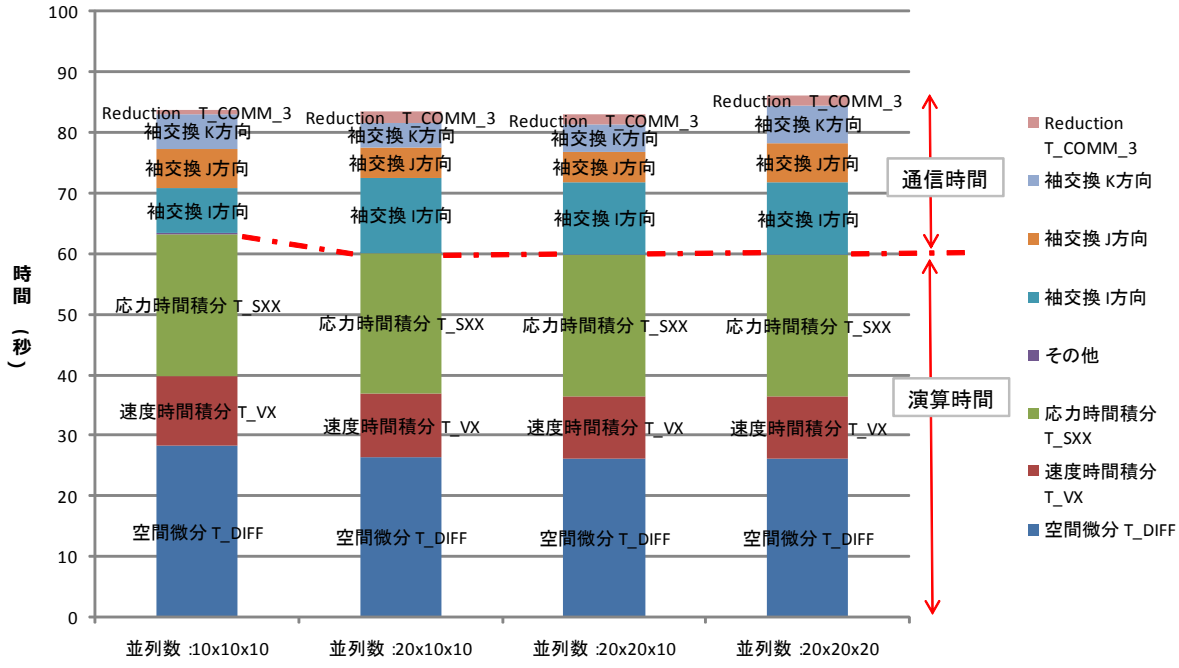
Seism3Dの並列特性分析

ウィークスケーリング測定（理研RICC）



Seism3Dの並列特性分析

ウィークスケーリング測定（理研RICC）



実行時間内訳（平均値）：(144,144,144)メッシュ/core

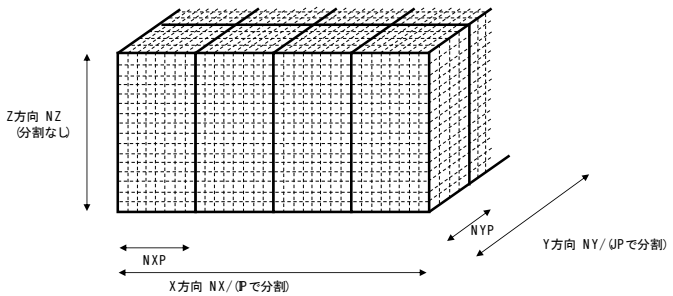
Seism3Dの並列特性分析

当初の評価アプリについて

	演算量	高並列性能
a)応力空間微分計算	O(N)	完全並列 ウィークスケールしている 隣接通信時間が通信量に比べると大 大域通信が並列数に応じ増大傾向
b)速度空間微分計算	O(N)	
c)応力時間積分計算	O(N)	
~~d)応力時間積分吸収計算~~		
e)速度時間積分計算	O(N)	
~~f) 速度時間積分吸収計算~~		

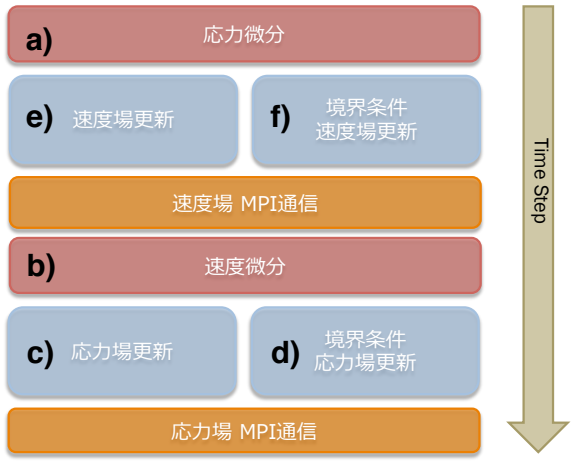
Seism3Dの並列化

2次元の領域分割



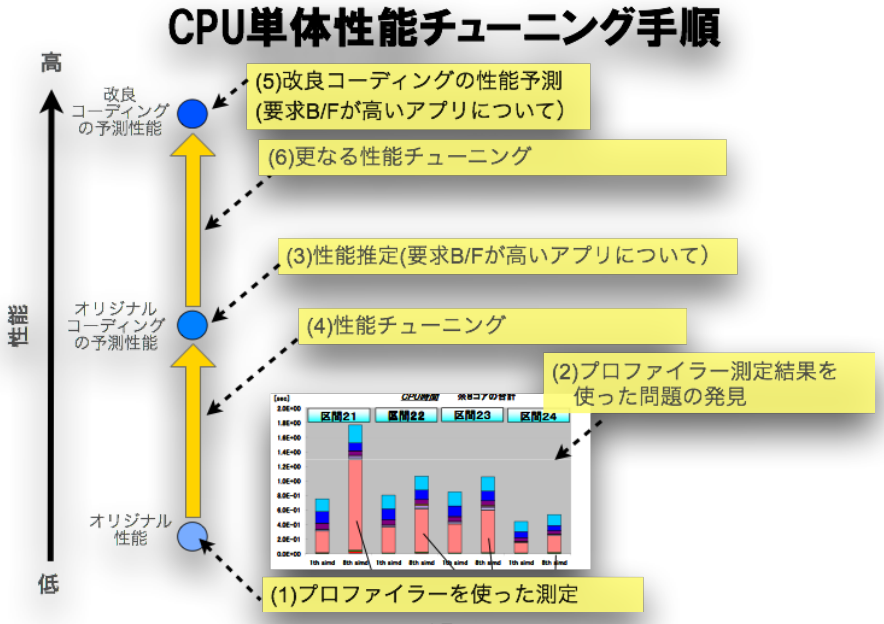
- 今後計算体系の上面のみで実行される津波の計算を鑑み２次元分割が計画された
- 2次元分割での通信時間を評価
- 通信量は4-5倍になるが通信回数は2/3になると評価
- 通信のパッキングは2次元が有利
- 通信時間は演算時間の数%ですむと評価

新しい評価アプリ



Seism3Dの単体性能最適化（性能見積りとチューニング）

Seism3Dは要求B/F値が大きいアプリケーション



空間微分Z方向の計算a) b)（1次元目の差分）

```
do J = 1, NY
   do I = 1, NX
     do K = 3, NZ-1
       DZV (k,I,J) = (V(k,I,J)  -V(k-1,I,J))*R40 &
                   - (V(k+1,I,J)-V(k-2,I,J))*R41
     end do
   end do
 end do
```

要求B/F	12/5 = 2.4
性能予測	0.36/2.4 = 0.15
実測値	0.153

要求Byteの算出:

1store,2loadと考える

4x3 = 12byte

要求flop:

add：3 mult：2 = 5

空間微分X方向の計算a) b)（2次元目の差分）

```
do J = 1, NY
  do I = 1, NX
    do K = 1, NZ
      DXV (k,I,J) = (V(k,I,J)  -V(k,I-1,J))*R40&
                  - (V(k,I+1,J)-V(k,I-2,J))*R41
    end do
  end do
end do
```

- 第2軸(I軸)が差分
- 1ストリームでその他の3配列は$L1or$L2に載っており再利用できる
- 従って1次元目が差分のパターンと同じ性能になる

要求Byteの算出:

P12より、メモリコストだけを考慮する。

1sore,2loadと考える

4x3 = 12byte

要求flop:

add：3 mult：2 = 5

要求B/F	12/5 = 2.4
性能予測	0.36/2.4 = 0.15
実測値	0.135

- 実測値が13.5%と少し低い
- 実測したメモリバンド幅は42.9GB/sec
- この値で性能予測をすると14.0%となる
- 14.0%に比較すれば13.5%は良い値
- L2キャッシュ負荷の増大によりメモリバンド幅が下がった可能性(京特有の現象)

空間微分Y方向の計算a) b)（3次元目の差分）

```
do J = 1, NY
  do I = 1, NX
    do K = 1, NZ
      DYV (k,I,J) = (V(k,I,J)  -V(k,I,J-1))*R40 &
                  - (V(k,I,J+1)-V(k,I,J-2))*R41
    end do
  end do
end do
```

- 第3軸が差分 →　再利用性なし

要求flop:

add：3 mult：2 = 5

要求Byteの算出:

1store/5loadより

(5+1) * 4byte = 24

要求B/F	24/5 = 4.8
性能予測	0.36/4.8= 0.075
実測値	0.076

空間微分Y方向の計算a) b)
(3次元目をcyclicでスレッド並列化)

```
!$OMP DO SCHEDULE(static,1),PRIVATE(I,J,K)
 do J = 1, NY
     do I = 1, NX
       do K = 1, NZ
         DYV (k,I,J) = (V(k,I,J)  -V(k,I,J-1))*R40 &
                     - (V(k,I,J+1)-V(k,I,J-2))*R41
       end do
     end do
  end do
```

キャッシュに載せる

- 第3軸をcyclic分割 → 1ストリームで3配列がL2に乗る(説明次項)
- 性能が２倍になる

要求B/F	12/5 = 2.4
性能予測	0.36/2.4 = 0.15
実測値	0.136

要求Byteの算出:

