

## 資料4-1: FFBのチューニング

2012年8月

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



RIKEN Advanced Institute for Computational Science

# Front Flow/blueの チューニング

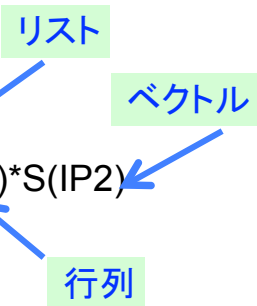
# Front Flow/blue(FFB)の概要

- 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
- 本講演の疎行列とベクトルの積は前者のソルバで使  
用される計算カーネル

# オリジナルコードの性能予測(理想的なケース)

## オリジナルコード

```
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 CONTINUE
```



- 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

$$2 * 4 = 8 \text{ byte}$$

## 要求flop:

$$\text{add} : 1 \quad \text{mult} : 1 = 2$$

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)

# オリジナルコードの性能予測と実測

## (スレッド並列なし：1コア)

- 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  $2 * 4\text{byte} = 8$

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)

- 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ccessのレイテンシが見える
- 1ラインのうち1要素しか使用し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

(スレッド並列なしピーク性能**16Gflops**に対して)

# チューニング1: フルアンロー

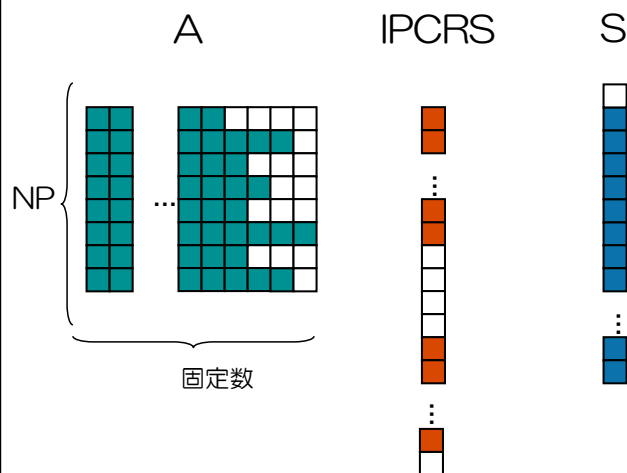
狙い:

- 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)

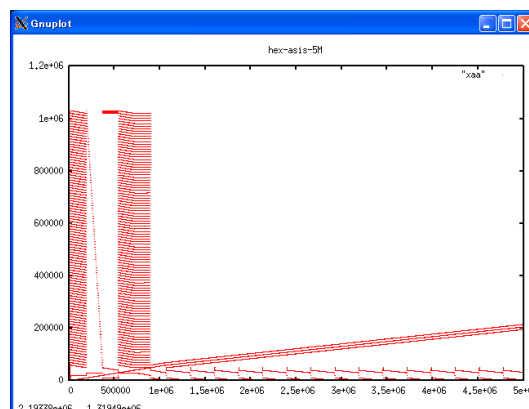
狙い:

- 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

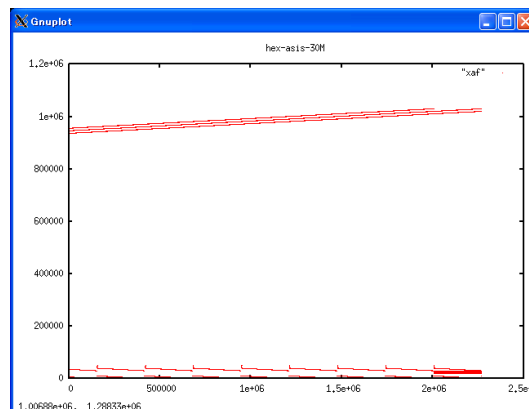
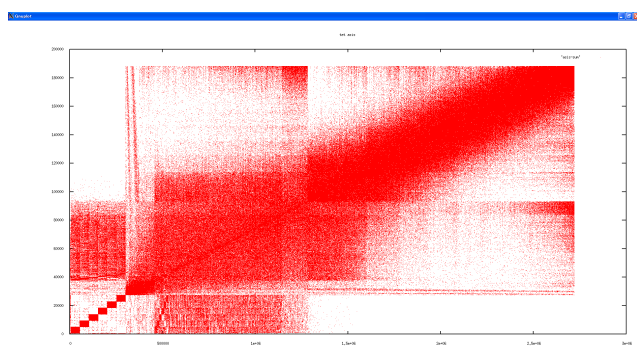
オリジナルデータの特徴:

- 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

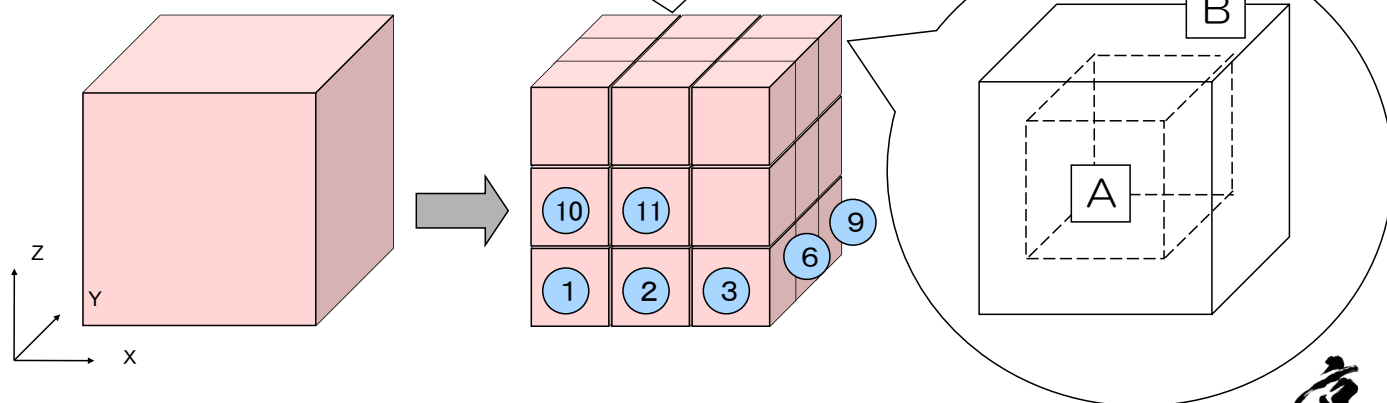


# チューニング2: リオーダーリング (2/4)

節点番号のリオーダーリング:

- 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外と内に分け、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)

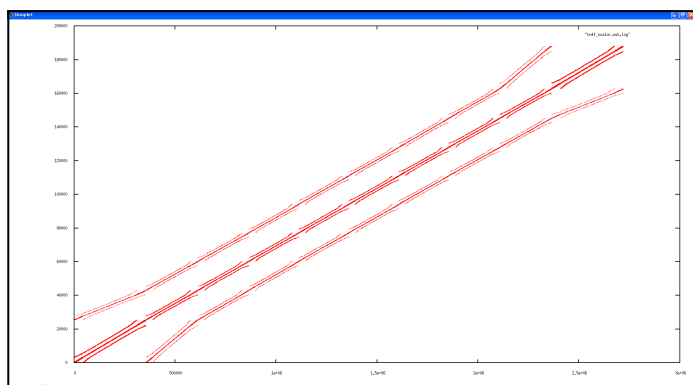




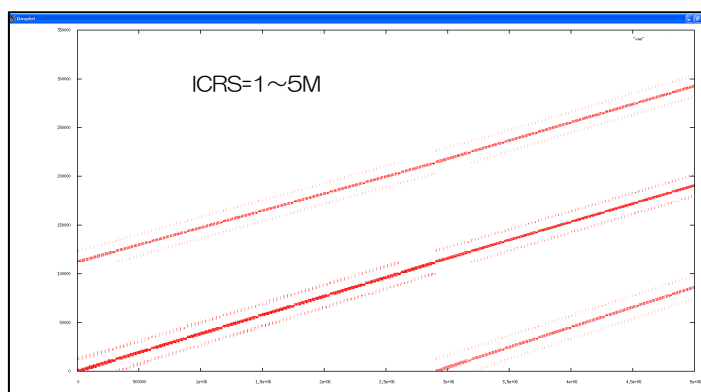
# チューニング2: リオーダーリング (3/4)

- 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オンキャッシュのデータを利用できる

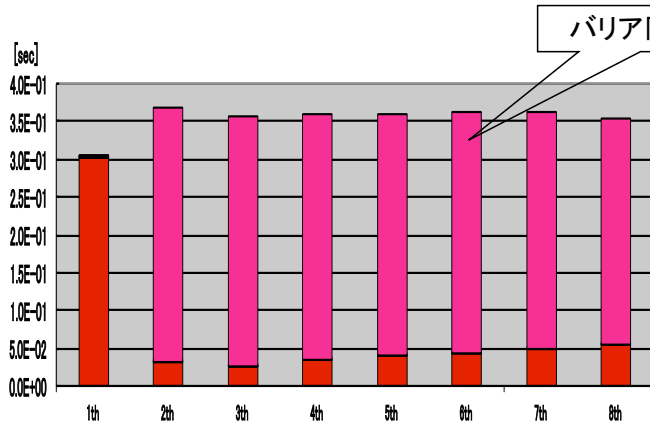
## ■ 4面体 リオーダーリング結果



## ■ 6面体 リオーダーリング結果



# チューニング2: リオーダーリング (4/4)

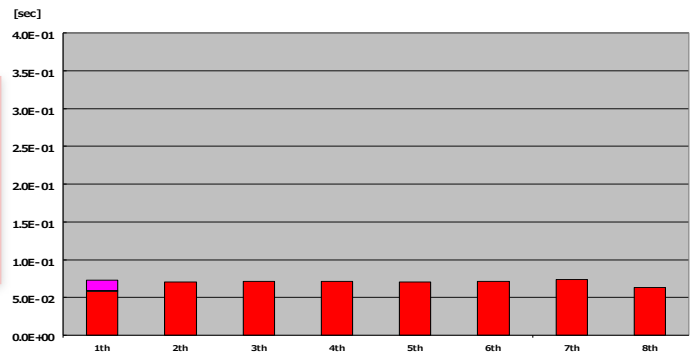


## リオーダーリング前

- 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

## リオーダーリング後

- 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



## FFBカーネルの結果まとめ

	6 面体	4 面体
オリジナル(1core)	5.9%	2.4%
フルアンロール (1core)	10.8%	4.2%
フルアンロール (8core)	5.4%	3.0%
フルアンロール + リオーダーリング (1core)	10.2%	10.2%
フルアンロール + リオーダーリング (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