東京大学本郷キャンパス 工学部8号館 84講義室 (地下1階)

新機能紹介：
節点データ並べ替え処理の
FrontISTRプログラムへの移植

2017年1月30日
第33回FrontISTR研究会
＜FrontISTR開発の最新動向＞

謝辞

本研究開発は，文部科学省ポスト「京」
重点課題⑧「近未来型ものづくりを先導する
革新的設計・製造プロセスの開発」の
一環として実施したものです。

新機能紹介：
節点データ並べ替え処理のFrontISTRプログラムへの移植

発表内容

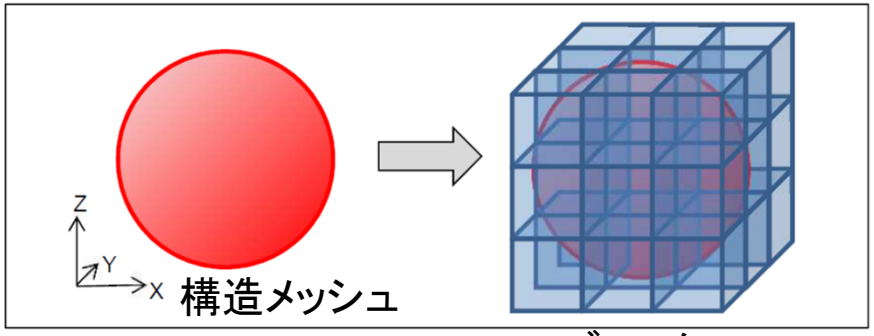
1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

新機能紹介：
節点データ並べ替え処理のFrontISTRプログラムへの移植

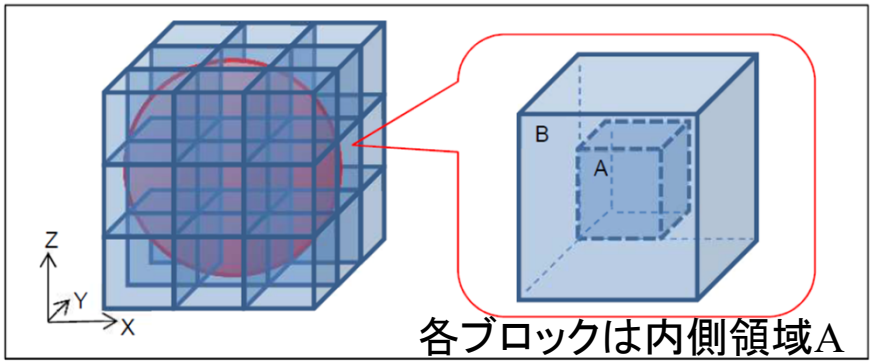
発表内容

1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

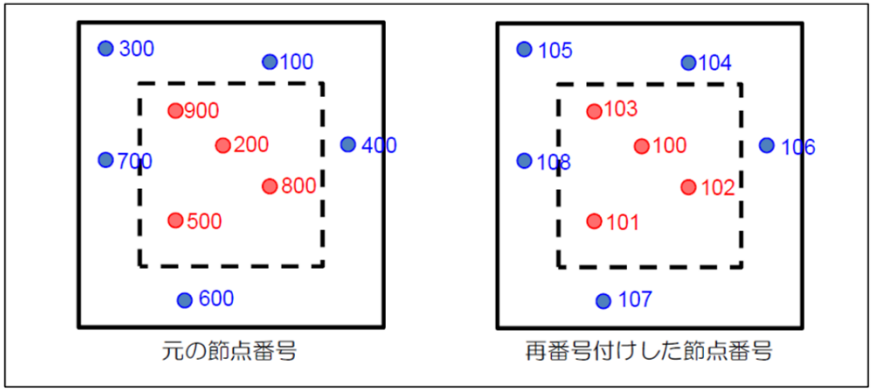
節点データ並べ替え処理 (リオーダリング処理)