1sore,2loadと考える

4x3 = 12byte

要求flop:

add : 3 mult : 2 = 5

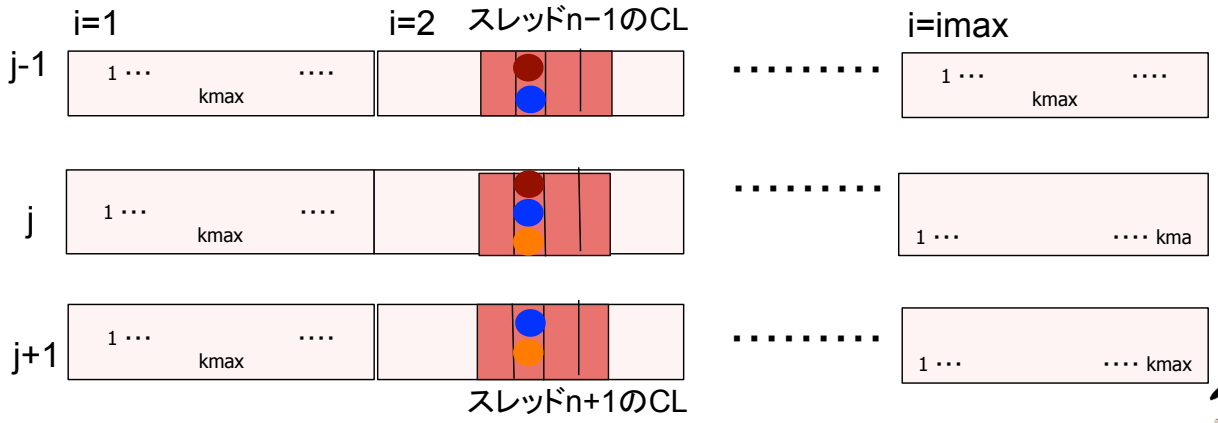
(cyclic分割スレッド並列の説明)

プログラム例

```
Do j=1,jmax
  do i=1,imax
    do k=1,kmax
      a(k,i,j)=・・・c0*v(k,i,j-1)+c1* v(k,i,j)+c2* v(k,i,j+1)・・・
    end do
  end do
end do
```

● :スレッドn−1で参照するデータ
● :スレッドnで参照するデータ
● :スレッドn+1で参照するデータ

キャッシュライン(CL)



空間微分Y方向の計算a) b)
(ZXYループ融合cyclicスレッド並列)

```
!$OMP DO SCHEDULE(static,1)
   do J = 1, NY
     do I = 1, NX
       do K = 3, NZ-1
         DZV (k,I,J) = (V(k,I,J)  -V(k-1,I,J))*R42 &
                     - (V(k+1,I,J)-V(k-2,I,J))*R43
         DXV (k,I,J) = (V(k,I,J)  -V(k,I-1,J))*R40&
                     - (V(k,I+1,J)-V(k,I-2,J))*R41
         DYV (k,I,J) = (V(k,I,J)  -V(k,I,J-1))*R40 &
                     - (V(k,I,J+1)-V(k,I,J-2))*R41
       end do
     end do
   end do
```

要求B/F	28/15 = 1.86
性能予測	0.36/1.86 = 0.19
実測値	0.177

要求B/F値を下げる
キャッシュに載せる

- K,I,J軸差分のループを融合することにより、V(K,I,J)のロードを共通化でき、プログラムの要求B/F比を下げる。

要求Byteの算出:

Store 3 +4 load と考えると、

(3+4)*4 = 28byte

要求flop:

add：9 mult：6 = 15

XFILL指示行によるチューニング

- **下記ループのdzvはストアのみの配列**
- **しかし通常はロードとストアが発生するためメモリアクセスは２と計算する**
- **京のコンパイラではXFILL指示行の機能をもつ**
- **この指示行を指定することにより余計なロードが発生しなくなる**
- **これにより要求B/F値は2.4から1.6に減少し推定性能が15%から22.5%に向上する**
- **実測値は21.2%**

```
!OCL XFILL
  do j = 1, NYP
    do i = 1, NXP
      do k = 3, NZ-1
        dzv (k,i,j) = ( v(k,i,j)   - v(k-1,i,j) ) * R40 &
                    - ( v(k+1,i,j) - v(k-2,i,j) ) * R41
      end do
    end do
  end do
```

キャッシュスラッシングの解消

- 連続アクセスのキャッシュミスの基準値はseism3Dは単精度であるため3.125%となる（1回/32回=1/（4B/128B））
- L1D$ミス率が基準値を超えL1D$ミスdm率が20%を超えるとL1キャッシュスラッシングの可能性が高い
- L1キャッシュスラッシングはループ内の配列ストリームアクセスが多い場合に起きる確率が高くなる
- このような場合はループ分割や配列融合が効果的

境界条件応力場更新	Org	Tune
L1Dミス率	3.54%	2.71%
L1Dミスdm率	49.93%	11.83%
L1Dミスhwpf率	25.99%	88.17%
L1Dミスswpf率	24.08%	0.00%
L2ミス率	2.11%	1.97%
L2スループット	42.32GB/s	35.79GB/s
メモリスループット	39.86GB/s	42.87GB/s
Peak Ratio	8.80%	10.05%

オリジナル:
- L1Dミス率:3.54%(>3.125%)
- L1Dミスdm 49.93%(>20%)
  → L1キャッシュ競合により，メモリスループットが十分でない．

ループ分割 ⇩ 配列融合

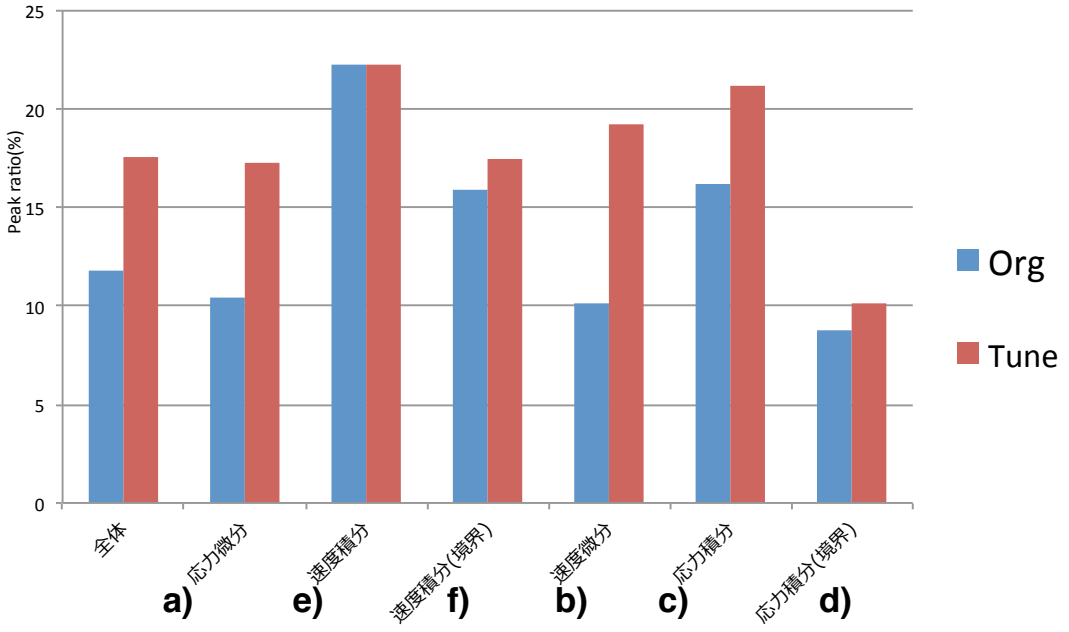
チューニング:
- L1Dミス率，L1Dミスdm率が基準値を下回る．
- ループ分割の結果，最内ストリームが減少し，hwpfによるミスが支配的．
- メモリスループットおよび性能が向上

速度時間積分の計算 e)

オリジナルコードの結果

要求B/F	72/52=1.38
性能予測	0.36/1.38 = 0.26
実測値	0.240

CPU単体性能チューニングの結果



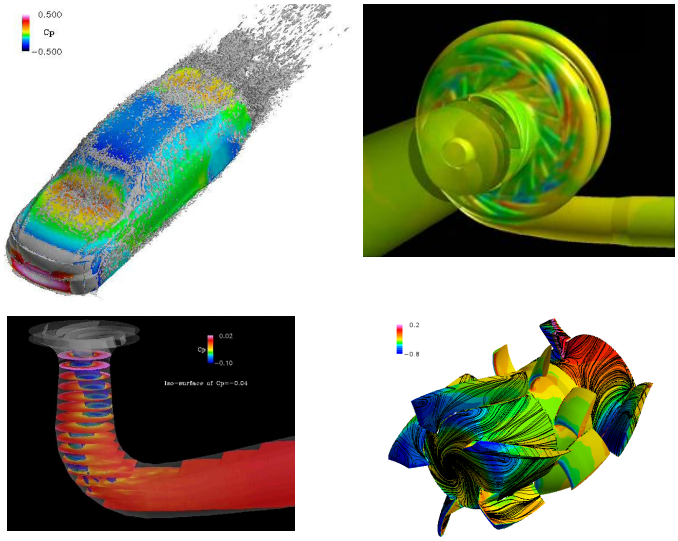
Seism3Dの総合性能

(*)通信を含む性能

The number of node	Elapse time(sec)	Ratio to peak performance
16	48.8	17.1%
256	48.8	17.6%
4096	48.9	17.7%
16384	48.8	17.8%
36864	48.6	17.9%
64512	48.5	17.9%
82944	48.5	17.9%

- 8万ノードまでの良好なウィークスケーラビリティを得られた
- フルノードでトータル性能1.9Pflopsを達成

FrontFlow/blueの性能最適化

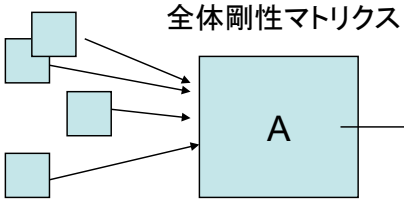


FrontFlow/blue (FFb)