(1) 構造メッシュ (単一領域メッシュ) の
バウンディングボックスを
均等なブロックで分割



(2) ブロック順に，各ブロックに対して
節点番号のリナンバリング



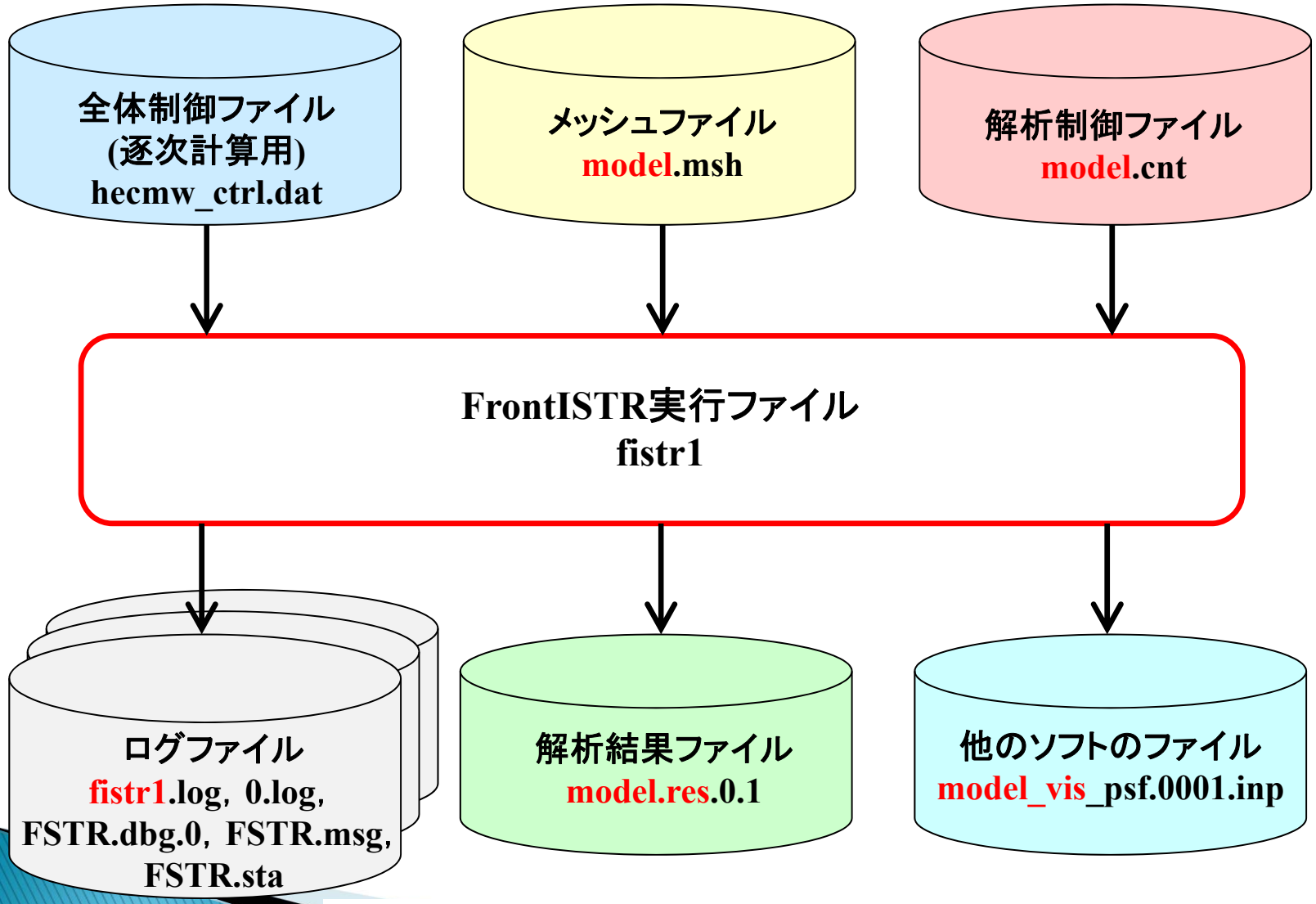
内側領域A，外側領域Bの順に節点番号を詰める

<u>リナンバリングのルール</u>

(a) 節点座標の値と無関係に
リナンバリング

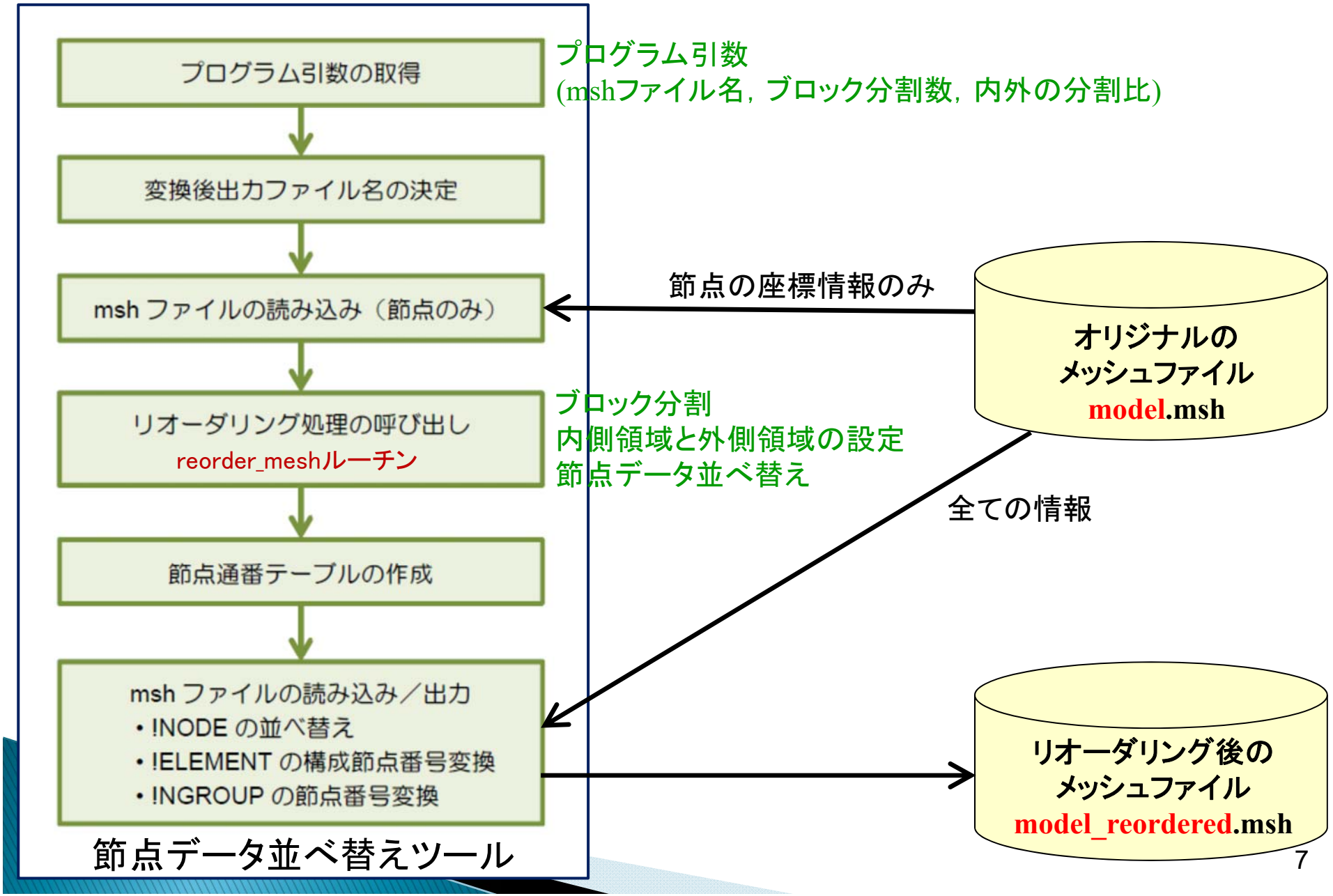
(b) 各ブロックに対して，
まず内側領域Aの番号を詰め，
次に外側領域Bの番号を詰める
ようにリナンバリング

FrontISTRの逐次計算の流れ

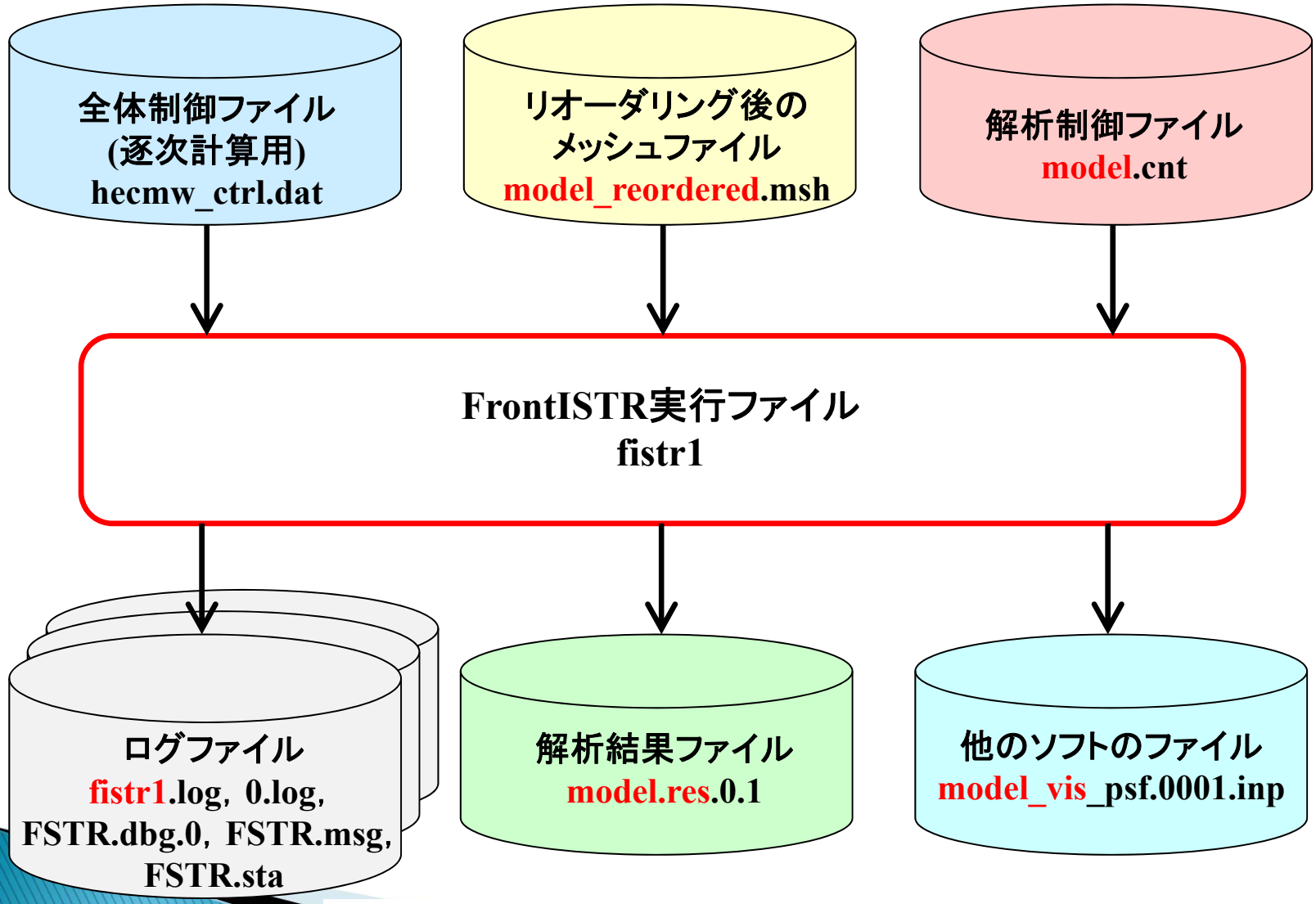


※ 赤字の名前は自由に変更可能

節点データ並べ替えツール



FrontISTRの逐次計算の流れ (節点データ並べ替えツール実行後)