- 有限要素法を用いた流体計算のプログラム
- 有限要素法には2つのタイプの計算方法がある
  - 全体剛性マトリクスを構築するタイプ
  - 全体構成マトリクスを構築せずに要素剛性マトリクスのみで計算を進めるタイプ（エレメント・バイ・エレメント法）
- FFBは新バージョンにおいて両方のソルバに対応

全体剛性マトリクスを使用する方法（陰解法の場合）

要素剛性マトリクス

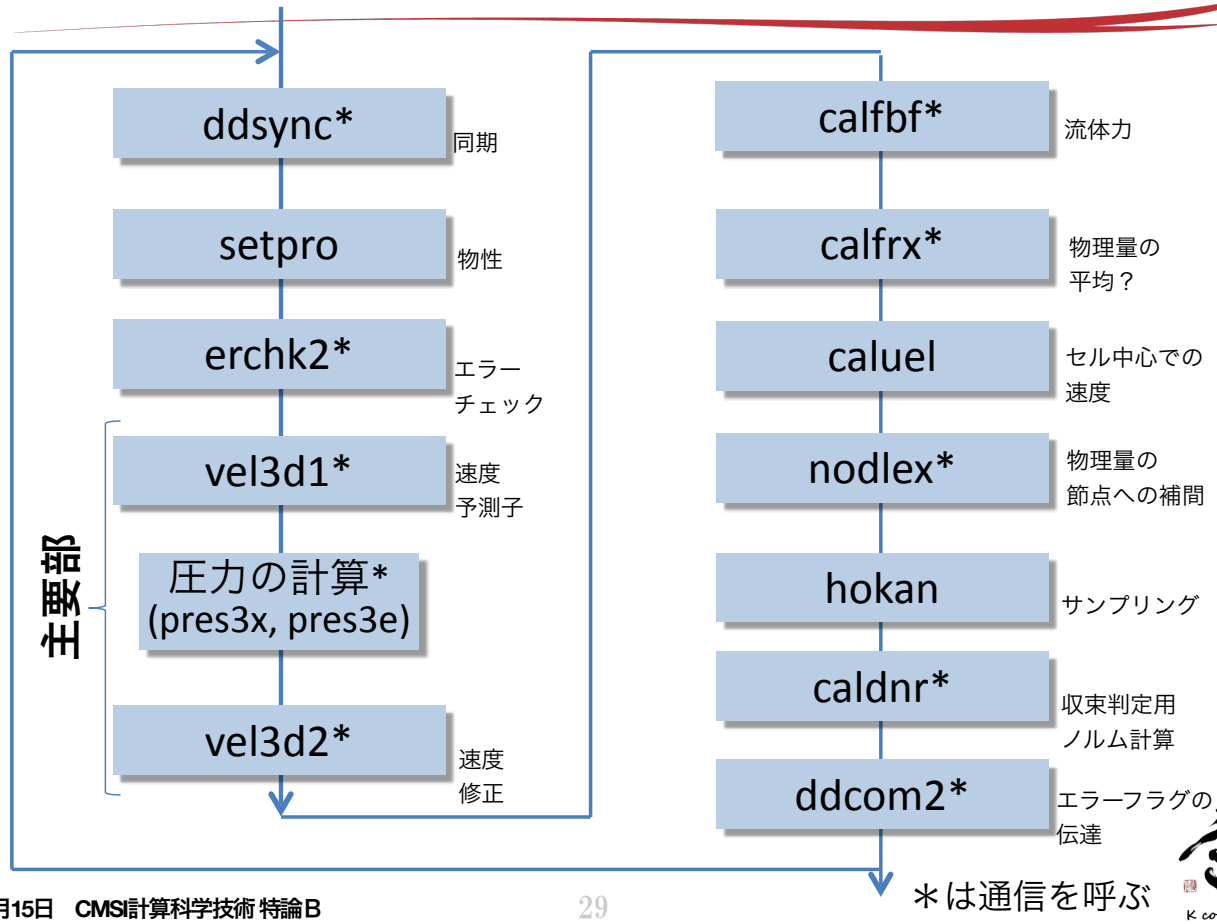


ソルバーを用いて解く

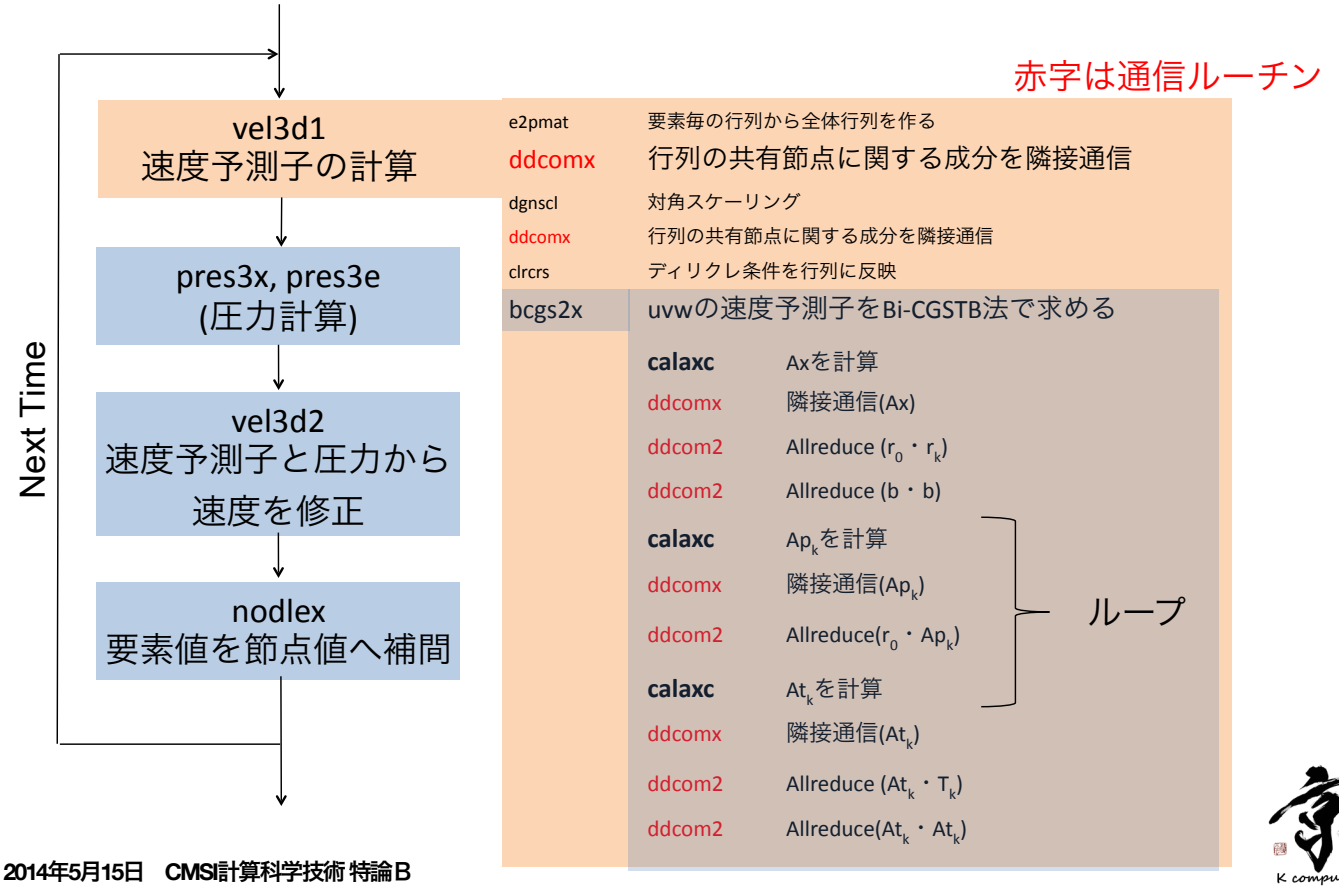
$Ax=b$

・Xは節点上の値となりAは周りの節点を関連づける粗行列となる
・ソルバーの演算は粗行列のマトリクス/ベクトル積が中心となる

FFbのタイムステップループ構造



FFbの主要部の構造

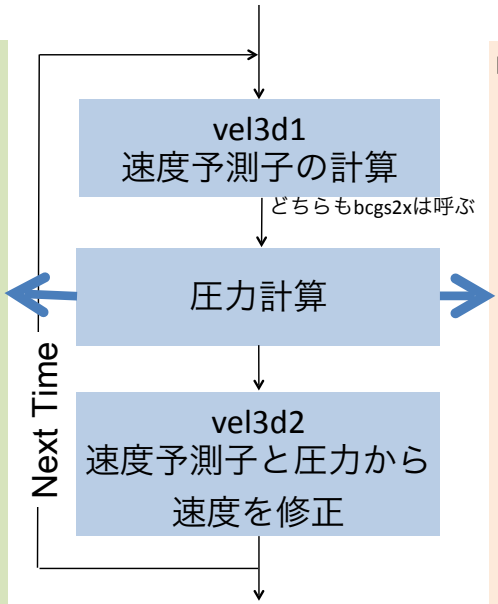


FFbの主要部の構造

節点圧力モード

pres3x
|
+--fild3x 要素内での速度の発散
|
+--ddcomx 右辺ベクトルbの隣接通信
|
+--bcfix2 境界条件をAとbに入れる
|
+--e2pmat CRS形式で全体行列を作る
|
+--bcgs2x Ax=bを解いて圧力を求める
| |
| +--**calaxc** -> ddcomx
| |
| +--ddcom2 内積×２
| |
| (ループ 要素毎)
| **calaxc** -> ddcomx
| ddcom2 内積
| **calaxc** -> ddcomx
| ddcom2 内積×４
|
+--aveprt 圧力(on節点)を隣接通信
|
+-- ddcomx

エレメントバイエレメント



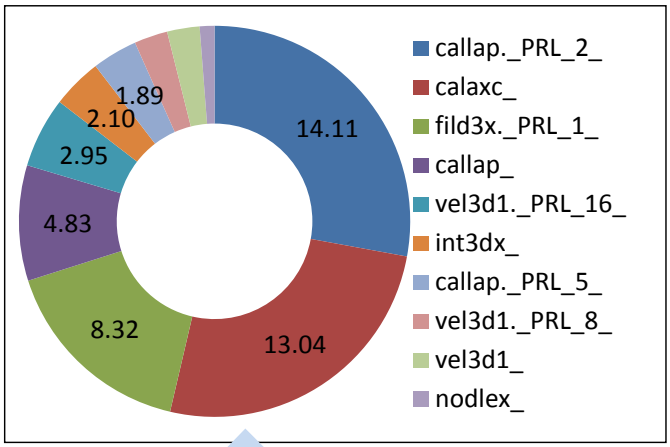
要素圧力モード

pres3e
|
+--fild3x 要素内での速度の発散
|
+--bcgsxe 要素毎で圧力を解く
| |
| +--**callap**
| | |
| | +--ddcomx
| | +--fild3x
| |
| +--ddcom2 内積×２
| |
| (ループ 要素毎
| |
| +--**callap**
| | |
| | +--ddcomx
| | +--fild3x
| |
| +--ddcom2 内積
| +--**callap**
| | |
| | +--ddcomx
| | +--fild3x
| |
| +--ddcom2 内積×４
|
+--nodlex 要素圧力を節点値に補間
|
+--ddcomx

全体剛性マトリクス作成

FFbの主要計算

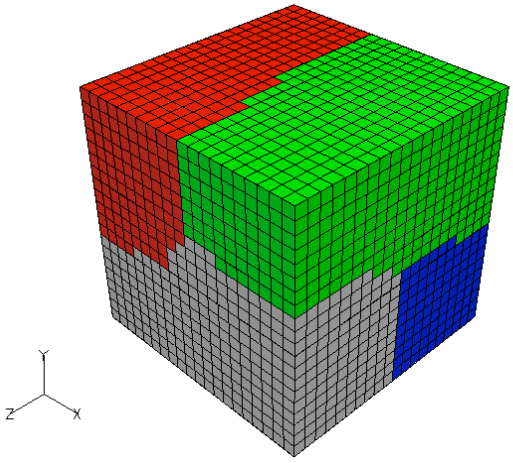
要素圧力モードでの処理時間内訳，上位10件

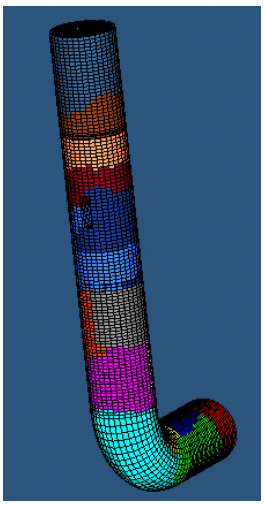


- 10万個の4面体要素＋α
- 14.11s(30%)が4面体要素の勾配計算
- 13.04s(27%)が疎行列ベクトル積
- 8.32s（17%)が4面体要素の発散計算

FFbの並列化

3次元の領域分割（非構造格子）

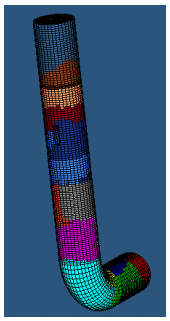




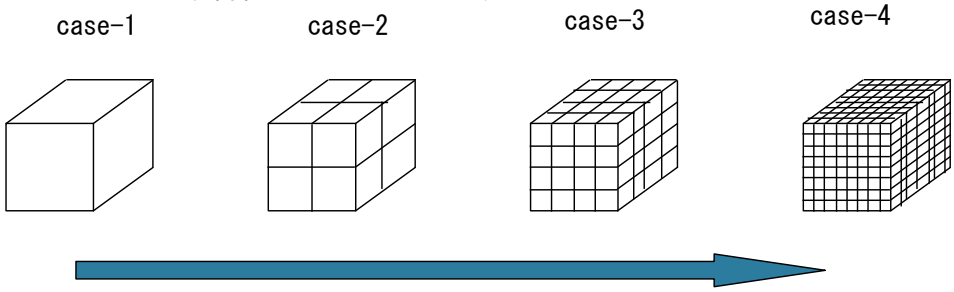
FFbの並列特性分析

初期のウィークスケール評価

評価モデルについて



メッシュ規模拡大イメージ（メッシュを細かくしただけで、評価モデルは同じもの）

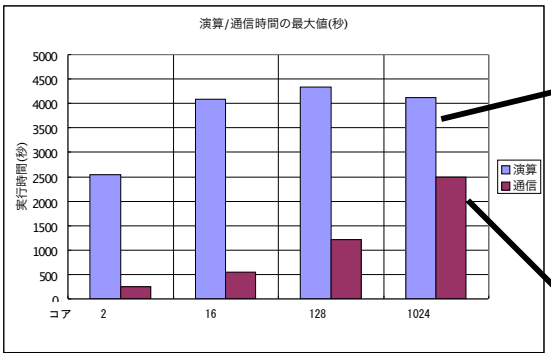


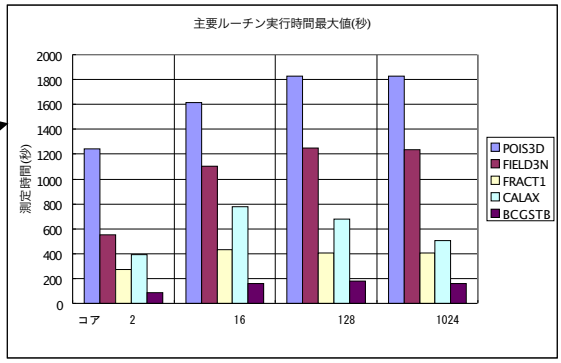
	基本モデル	基本モデル×8	基本モデル×64	基本モデル×512
領域分割数（コア数）	2	16	128	1024
要素数	53, 460	427, 680	3, 421, 440	27, 371, 520
節点数	60, 680	456, 342	3, 535, 646	27, 827, 454
要素数/コア	26, 730	26, 730	26, 730	26, 730
節点数/コア	30, 340	28, 521	27, 622	27, 175

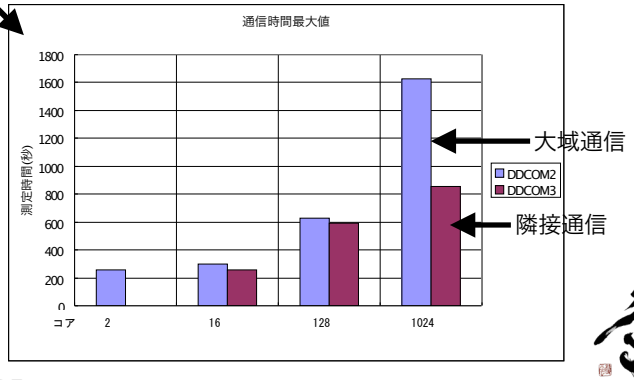
規模を拡大しても、1コア当たりの計算規模は同等としている

FFbの並列特性分析

初期のウィークスケール評価





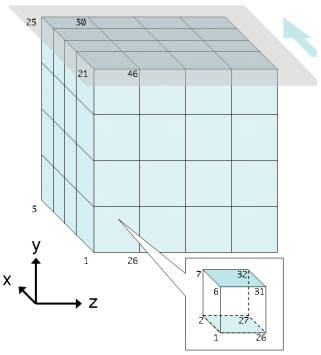


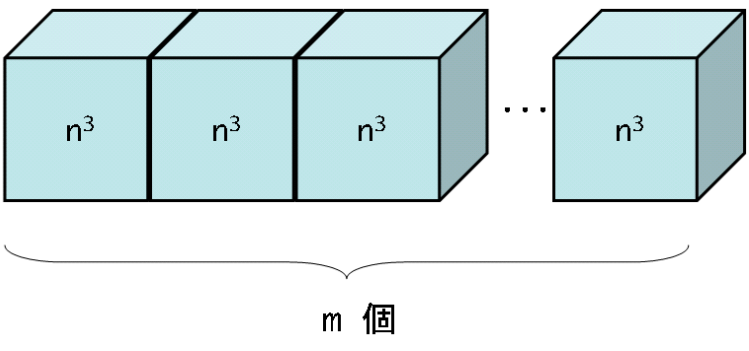
- 演算がウィークスケールしているのかよく分からない
- 隣接通信も大域通信も増大しているように見える
- 複雑な問題で測定しているので色々な要素が入ってきている
- シンプルにアプリの問題だけをあぶり出せていない