※ 赤字の名前は自由に変更可能

新機能紹介：
節点データ並べ替え処理のFrontISTRプログラムへの移植

発表内容

1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

計測に使用する計算機

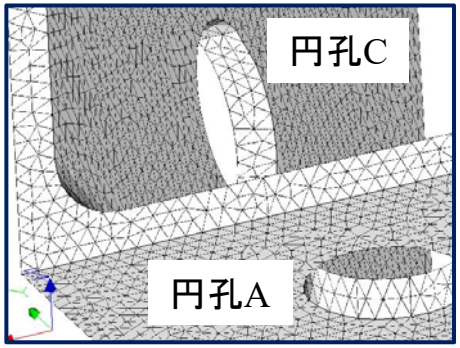
富士通FX10

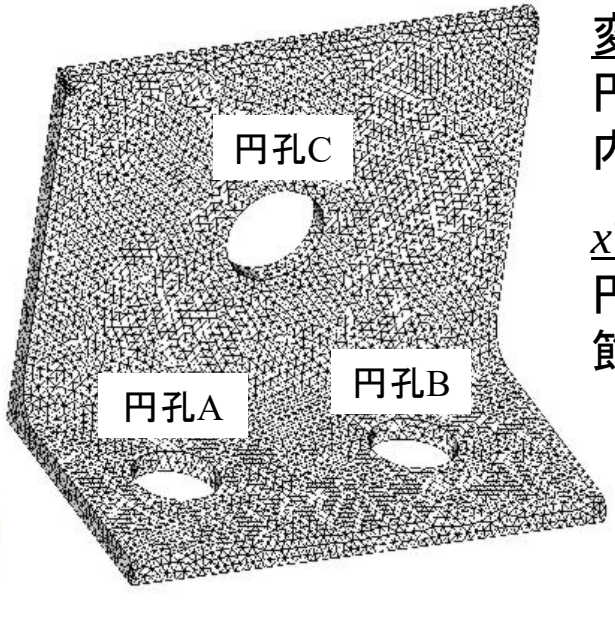
項目	仕様
プロセッサ	SPARC64™ IXfx （1.848GHz,16 コア）
キャッシュ容量	L1：32KB/コア、L2：12MB/CPU
理論演算性能	236.5Gflops/CPU
主記憶容量	64GB/CPU
主記憶-CPU 間帯域幅	85.0GB/s/CPU
主記憶-CPU 間帯域幅（STREAM Triad）	67.8GB/s/CPU
主記憶-CPU 間理論 B/F 比（倍精度）	0.36
主記憶-CPU 間実効 B/F 比（倍精度）	0.29

計測に使用する解析モデル

解析メッシュのデータ

要素数	49,871（TETRA2 次）
節点数	84,056
下三角部分の非零節点数	1,015,956
上三角部分の非零節点数	1,015,956
全体非零節点数	2,115,968





<u>変位零</u>
円孔Aと円孔Bの
内周面上の節点

<u>*x*方向に節点集中荷重</u>
円孔Cの内周面上の
節点

解析メッシュおよび荷重・境界条件

計測に使用するFrontISTRの線形ソルバーと対象ルーチン

使用する線形ソルバー: ブロックSSOR前処理付きCG法

基本プロファイラの結果

	OpenMP並列数			プログラムの行番号
	1スレッド	8スレッド	16スレッド	
前処理 (precond)	45.7%	51.1%	41.3%	322-440
SpMV (matvec)	45.3%	29.2%	28.3%	271-314
その他	9.0%	19.7%	30.4%	ー

- 前処理 (percond) とSpMV (matvec) のルーチンが
  プログラムの中でかなりのコストを占める
  (1スレッド: 約90%, 8スレッド: 約80%, 16スレッド: 約70%)
- 今回はSpMVのルーチンを作業対象とする

FrontISTRのSpMVのルーチン

```
do blockNum
  do i
    do j
    enddo
    do j
    enddo
  enddo
enddo
```

下三角行列の演算ループ

上三角行列の演算ループ

節点ループ

```
do blockNum = 0 , numOfBlockPerThread - 1
  blockIndex = blockNum * numOfThread   + threadNum
  do i = startPos(blockIndex), endPos(blockIndex)
    X1= X(3*i-2)
    X2= X(3*i-1)
    X3= X(3*i   )
    YV1= D(9*i-8)*X1 + D(9*i-7)*X2 + D(9*i-6)*X3
    YV2= D(9*i-5)*X1 + D(9*i-4)*X2 + D(9*i-3)*X3
    YV3= D(9*i-2)*X1 + D(9*i-1)*X2 + D(9*i   )*X3
         :
         : (省略)
         :
    Y(3*i-2)= YV1
    Y(3*i-1)= YV2
    Y(3*i   )= YV3
  enddo
enddo
```

節点ループ

回転数：84,056
ロード：12 (X:3, D:9)
ストア：3 (Y:3)
FLOP：15 (*:9, +:6)

```
jS= indexL(i-1) + 1
jE= indexL(i   )
do j= jS, jE
  in   = itemL(j)
  X1= X(3*in-2)
  X2= X(3*in-1)
  X3= X(3*in   )
  YV1= YV1 + AL(9*j-8)*X1 + AL(9*j-7)*X2 + AL(9*j-6)*X3
  YV2= YV2 + AL(9*j-5)*X1 + AL(9*j-4)*X2 + AL(9*j-3)*X3
  YV3= YV3 + AL(9*j-2)*X1 + AL(9*j-1)*X2 + AL(9*j   )*X3
enddo
```

下三角行列の演算ループ

回転数：1,015,956
ロード：12 (X:3, AL:9)
ストア：0
FLOP：18 (*:9, +:9)

```
jS= indexU(i-1) + 1
jE= indexU(i   )
do j= jS, jE
  in   = itemU(j)
  ! if (async_matvec_flg .and. in > N) cycle
  X1= X(3*in-2)
  X2= X(3*in-1)
  X3= X(3*in   )
  YV1= YV1 + AU(9*j-8)*X1 + AU(9*j-7)*X2 + AU(9*j-6)*X3
  YV2= YV2 + AU(9*j-5)*X1 + AU(9*j-4)*X2 + AU(9*j-3)*X3
  YV3= YV3 + AU(9*j-2)*X1 + AU(9*j-1)*X2 + AU(9*j   )*X3
```

上三角行列の演算ループ

回転数：1,015,956
ロード：12 (X:3, AL:9)
ストア：0
FLOP：18 (*:9, +:9)

節点リオーダリングによって，
行列の演算における節点番号アクセスリストの
データ参照の局所性が向上し，
キャッシュ効果が向上することが期待される

ルーフラインモデルによるSpMVルーチンのピーク性能比

matvec 対象ループ	回転数	ロード (8byte)	ストア (8byte)	1 回転あたり		total	
				要求 Byte	FLOP	要求 Byte	FLOP
blockNum+i	84,056	12	3	120	15	10,086,720	1,260,840
j1	1,015,956	12	0	96	18	97,531,776	18,287,208
j2	1,015,956	12	0	96	18	97,531,776	18,287,208
					total	205,150,272	37,835,256

FX10 のハードウェア性能（16 コア）	
理論ピーク性能	236.5 Gflops/CPU
理論バンド幅	85 GB/s/CPU
理論 B/F（倍精度）	0.36
実効 B/F	0.29

要求 B/F

5.42

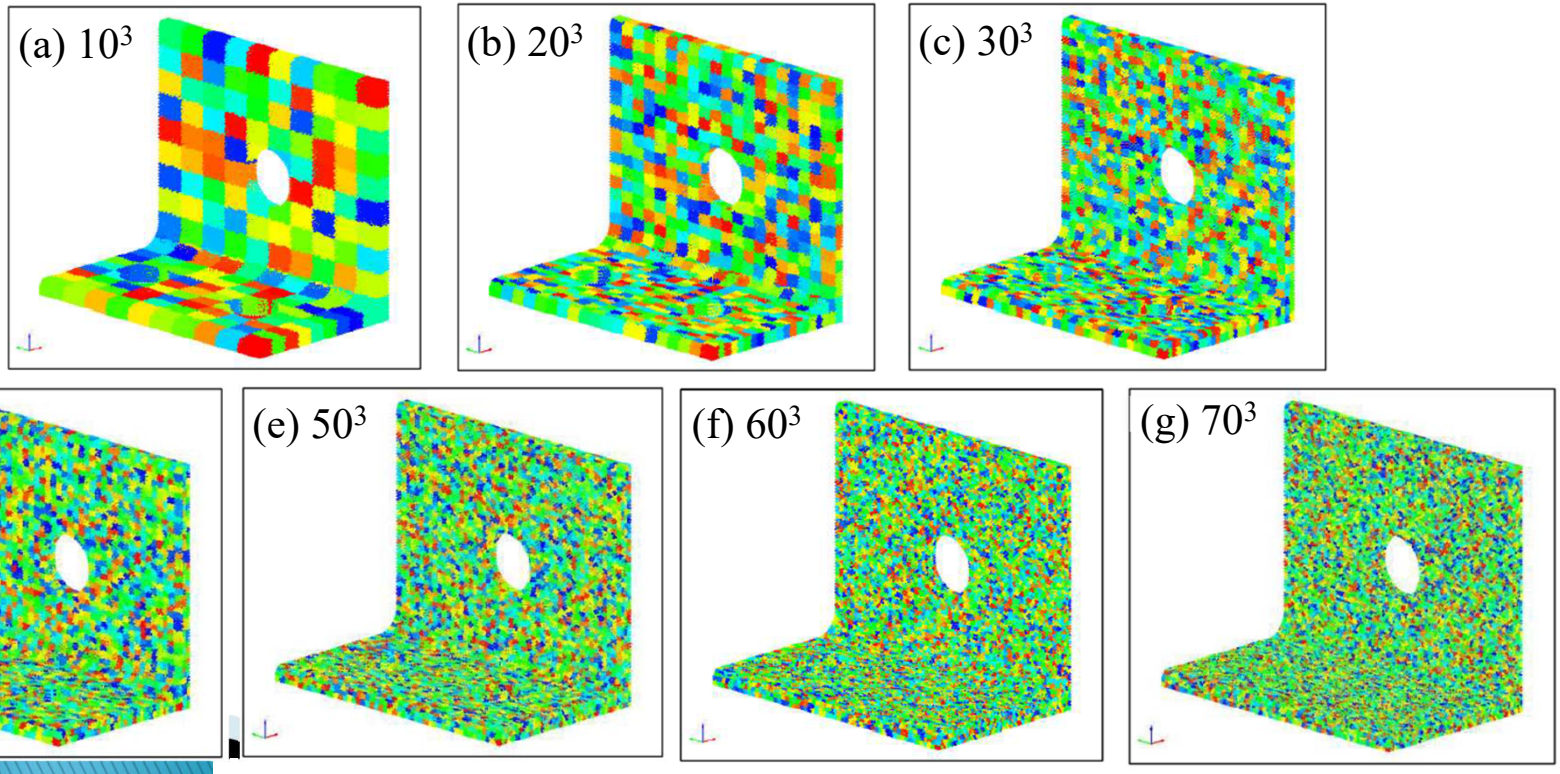
ルーフラインモデルによるピーク性能比

0.29/5.42 = **5.35%**

節点データ並べ替えツールを用いたブロック分割の結果

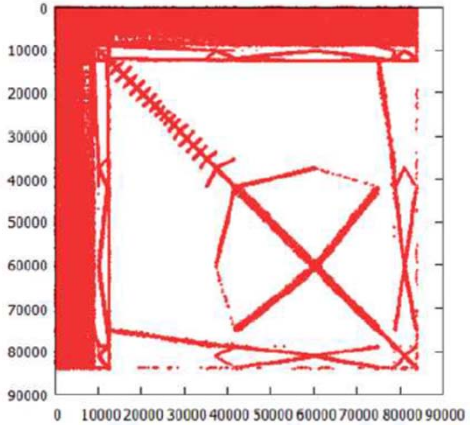
設定したブロック分割数に対するブロック数および節点数

分割数		10^3	20^3	30^3	40^3	50^3	60^3	70^3
ブロック数		190	1475	4213	8951	16520	29815	43409
節点数	平均	442.40	56.99	19.95	9.39	5.09	2.82	1.94
	最大	707	145	84	48	34	20	15
	最小	14	1	1	1	1	1	1