FFbの並列特性分析

問題設定を簡素化

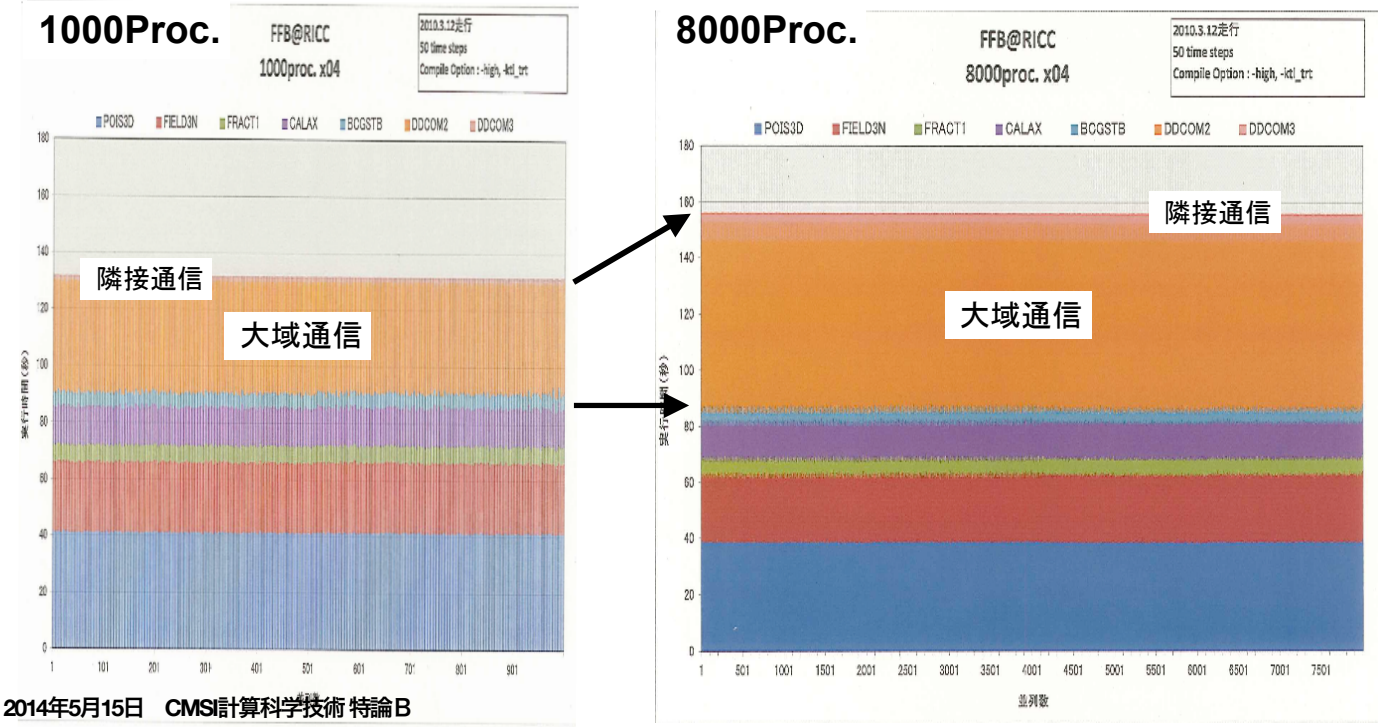
- 下左図のようなキャビティの問題がある
- その問題を下右図のように横に連結してそれぞれのノードが同じ問題を解く仮想的な問題を作成
- こうすることにより各ノードは全く同じ問題を計算する完全なウィークスケールの問題となる
- これでシンプルにアプリケーションの問題のみをあぶり出す事が可能となる





FFbの並列特性分析

- ここでは1000プロセスと8000プロセスの結果を示す
- 演算は完全にウィークスケールしておりロードインバランスもない
- 隣接通信の増大もないが大域通信（スカラ値のallreduce）のみが問題と分かった



FFbの単体性能最適化（性能見積りとチューニング）

FFbは要求B/F値が大きく，かつリストアクセスを使用するアプリケーション

節点圧力カーネル（calaxc相当）の性能見積り

オリジナルコード(疎行列とベクトルの積)

```
ICRS=0
    DO 110 IP=1,NP
      BUF=0.0E0
      DO 100 K=1,NPP(IP)
        ICRS=ICRS+1
        IP2=IPCRS(ICRS)
        BUF=BUF+A(ICRS)*S(IP2)
100   CONTINUE
      AS(IP)=AS(IP)+BUF
110 CONTUINE
```

リスト　ベクトル　行列

- CRS格納形式の行列ベクトル積
- ベクトルアクセスがリストアクセスとなる
- ベクトルの部分がL1キャッシュに載っていると仮定した場合
- ベクトルのメモリへのアクセスを全く無視してよい
- メモリからのロードは行列とリストのみ

要求Byteの算出:

単精度：2 load なので

2*4 = 8byte

要求flop:

add：1 mult：1 = 2

要求B/F	8/2 = 4
性能予測	0.36/4 = 0.09

(スレッド並列を仮定しピーク性能**128Gflops**に対して)

節点圧力カーネル（calaxc相当）の実測性能

(スレッド並列なし：1コア)

- オリジナルコードはスレッド並列されていなかった
- この状態の推定性能を前頁と同様な方法で見積もる
- メモリバンド幅を1コアで占有する場合のSTREAMベンチマークの結果は20GB/秒
- 1コアの理論ピーク性能は16GFLOPS
- 従って理論的なB/F値は20GB/16GFLOPで1.25

要求Byteの算出：2loadより 2* 4byte = 8

要求flop：1(add)+1(mult) = 2

要求B/F	8/2 = 4
性能予測	1.25/4= 0.313
実測値	0.059(六面体) 0.024(四面体)

(スレッド並列なしピーク性能**16Gflops**に対して)

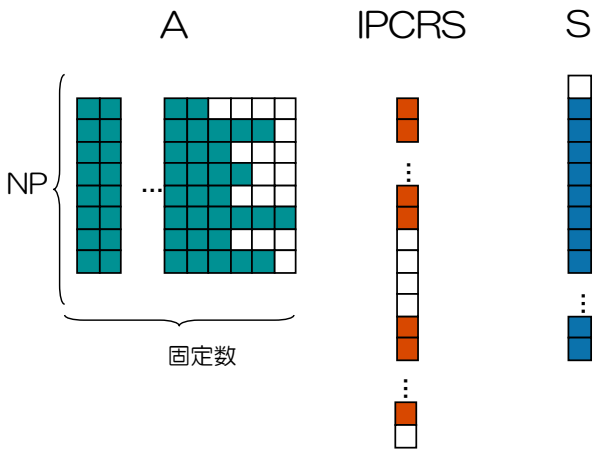
- ベクトルがリストアクセス
- 連続アクセスでないためプリフェッチが効きにくい
- メモリアクセスのレイテンシが見える
- 最悪1キャッシュラインのうち1要素しか使用できない事による大きなペナルティが発生
- 著しい性能低下が発生
- **L2**オンキャッシュでも同様のペナルティが発生

チューニング1: フルアンロール

狙い：
■　スケジューリングの改善(演算待ちの削減)

変更点:
■ 行列要素の配列に0を代入
■ベクトルインデックスの配列に0を代入
■余分な配列同士は0*0の演算を実施

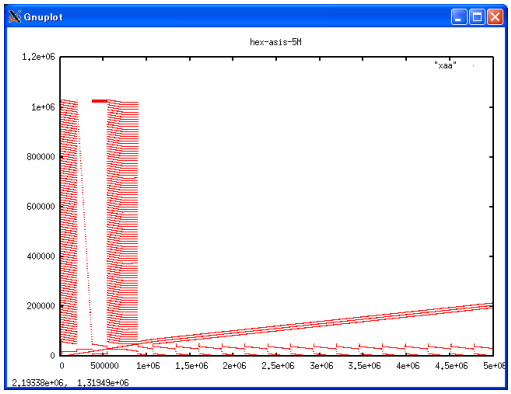
```
 ICRS=0
       DO 110 IP=1,NP
         BUF=0.0E0
!        DO 100 K=1,NPP(IP)    MAX_NZ=27
           BUF=BUF+A(ICRS+ 1)*S(IPCRS(ICRS+ 1))
     &            +A(ICRS+ 2)*S(IPCRS(ICRS+ 2))
     &            +A(ICRS+ 3)*S(IPCRS(ICRS+ 3))
     &            +A(ICRS+ 4)*S(IPCRS(ICRS+ 4))
      ･･･････････････････････
      ･･･････････(省略)･････
      ･･･････････････････････
     &            +A(ICRS+24)*S(IPCRS(ICRS+24))
     &            +A(ICRS+25)*S(IPCRS(ICRS+25))
     &            +A(ICRS+26)*S(IPCRS(ICRS+26))
     &            +A(ICRS+27)*S(IPCRS(ICRS+27))
           ICRS=ICRS+27
! 100     CONTINUE
          AS(IP)=AS(IP)+BUF
  110 CONTINUE
```



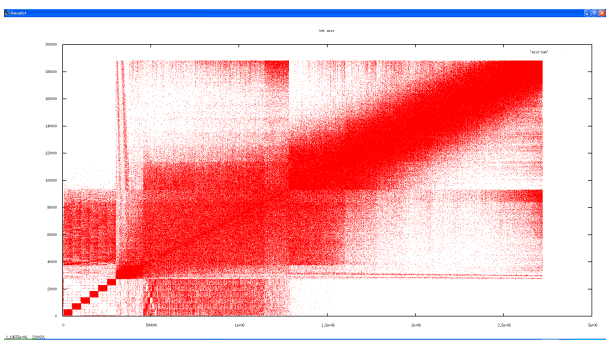
チューニング2: リオーダリング (1/4)

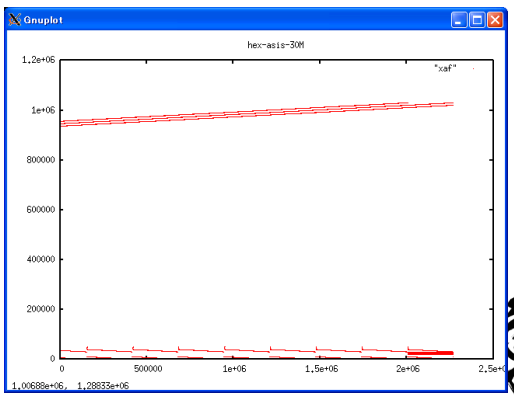
狙い：
■ベクトルデータ(S)のブロック化によるL1,L2キャッシュミスの削減
オリジナルデータの特徴：
■6面体(総回転数: 約2700万)
　■ 最初の1M回のSへの広範囲なランダムアクセス
　■ それ以降は二極化するが、局所的アクセス
■4面体(総回転数:約270万)
　■ 全アクセスとも、広範囲なランダムアクセス

■　6面体 オリジナルデータ



■　4面体　オリジナルデータ

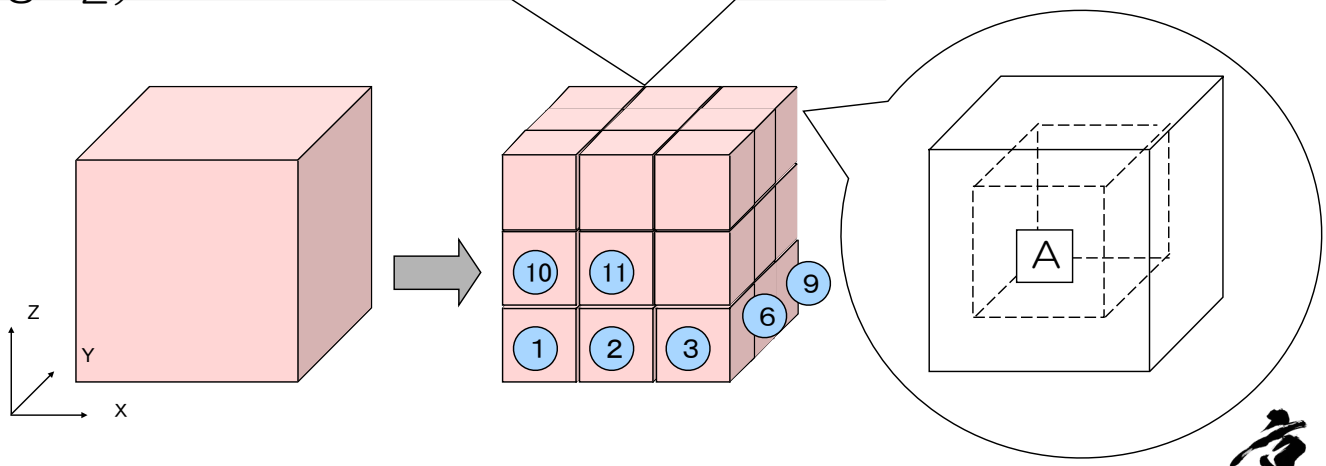




チューニング2: リオーダリング (2/4)

節点番号のリオーダリング:
■オリジナルデータを各軸分割しブロックを作成
■各ブロックを外と内に分割し物理座標に基づき内側・外側の順にナンバリング

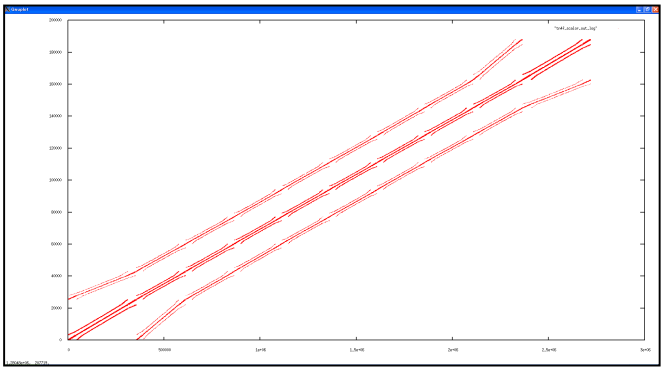
ひとつのブロックを内と外に分け、内側(A)のナンバリング後、外側(B)のナンバリングを実施(内:外の比８：２)



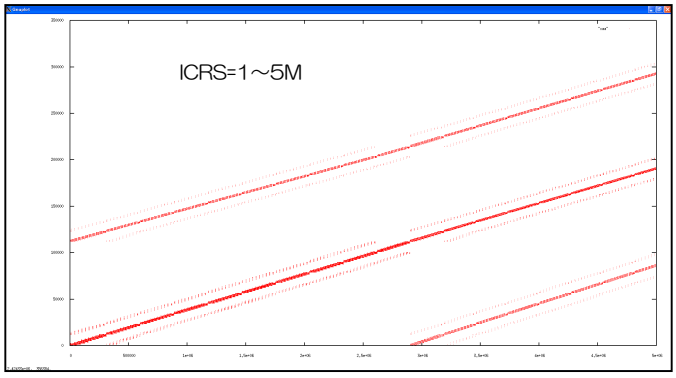
チューニング2: リオーダリング (3/4)

- 物理的に近い節点が配列の並びとしても近い位置に配置される事を期待
- 一要素を構成する節点の番号が近くなる
- 一箱の大きさを調整することによりベクトルのリストアクセスの多くに対しL1オンキャシュのデータを利用できる

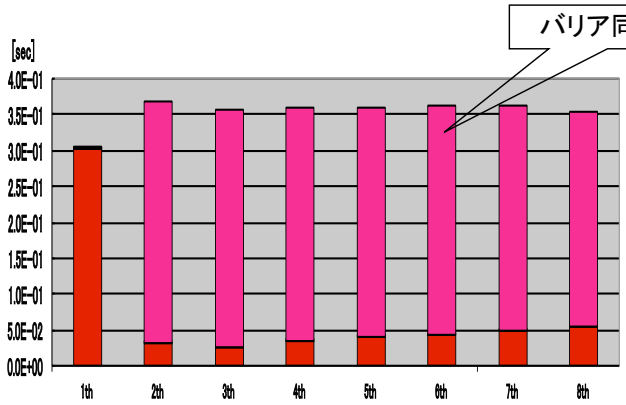
■　４面体 リオーダリング結果



■　６面体 リオーダリング結果

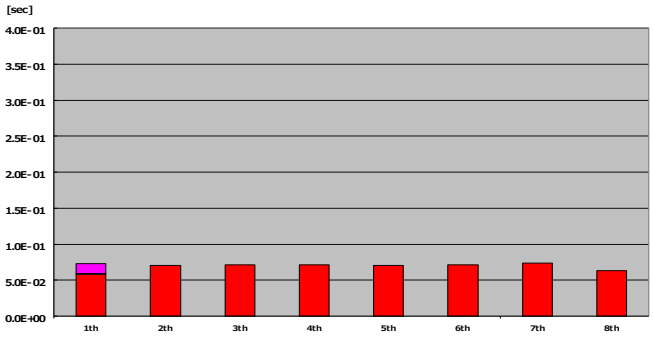


チューニング2: リオーダリング (4/4)



リオーダリング前
■マスタースレッドが担当する範囲にランダムアクセスが集中しメモリアクセスコストが増大
■スレーブスレッドが担当する範囲は局所アクセスが多く結果としてバリア同期待ちのコストが支配的

リオーダリング後
■全てのスレッドが均一的に局所アクセス化
■メモリアクセスバランスの改善と同期コストの大幅な削減



節点圧力カーネル (calaxc相当) チューニング結果

	6面体	4面体
フルアンロール (8core)	5.4%	3.0%
フルアンロール ＋ リオーダリング (8core)	8.1%	7.7%

L1 オンキャッシュである時の理論性能値である9%に近い性能値を実現

節点圧力カーネル（callap）のチューニング

カーネル概要 1/2

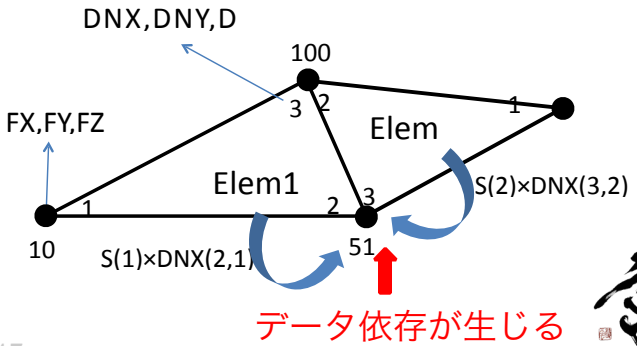
四面体要素について∇pの有限要素近似 $\nabla p = \frac{\partial p}{\partial x_i} \cong \sum_{j=1} \frac{\partial N_j}{\partial x_i} p_e$ を計算する

```
DO IE=1,NE
   IP1=NODE(1,IE)
   IP2=NODE(2,IE)
   IP3=NODE(3,IE)
   IP4=NODE(4,IE)
   SWRK=S(IE)
   FX(IP1)=FX(IP1)-SWRK*DNX(1,IE)
   FX(IP2)=FX(IP2)-SWRK*DNX(2,IE)
   FX(IP3)=FX(IP3)-SWRK*DNX(3,IE)
   FX(IP4)=FX(IP4)-SWRK*DNX(4,IE)

   FY(IP1)=FY(IP1)-SWRK*DNY(1,IE)
   FY(IP2)=FY(IP2)-SWRK*DNY(2,IE)
   FY(IP3)=FY(IP3)-SWRK*DNY(3,IE)
   FY(IP4)=FY(IP4)-SWRK*DNY(4,IE)

   FZ(IP1)=FZ(IP1)-SWRK*DNZ(1,IE)
   FZ(IP2)=FZ(IP2)-SWRK*DNZ(2,IE)
   FZ(IP3)=FZ(IP3)-SWRK*DNZ(3,IE)
   FZ(IP4)=FZ(IP4)-SWRK*DNZ(4,IE)
ENDDO
```

型	配列名とサイズ	内容
INTEGER*4	NODE(9,NE)	節点リスト
REAL*4	S(NE)	圧力
REAL*4	DNX(9,NE), DNY(9,NE), DNZ(9,NE)	形状関数の導関数
REAL*4	FX(NP), FY(NP), FZ(NP)	圧力勾配ベクトル