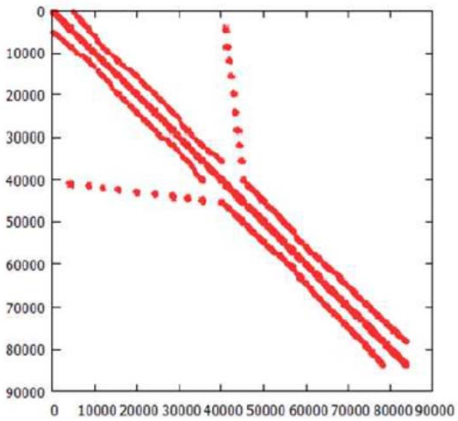


設定したブロック分割数とブロック

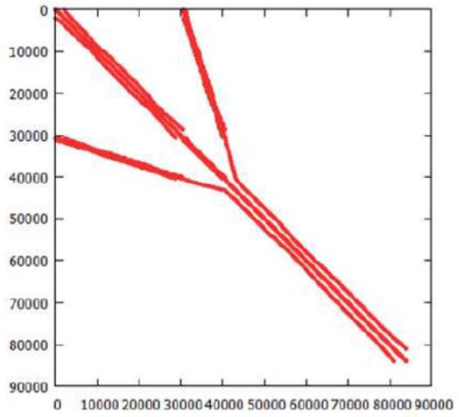
SpMVルーチンの節点番号アクセスリスト



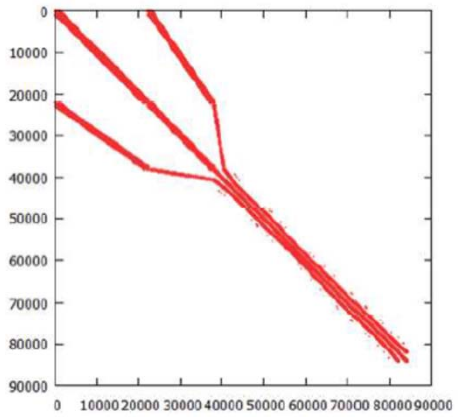
(a) リオーダリング無し



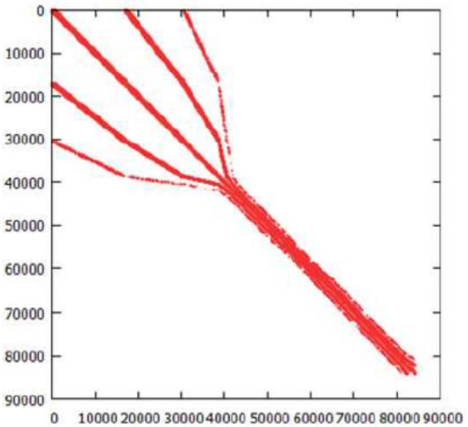
(b) 設定したブロック
分割数10^3



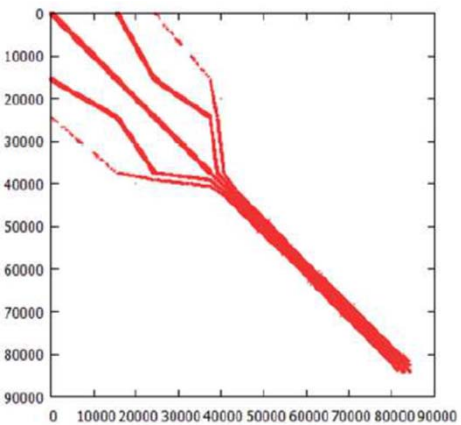
(c) 設定したブロック
分割数20^3



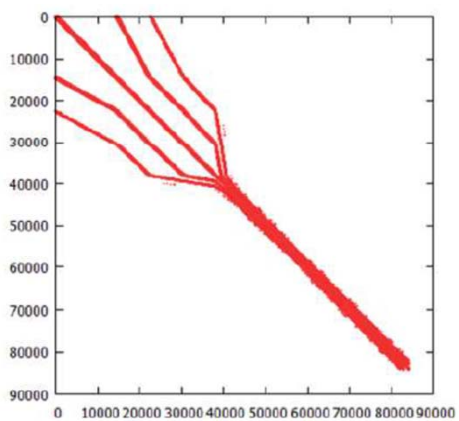
(d) 設定したブロック
分割数30^3



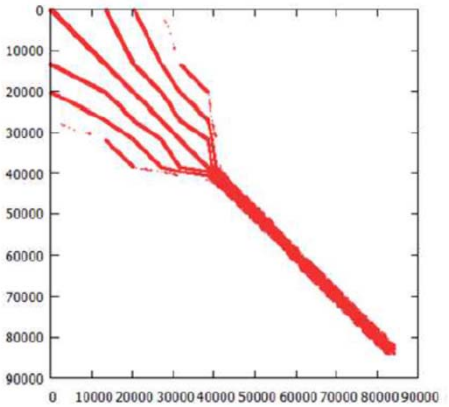
(e) 設定したブロック
分割数40^3



(f) 設定したブロック
分割数50^3



(g) 設定したブロック
分割数60^3

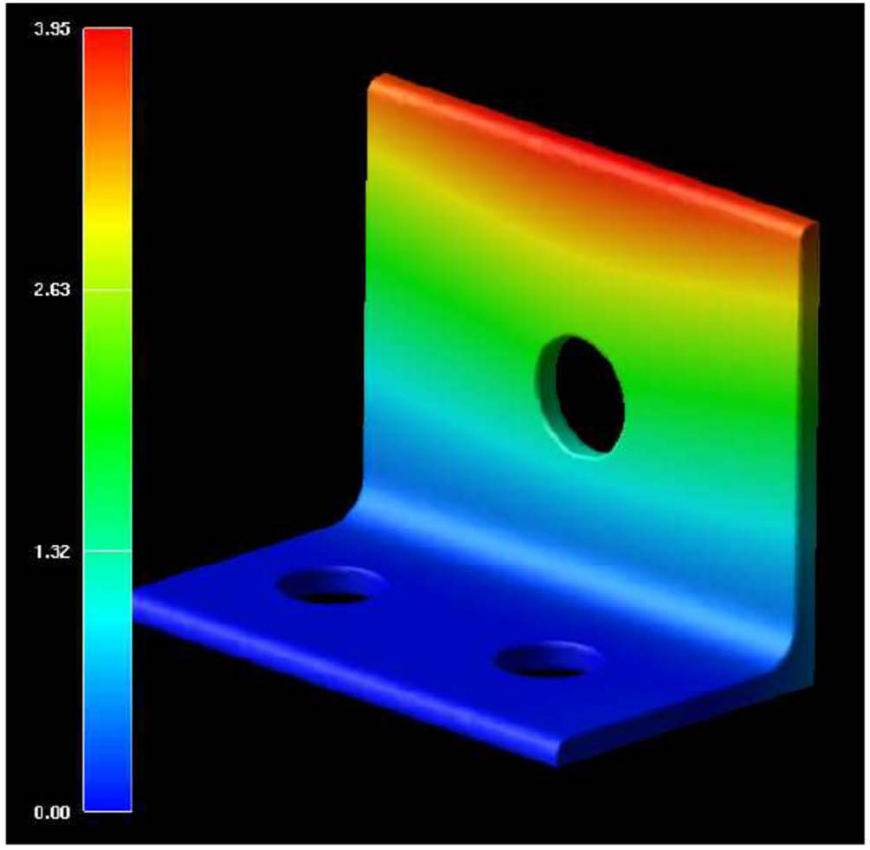


(h) 設定したブロック
分割数70^3

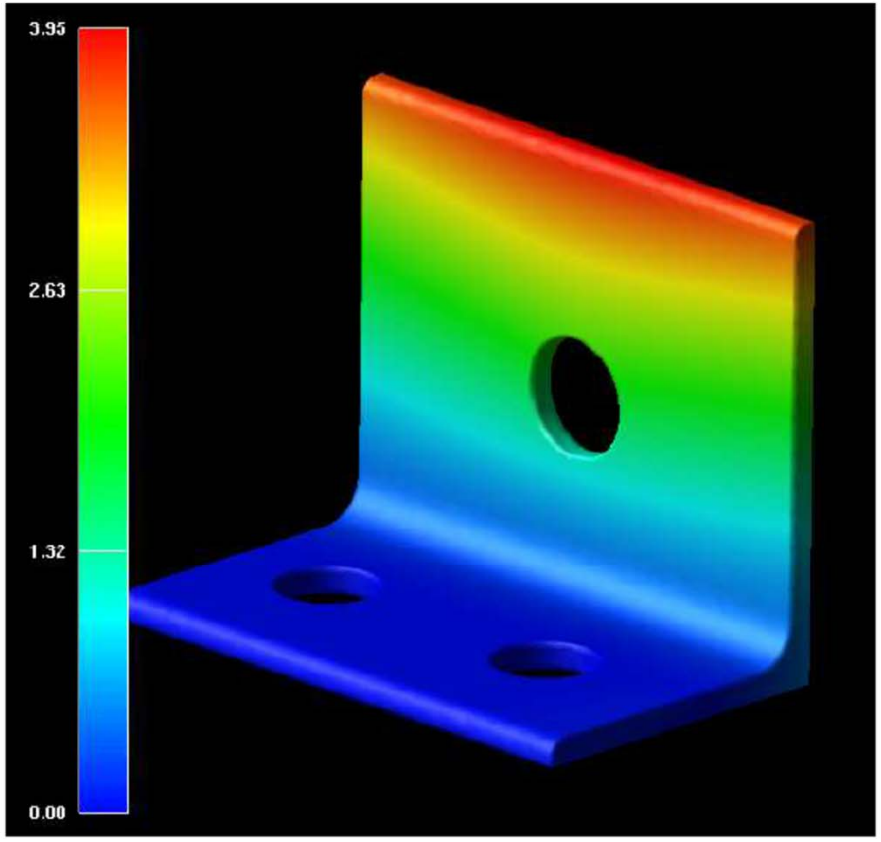
SpMVルーチンの節点番号アクセスリスト (縦軸と横軸は節点番号)

節点データ並べ替えツールを用いた場合の変位分布

リオーダリング無しの場合とリオーダリング有りの場合の
変位分布やMises応力分布は良好に一致することを確認



(a) リオーダリング無し



(b) リオーダリング有り
(設定したブロック分割数50^3)

リオーダリング有りの場合と無しの場合の変位分布

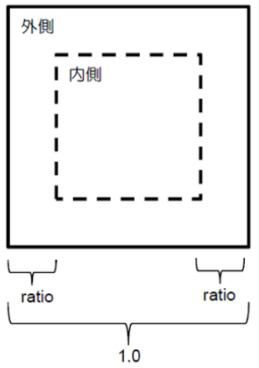
OpenMPスレッド並列計算における実行時間と演算性能 (1/2)

スレッド並列数16での実行時間および演算性能 (詳細プロファイラの結果)

(a) 内外の分割比ratio=0.1の場合

リオーダ
リング無し

設定したブロック分割数	反復回数	実行時間 (sec)	実行時間 /iter (sec)	高速化率 /iter	浮動小数点演算ピーク比	MFLOPS
original	2099	13.7249	6.5388E-03	100.00%	5.02%	11871
10^3	2081	13.0678	6.2796E-03	104.13%	5.23%	12364
20^3	2065	12.7229	6.1612E-03	106.13%	5.33%	12602
30^3	2061	12.5300	6.0796E-03	107.55%	5.40%	12773
40^3	2058	12.3659	6.0087E-03	108.82%	5.46%	12927
50^3	2067	12.2989	5.9501E-03	109.89%	5.52%	13057
60^3	2054	12.4259	6.0496E-03	108.09%	5.43%	12845
70^3	2056	12.7150	6.1844E-03	105.73%	5.31%	12566



(b) 内外の分割なしの場合

リオーダ
リング無し

設定したブロック分割数	反復回数	実行時間 (sec)	実行時間 /iter (sec)	高速化率 /iter	浮動小数点演算ピーク比	MFLOPS
original	2099	13.7249	6.5388E-03	100.00%	5.02%	11871
10^3	2094	12.9473	6.1830E-03	105.75%	5.31%	12556
20^3	2069	12.6427	6.1105E-03	107.01%	5.37%	12708
30^3	2070	12.5491	6.0624E-03	107.86%	5.42%	12811
40^3	2067	12.3876	5.9930E-03	109.11%	5.48%	12962
50^3	2066	12.2856	5.9466E-03	109.96%	5.52%	13065
60^3	2061	12.4688	6.0499E-03	108.08%	5.43%	12844
70^3	2064	12.7650	6.1846E-03	105.73%	5.31%	12565

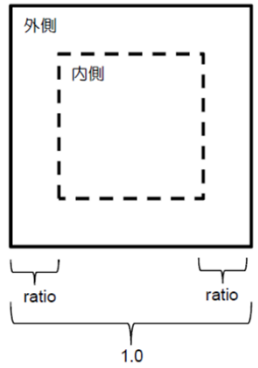
OpenMPスレッド並列計算における実行時間と演算性能 (2/2)

スレッド並列数16でのキャッシュ関連の結果 (詳細プロファイラの結果)

(a) 内外の分割比ratio=0.1の場合

リオーダ
リング無し

	メモリ 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	L1D ミス数	L1D ミス dm 率 (/L1D ミス数)	浮動小数点ロード キャッシュアクセス待ち (sec)
original	52.75	65.42%	53.11%	31.39%	7.97E+09	33.10%	3.47E+00
10^3	54.90	68.16%	53.79%	32.38%	7.50E+09	29.90%	2.83E+00
20^3	56.01	69.60%	53.90%	32.74%	7.18E+09	27.44%	2.56E+00
30^3	56.77	70.53%	54.11%	32.94%	7.03E+09	24.93%	2.34E+00
40^3	57.39	71.24%	53.99%	33.13%	6.81E+09	22.58%	2.12E+00
50^3	57.94	71.95%	54.12%	33.32%	6.73E+09	21.09%	1.96E+00
60^3	57.00	70.88%	53.66%	32.89%	6.81E+09	22.23%	2.03E+00
70^3	55.75	69.22%	53.32%	32.51%	7.05E+09	24.55%	2.40E+00



(b) 内外の分割なしの場合

リオーダ
リング無し

	メモリ 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	L1D ミス数	L1D ミス dm 率 (/L1D ミス数)	浮動小数点ロード キャッシュアクセス待ち (sec)
original	52.75	65.42%	53.11%	31.39%	7.97E+09	33.10%	3.47E+00
10^3	55.73	69.14%	54.13%	32.81%	7.41E+09	28.38%	2.63E+00
20^3	56.47	70.15%	53.94%	32.93%	7.08E+09	26.13%	2.45E+00
30^3	56.92	70.74%	54.01%	32.97%	6.99E+09	24.42%	2.30E+00
40^3	57.55	71.45%	54.11%	33.25%	6.83E+09	22.45%	2.12E+00
50^3	58.00	71.99%	54.18%	33.31%	6.73E+09	21.15%	1.97E+00
60^3	57.00	70.80%	53.68%	32.84%	6.83E+09	22.20%	2.02E+00
70^3	55.75	69.30%	53.34%	32.51%	7.08E+09	24.59%	2.41E+00

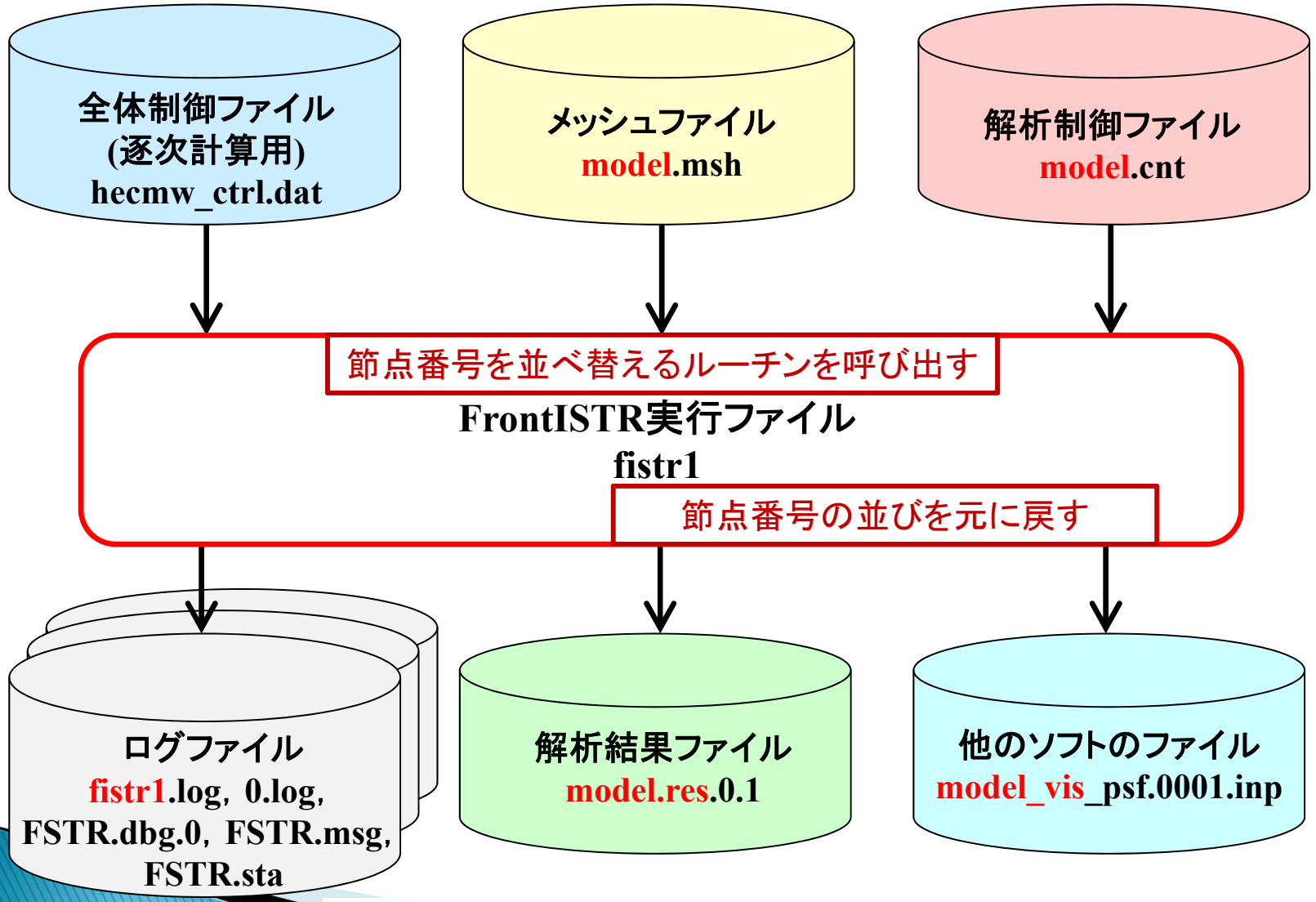
新機能紹介:
節点データ並べ替え処理のFrontISTRプログラムへの移植

発表内容

1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

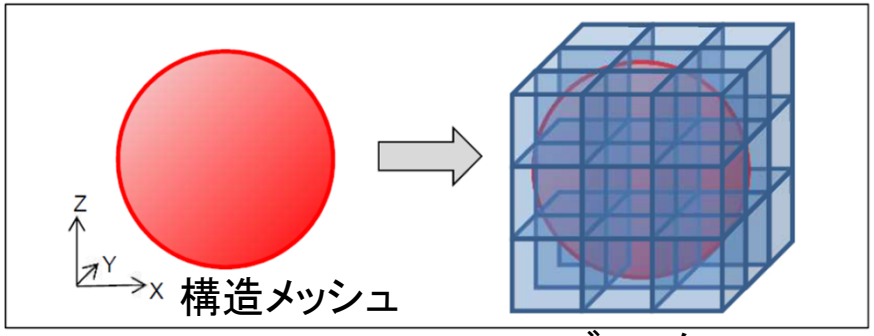
節点並べ替え処理を移植したFrontISTRの逐次計算の流れ

実行方法はこれまでと同じ

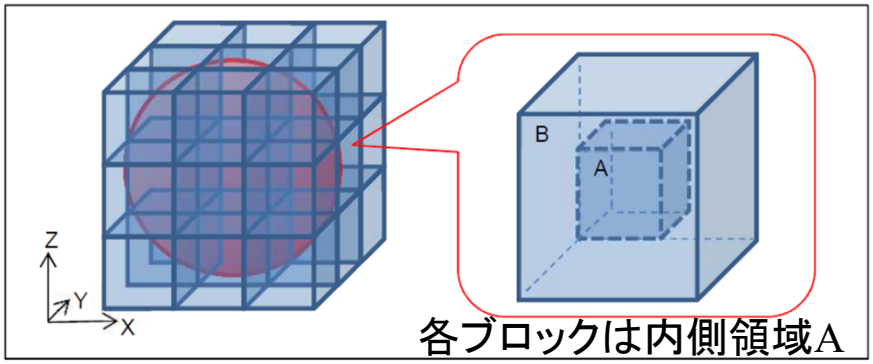


※ 赤字の名前は自由に変更可能

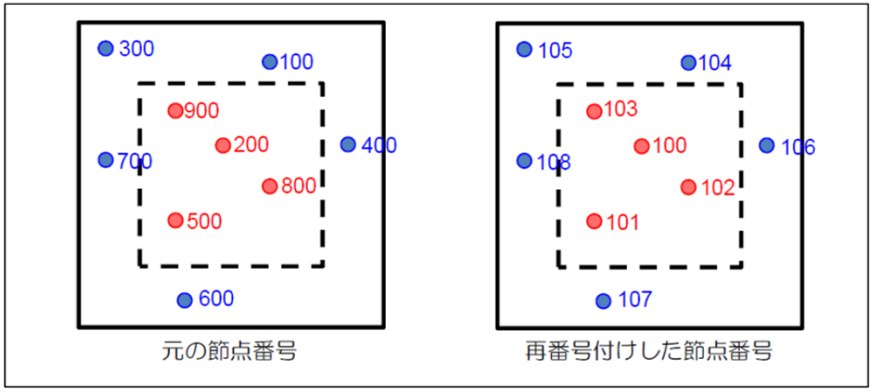
節点データ並べ替え処理 (リオーダリング処理)