※NEは要素数，NPは節点数

単精度だが工学上問題はない

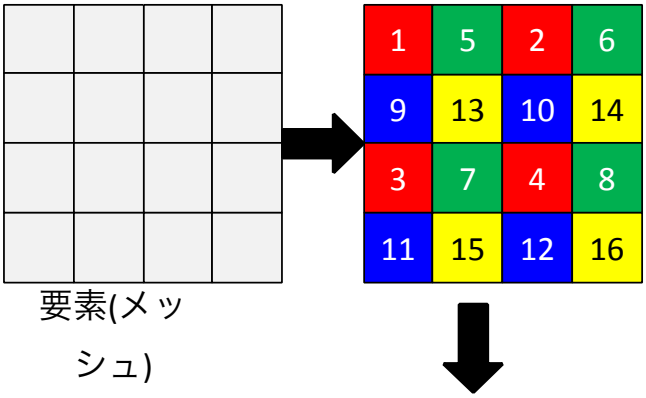


節点圧力カーネル（callap）のチューニング

カーネル概要 2/2

```
DO ICOLOR=1,NCOLOR(1)
   IES=LLOOP(ICOLOR  ,1)+1
   IEE=LLOOP(ICOLOR+1,1)
   DO IE=IES,IEE・・・・・・・ここで並列
      IP1=NODE(1,IE)
      IP2=NODE(2,IE)
      IP3=NODE(3,IE)
      IP4=NODE(4,IE)
      SWRK=S(IE)
      FX(IP1)=FX(IP1)-SWRK*DNX(1,IE)
      FX(IP2)=FX(IP2)-SWRK*DNX(2,IE)
      FX(IP3)=FX(IP3)-SWRK*DNX(3,IE)
      FX(IP4)=FX(IP4)-SWRK*DNX(4,IE)

      FY(IP1)=FY(IP1)-SWRK*DNY(1,IE)
      FY(IP2)=FY(IP2)-SWRK*DNY(2,IE)
      FY(IP3)=FY(IP3)-SWRK*DNY(3,IE)
      FY(IP4)=FY(IP4)-SWRK*DNY(4,IE)

      FZ(IP1)=FZ(IP1)-SWRK*DNZ(1,IE)
      FZ(IP2)=FZ(IP2)-SWRK*DNZ(2,IE)
      FZ(IP3)=FZ(IP3)-SWRK*DNZ(3,IE)
      FZ(IP4)=FZ(IP4)-SWRK*DNZ(4,IE)
   ENDDO
ENDDO
```

カラーリングによりデータ依存が生ずる要素を別のグループ(カラー)に分ける



節点圧力カーネル(callap)の性能見積り

- 要求バイト
  16要素×4B×(9/4)+1要素×4B=148B

- 浮動小数点演算数
  24FLOP

- 要求B/F値は148/24=6.17

- STREAMベンチマークによる「京」の実効メモリバンド幅は46.6GB/s

- 「京」の実効B/F値は46.6/128=0.36

- メモリバンド幅がネックとなり実効上の性能上限は128GFLOPSの0.36/6.17=5.83%

節点圧力カーネル(callap)の性能

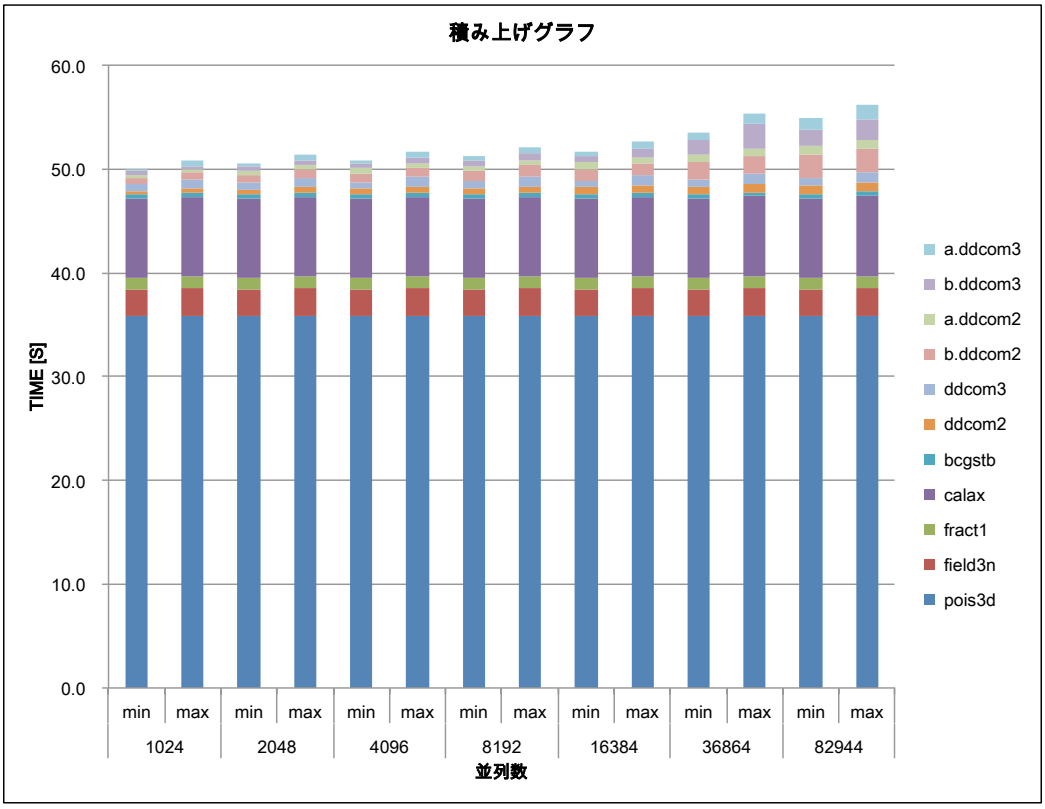
- カラーリングのみを実施しスレッド並列した時点で1.6%のピーク性能比
- calaxcと同様の節点のオーダリングを行う事でピーク性能比：3.78%を達成
- さらに配列融合を実施し4.41%まで向上

パターンNo	内容
1	DNX, DNY, DNZをDNXYZに融合
2	FX, FY, FZをFXYZに融合
3	パターン2＋演算淳子変更
4	パターン3＋パターン1

パターンNo	ピーク性能比%	L1D キャッシュミス率%	メモリスループット GB/s
1	3.95	3.98	35.70
2	4.05	3.69	35.91
4	4.05	3.69	35.88
5	4.41	3.37	38.80

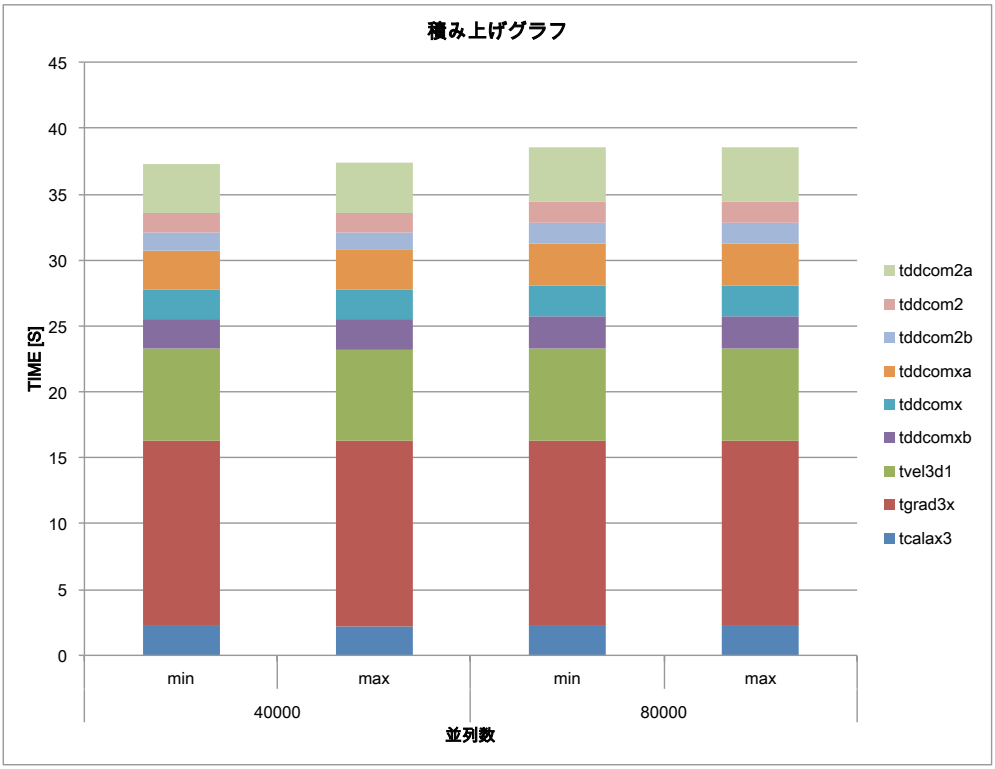
- 加えてブロック間の要素数のバランスを改善することによりピーク性能比：4.58%，メモリスループット：41.22GB/secまで到達

FFbのウィークスケール性能 (Ver5)



FFbの総合性能 (Ver7)

現状の総合性能は80000ノードでピーク性能比：3.16%



NPB MG（講義第2回で紹介）のチューニング事例

プロファイラーによるコスト分析

Procedures profile

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

Cost	%	Barrier	%	MPI	%	Start	End	
1367	100.0000	1	0.0732	8	0.5852	--	--	Application
620	45.3548	0	0.0000	0	0.0000	736	758	resid._OMP_1_
277	20.2633	0	0.0000	0	0.0000	664	686	psinv._OMP_1_
140	10.2414	0	0.0000	0	0.0000	826	858	rprj3._OMP_1_
131	9.5830	0	0.0000	0	0.0000	913	948	interp._OMP_1_
80	5.8522	0	0.0000	0	0.0000	1241	1243	ready._PRL_1_
20	1.4631	0	0.0000	2	10.0000	1398	1493	take3_
19	1.3899	0	0.0000	6	31.5789	1263	1393	give3_
17	1.2436	0	0.0000	0	0.0000	2290	2296	zran3._PRL_1_
15	1.0973	0	0.0000	0	0.0000	--	--	__jwe_etbf
14	1.0241	0	0.0000	0	0.0000	2115	2310	zran3_

ある程度チューニングされている
rprj3(10.2%)を分析しさらに高速化を目指す

ループのB/Fから推定性能を算出する

		回転数	Mst 配列	Mld 配列	L2st 配列	L2ld 配列	L1st 配列	L1ld 配列	加減算	乗算	要求B/F	推定性能
	rprj3	(最大値)	1	7	0	12	2	7	20	4	2.666667	0.135
1	do j3=2,m3j-1	256										
	i3 = 2*j3-d3											
2	do j2=2,m2j-1	256										
3	i2 = 2*j2-d2											
4	do j1=2,m1j	514										
5	i1 = 2*j1-d1											
6	x1(i1-1) = r(i1-1,i2-1,i3) + r(i1-1,i2+1,i3)											
	> + r(i1-1,i2, i3-1) + r(i1-1,i2, i3+1)			3		1	1	1	3			
7	y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)											
	> + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)					4	1	1	3			
8	enddo											
9	do j1=2,m1j-1	512										
10	i1 = 2*j1-d1											
11	y2 = r(i1, i2-1,i3-1) + r(i1, i2-1,i3+1)											
	> + r(i1, i2+1,i3-1) + r(i1, i2+1,i3+1)			2		2			3			
12	x2 = r(i1, i2-1,i3) + r(i1, i2+1,i3)											
	> + r(i1, i2, i3-1) + r(i1, i2, i3+1)			1		3			3			
13	s(j1,j2,j3) =											
	> 0.5D0 * r(i1,i2,i3)											
	> + 0.25D0 * (r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)											
	> + 0.125D0 * (x1(i1-1) + x1(i1+1) + y2)											
	> + 0.0625D0 * (y1(i1-1) + y1(i1+1))		1	1		2		5	8	4		
14	enddo											
15	end											
16	enddo											

性能の目安　B/F = 64/24= 2.67
メモリ律速と仮定すると、期待される性能値は　0.36/2.57 = 13.5%

ループrprj3の分析(１)

- ✓ ピーク性能比　4.94%
- ✓ メモリスループット

最大値: 46G/s

メモリビジー率：
メモリスループットの最大値(46G/s)に対する割合

Memory・Cache

メモリスループット (GB/sec)	L2エンジン スループット (GB/sec)	L2 スループット (GB/sec)	メモリビジー率	L2エンジンビジー率	L1エンジンビジー率
3.37	15.76	11.17			44%
3.34	15.71	11.18			44%
3.34	15.72	11.16			44%
3.35	15.69	11.14			44%
3.35	15.66	11.09	58%	70%	44%
3.33	15.72	11.18			44%
3.37	15.80	11.22			44%
3.35	15.73	11.17			44%
26.78	125.70	89.23			44%

B/Fが高いループであるが、メモリスループットが低い。。

ループrprj3の分析(２)

✓ キャッシュ状況

L1D、L2ミス率
倍精度であれば6.25%が基準値
それを上回り、L1Dミスdm率が20%を超えている場合は
L1キャッシュスラッシングが発生していると考える。

Cache

	L1I ミス率(/有効総命令数)	L1D ミス率(/ロード・ストア数)	ロード・ストア数	L1D ミス数	L1D ミス dm 率(/L1D ミス数)	L1D ミス hwpf 率(/L1D ミス数)	L1D ミス swpf 率(/L1D ミス数)	L2 ミス率(/ロード・ストア数)	L2 ミス数	L2 ミス dm 率(/L2 ミス数)	L2 ミス pf 率(/L2 ミス数)	μDTLB ミス率(/ロード・ストア数)	mDTLB ミス率(/ロード・ストア数)
Thread 0	0.03%	12.52%	2.99E+08	3.75E+07	41.47%	58.53%	0.00%	3.43%	1.03E+07	9.62%	90.38%	0.00599%	0.00017%
Thread 1	0.03%	12.58%	2.99E+08	3.76E+07	41.52%	58.48%	0.00%	3.40%	1.02E+07	8.57%	91.43%	0.00449%	0.00010%
Thread 2	0.03%	12.57%	2.99E+08	3.76E+07	41.81%	58.19%	0.00%	3.42%	1.02E+07	8.04%	91.96%	0.00431%	0.00005%
Thread 3	0.03%	12.53%	2.99E+08	3.75E+07	41.36%	58.64%	0.00%	3.41%	1.02E+07	9.05%	90.95%	0.00417%	0.00004%
Thread 4	0.03%	12.48%	2.99E+08	3.73E+07	41.78%	58.22%	0.00%	3.43%	1.03E+07	9.20%	90.80%	0.00414%	0.00003%
Thread 5	0.03%	12.59%	2.99E+08	3.77E+07	41.66%	58.33%	0.00%	3.41%	1.02E+07	8.31%	91.69%	0.00430%	0.00006%
Thread 6	0.03%	12.63%	2.99E+08	3.78E+07	41.84%	58.16%	0.00%	3.44%	1.03E+07	7.97%	92.03%	0.00415%	0.00004%
Thread 7	0.03%	12.57%	2.99E+08	3.76E+07	41.56%	58.44%	0.00%	3.43%	1.03E+07	8.96%	91.04%	0.00494%	0.00004%
Process	0.03%	12.56%	2.39E+09	3.01E+08	41.62%	58.37%	0.00%	3.42%	8.19E+07	8.71%	91.29%	0.00456%	0.00007%

L1キャッシュスラッシングが発生していると判定。

ループrprj3の分析結果

- L1キャッシュスラッシングが発生している。
- 演算系はSIMD化率が低い

⇒　本ループは、演算の比率が低い（B/Fが高い）ため、
まずはスラッシングを回避する処置が必要と判断する。

スラッシングの回避策:

- 使用する配列数を減らす
  - ✓ ループ分割、配列融合、スカラ化
- 使用する配列のアドレスをずらす
  - ✓ パディング
  - ✓ Common化

ループrprj3のチューニング(１)

テンポラリ配列x,yの利用をやめることにより、配列数を少なくするのと同時に
ループ融合を実施し、無駄なロードを発生させないチューニングを採用する。

```
 do  j3=2,m3j-1
       i3 = 2*j3-d3
C       i3 = 2*j3-1
       do  j2=2,m2j-1
         i2 = 2*j2-d2
C         i2 = 2*j2-1

         do j1=2,m1j
           i1 = 2*j1-d1
C           i1 = 2*j1-1
           x1(i1-1) = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >              + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
           y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >              + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
         enddo

         do  j1=2,m1j-1
           i1 = 2*j1-d1
C           i1 = 2*j1-1
           y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >        + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
           x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >        + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
           s(j1,j2,j3) =
     >           0.5D0 * r(i1,i2,i3)
     >         + 0.25D0 * ( r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >         + 0.125D0 * ( x1(i1-1) + x1(i1+1) + y2)
     >         + 0.0625D0 * ( y1(i1-1) + y1(i1+1) )
         enddo

       enddo
     enddo
```

```
!$OMP PARALLEL DO PRIVATE(xy1,y2,x2,i3,i2,i1)
       do  j3=2,m3j-1
         i3 = 2*j3-d3
         do  j2=2,m2j-1
           i2 = 2*j2-d2
           do  j1=2,m1j-1
             i1 = 2*j1-d1
C             i1 = 2*j1-1
             xy11m = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >               + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
             xy11p = r(i1+1,i2-1,i3  ) + r(i1+1,i2+1,i3  )
     >               + r(i1+1,i2,  i3-1) + r(i1+1,i2,  i3+1)
             xy12m = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >               + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
             xy12p = r(i1+1,i2-1,i3-1) + r(i1+1,i2-1,i3+1)
     >               + r(i1+1,i2+1,i3-1) + r(i1+1,i2+1,i3+1)
             y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >          + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
             x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >          + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
             s(j1,j2,j3) =
     >             0.5D0 * r(i1,i2,i3)
     >           + 0.25D0 * ( r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >           + 0.125D0 * ( xy11m + xy11p + y2)
     >           + 0.0625D0 * ( xy12m + xy12p )
           enddo

         enddo
       enddo
```


ループrprj3のチューニング(２)

配列rをpaddingすることにより、アドレスを明にずらす。
（rは1次元配列を3次元配列の引数で受け取るため、様々なルーチンに手が入る）

```
 do  j3=2,m3j-1
       i3 = 2*j3-d3
C       i3 = 2*j3-1
       do  j2=2,m2j-1
         i2 = 2*j2-d2
C         i2 = 2*j2-1

         do j1=2,m1j
           i1 = 2*j1-d1
C           i1 = 2*j1-1
           x1(i1-1) = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >              + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
           y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >              + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
         enddo

         do  j1=2,m1j-1
           i1 = 2*j1-d1
C           i1 = 2*j1-1
           y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >        + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
           x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >        + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
           s(j1,j2,j3) =
     >             0.5D0 * r(i1,i2,i3)
     >           + 0.25D0 * ( r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >           + 0.125D0 * ( x1(i1-1) + x1(i1+1) + y2)
     >           + 0.0625D0 * ( y1(i1-1) + y1(i1+1) )
         enddo

       enddo
     enddo
```

```
 double precision r(m1k+F_PAD,m2k,m3k), s(m1j+F_PAD,m2j,m3j)
                                :
!$OMP PARALLEL DO PRIVATE(xy11m,xy11p,xy12m,xy12p,y2,x2,i3,i2,i1)
       do  j3=2,m3j-1
         i3 = 2*j3-d3
         do  j2=2,m2j-1
           i2 = 2*j2-d2
           do  j1=2,m1j-1
             i1 = 2*j1-d1
C             i1 = 2*j1-1
             xy11m = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >               + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
             xy11p = r(i1+1,i2-1,i3  ) + r(i1+1,i2+1,i3  )
     >               + r(i1+1,i2,  i3-1) + r(i1+1,i2,  i3+1)
             xy12m = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >               + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
             xy12p = r(i1+1,i2-1,i3-1) + r(i1+1,i2-1,i3+1)
     >               + r(i1+1,i2+1,i3-1) + r(i1+1,i2+1,i3+1)
             y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >          + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
             x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >          + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
             s(j1,j2,j3) =
     >             0.5D0 * r(i1,i2,i3)
     >           + 0.25D0 * ( r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >           + 0.125D0 * ( xy11m + xy11p + y2)
     >           + 0.0625D0 * ( xy12m + xy12p )
           enddo

         enddo
       enddo
```

ループrprj3のチューニング(３)

	Before	After
L1Dミス率	12.56%	6.45%
L1Dミスdm率	41.62%	15.44%
メモリスループット	26.78G/s	41.33G/s
Peak性能	4.94%	10.96%

倍精度の連続ミス率基準値(6.25%)に近くなり、
性能、メモリスループットが改善
ピーク性能についてはまだ改善の余地はあると思われる。

まとめ

- 理研で進めた性能最適化
- Seism3Dの性能最適化
- FrontFlow/blueの性能最適化
- NPB MGのチューニング事例