(1) 構造メッシュ (単一領域メッシュ または分散領域メッシュ) の バウンディングボックスを 均等なブロックで分割



(2) ブロック順に，各ブロックに対して 節点番号のリナンバリング

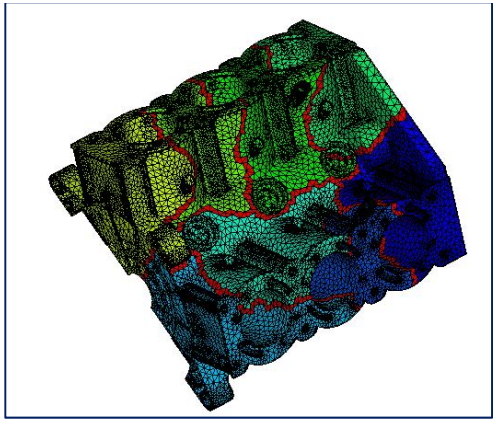


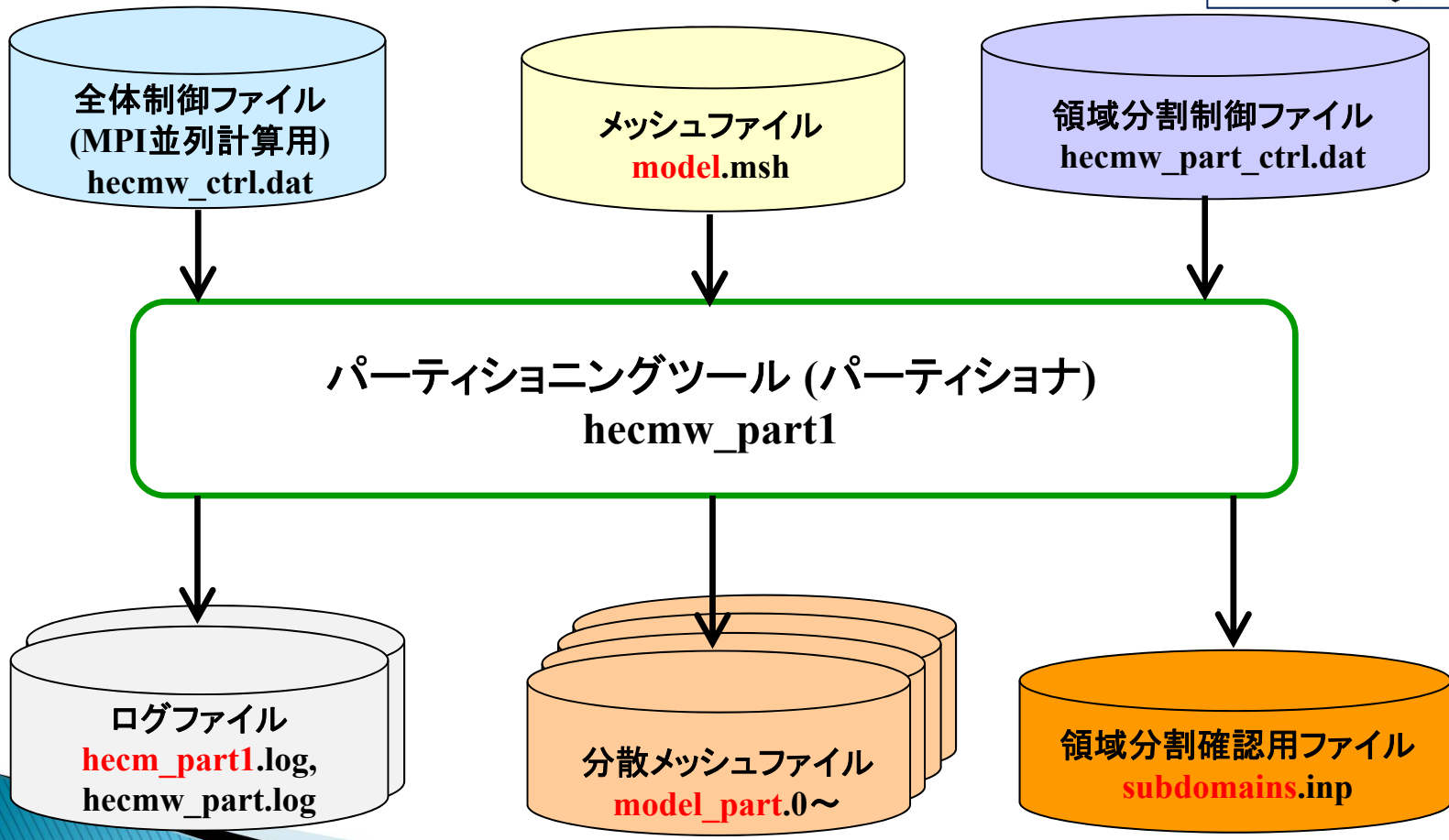
内側領域A，外側領域Bの順に節点番号を詰める

リナンバリングのルール
(a) 節点座標の値と無関係に リナンバリング
(b) 各ブロックに対して， まず内側領域Aの番号を詰め， 次に外側領域Bの番号を詰める ようにリナンバリング

(1) と (2) をFrontISTRプログラム の中で行う

節点並べ替え処理を移植した FrontISTRの並列計算の流れ (1/2)

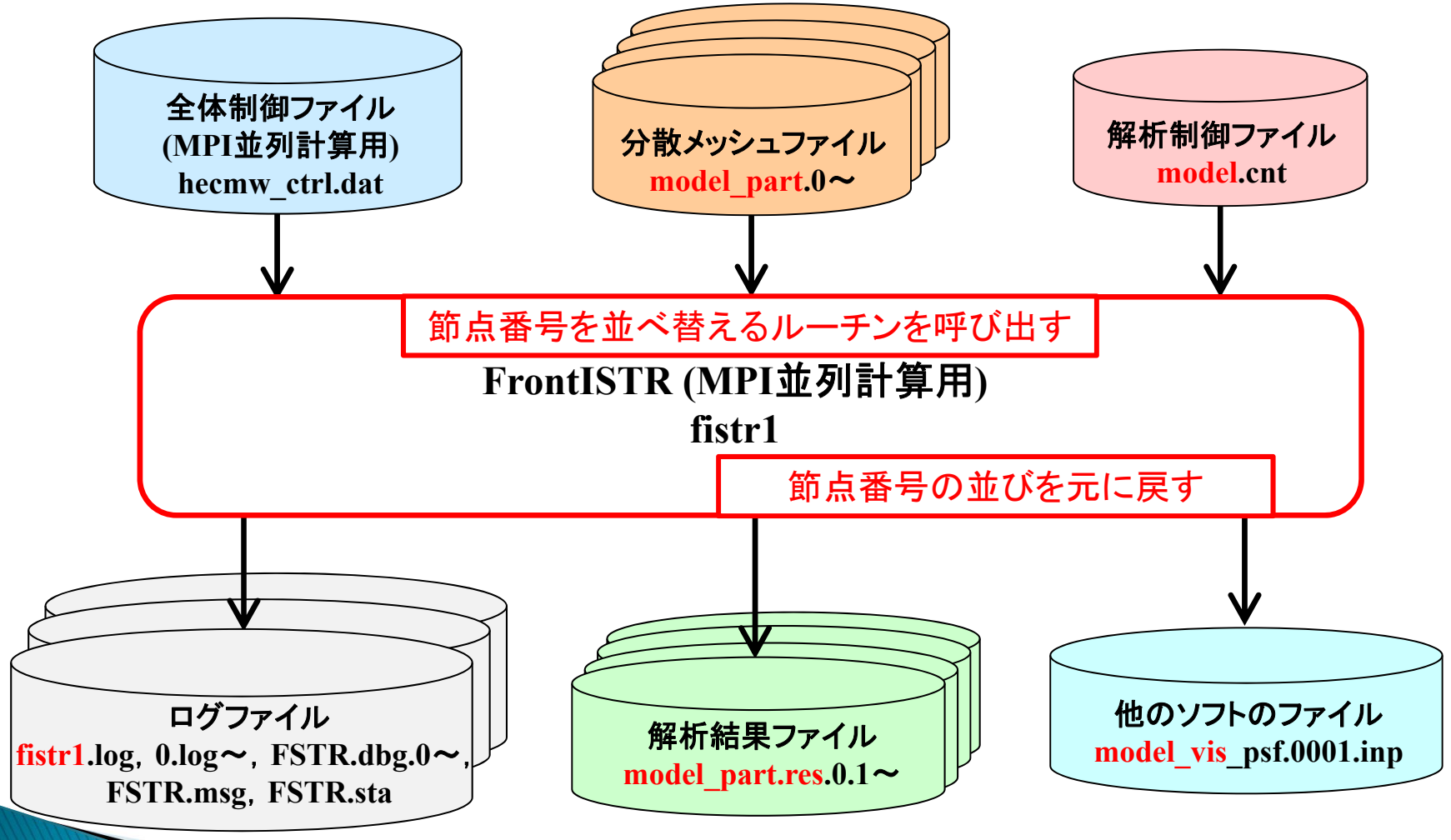




※ 赤字の名前は自由に変更可能

節点並べ替え処理を移植したFrontISTRの並列計算の流れ (2/2)

実行方法はこれまでと同じ



※ 赤字の名前は自由に変更可能

新機能紹介：
節点データ並べ替え処理のFrontISTRプログラムへの移植

発表内容

1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

動作検証に使用する計算機，線形ソルバー，解析メッシュ

使用する計算機

CPU	Intel Xeon E3-1246 v3 3.50GHz
コア数	4
メモリ	32GBytes

使用する線形ソルバー：ブロックSSOR前処理付きCG法

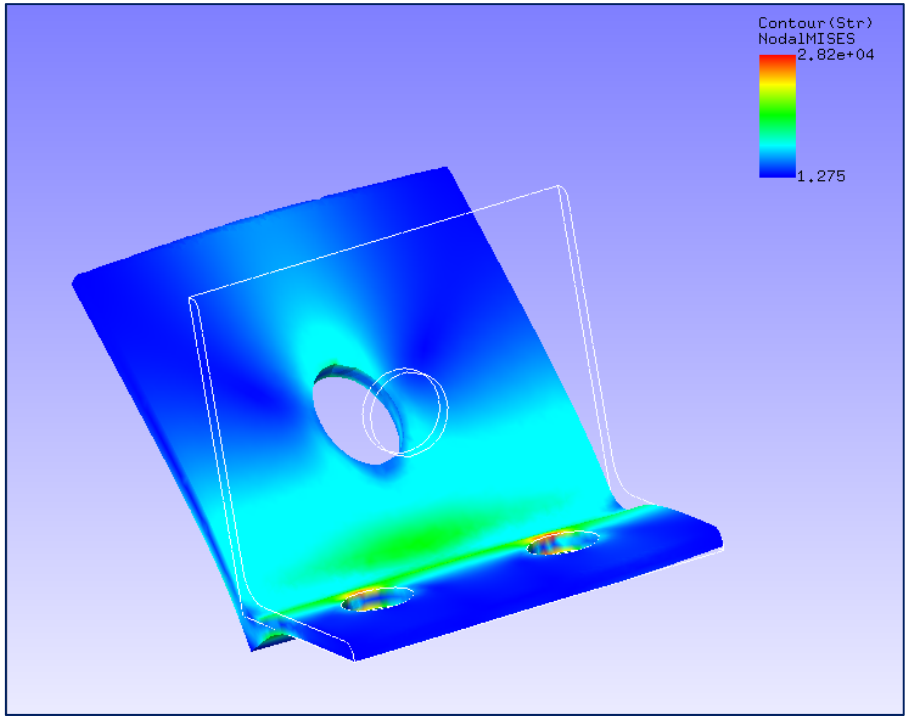
使用する解析メッシュのデータ

hingeモデル

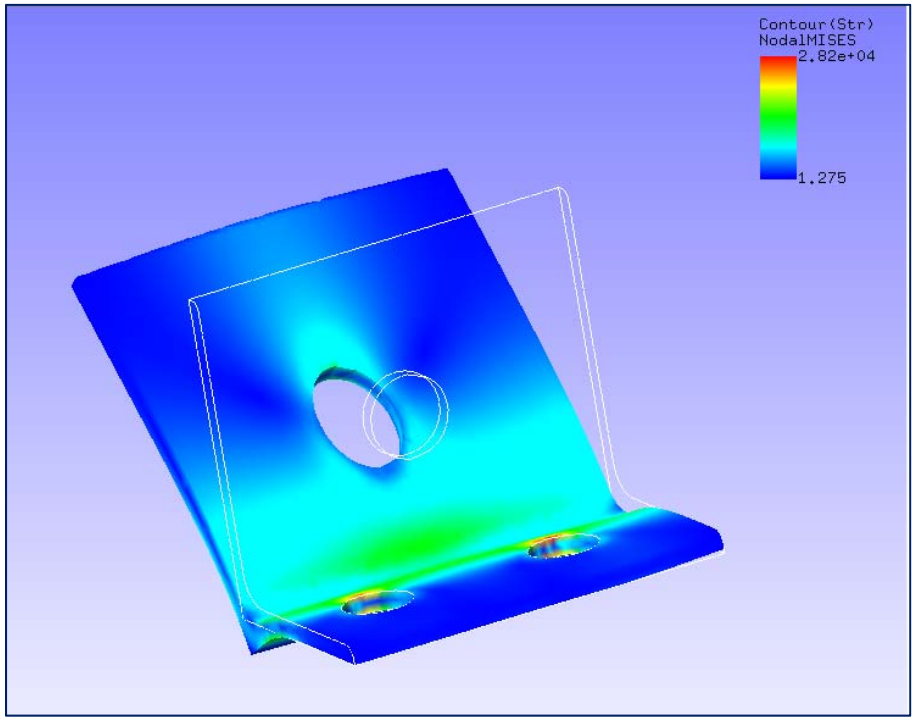
要素数	49,871（TETRA2 次）
節点数	84,056
下三角部分の非零節点数	1,015,956
上三角部分の非零節点数	1,015,956
全体非零節点数	2,115,968

動作検証に使用する計算機，線形ソルバー，解析メッシュ

リオーダリング無しの場合とリオーダリング有りの場合の
変位分布やMises応力分布は良好に一致することを確認



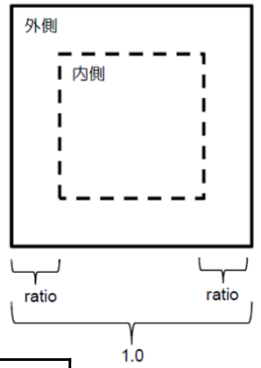
(a) リオーダリング無し，逐次計算



(b) リオーダリング有り，並列計算
(MPIプロセス数4,
設定したブロック分割数20^3)

リオーダリング有りの場合と無しの場合のMises応力分布

フラットMPI並列計算における ブロックSSOR前処理付きCG法の反復数と実行時間



ブロックSSOR前処理付きCG法の反復数

MPIプロセス数	リオーダリング無し	リオーダリング有り (設定したブロック分割20^3, ratio=0.1)	リオーダリング有り／ リオーダリング無し
1	2,099	1,980	0.943
2	2,079	1,977	0.950
4	2,091	1,908	0.912

1反復当たりのブロックSSOR前処理付きCG法の計算時間

MPIプロセス数	リオーダリング無し	リオーダリング有り (設定したブロック分割20^3, ratio=0.1)	リオーダリング有り／ リオーダリング無し
1	0.0608 s	0.0570 s	0.937
2	0.0381 s	0.0361 s	0.947
4	0.0345 s	0.0341 s	0.988

ブロックSSOR前処理付きCG法の計算時間

MPIプロセス数	リオーダリング無し	リオーダリング有り (設定したブロック分割20^3, ratio=0.1)	リオーダリング有り／ リオーダリング無し
1	127.81 s	112.91 s	0.883
2	79.28 s	71.52 s	0.902
4	72.20 s	65.47 s	0.906

新機能紹介:
節点データ並べ替え処理のFrontISTRプログラムへの移植

発表内容

1. 節点データ並べ替えツール
2. OpenMPスレッド並列計算における
   実行時間と演算性能
3. 節点データ並べ替え処理の移植
4. プログラムの動作検証
5. 現状のまとめと今後の課題

現状のまとめ

➢ データ並べ替え処理の導入について

- 節点データ並べ替えツールを用いて，mshファイルに対して
  リオーダリング処理を実施した.
- ユーザビリティやメンテナンス性を考慮して，
  データ並べ替え処理をFrontISTRのプログラム内部に移植した.

➢ 演算性能向上の評価について

- 単一領域メッシュに対してスレッド並列数16で計算し，
  SpMVのみの演算性能を評価した.
  設定したブロック分割数50^3の場合，計算速度が約10%向上し，
  メモリスループットが約10%向上した.
- 分散領域メッシュに対してMPIプロセス数2で計算し，
  ブロックSSOR前処理付きCG法の反復数と計算時間を計測した.
  設定したブロック分割数20^3の場合，反復数が約5%減少し，
  計算速度が約5%向上した.

今後の課題

- 演算性能向上の評価について
  - データ並べ替え処理によって，ブロックSSOR前処理付きCG法の反復数が減少する原因を調査する.
  - データ並べ替え処理に対する前処理の性能も調査する.
- ブロック分割数の調査について
  - hingeモデル以外の解析モデルに対する評価を行い，汎用的な効果が発揮できるブロック分割の決定方法を調査する．そして，特別な経験がないユーザに対して，推奨されるブロック分割数を示す.