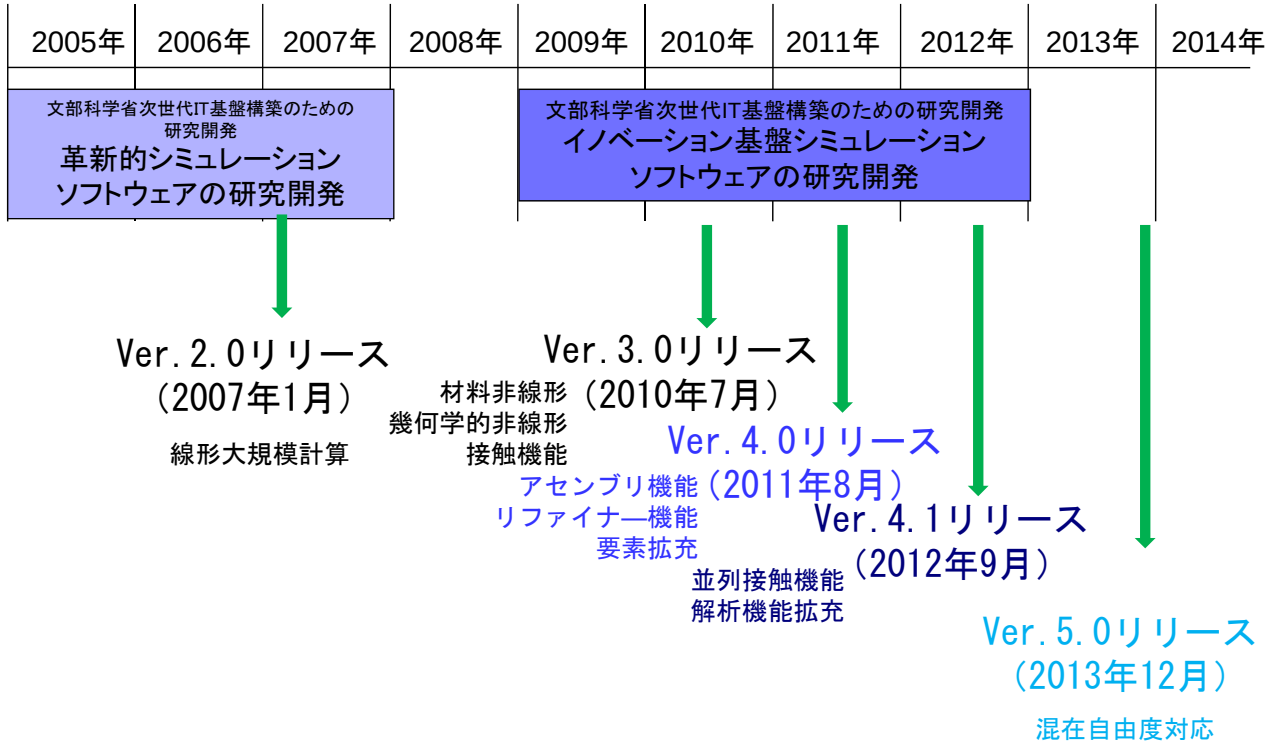
構造解析ソフトウェア Advance/FrontSTRの概要

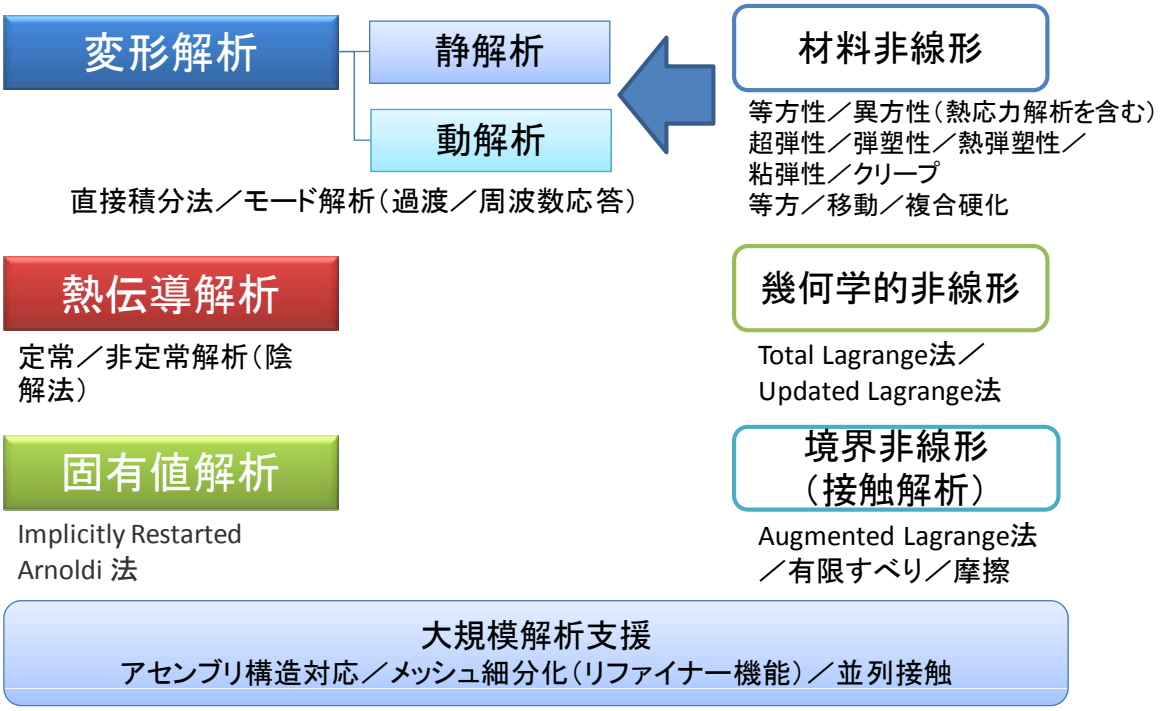
第1事業部　末光 啓二

アドバンスソフト株式会社の
スーパーコンピューティング・サービス セミナー
2015年1月22日（木）
アドバンスソフト株式会社

Advance/FrontSTRの開発経緯



Advance/FrontSTRの機能



UV2000における最適化項目

- ベクトル演算機能の活用
  - AVXアーキテクチャ（従来SSEの倍となる256ビットSIMDレジスタ）
  - 大規模（20MB）L3キャッシュ
- 大規模共有メモリシステムによる大規模モデルの直接解法
  - 115GB/（socket=cpu）
  - 10コア/socket
  - max256socket

256×10のハイブリッド並列で
30TBモデルを直接解法
解析可能な節点数？

ベクトル演算機能の活用

オリジナルコード（行列ベクトル積の一部）

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

改良コード

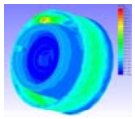
```
do i= 1, hecMAT%N
  jS= hecMAT%indexL(i-1) + 1
  jE= hecMAT%indexL(i  )
  do j= jS, jE
    in  = hecMAT%itemL(j)
    X1= X(3*in-2)
    X2= X(3*in-1)
    X3= X(3*in  )
    Y(3*i-2)= Y(3*i-2) + hecMAT%AL(9*j-8)*X1 + hecMAT%AL(9*j-7)*X2  &
                                             + hecMAT%AL(9*j-6)*X3
    Y(3*i-1)= Y(3*i-1) + hecMAT%AL(9*j-5)*X1 + hecMAT%AL(9*j-4)*X2  &
                                             + hecMAT%AL(9*j-3)*X3
    Y(3*i  )= Y(3*i  ) + hecMAT%AL(9*j-2)*X1 + hecMAT%AL(9*j-1)*X2  &
                                             + hecMAT%AL(9*j  )*X3
  enddo
enddo
do i= 1, hecMAT%N
  jS= hecMAT%indexU(i-1) + 1
  jE= hecMAT%indexU(i  )
  do j= jS, jE
    in  = hecMAT%itemU(j)
    X1= X(3*in-2)
    X2= X(3*in-1)
    X3= X(3*in  )
    Y(3*i-2)= Y(3*i-2) + hecMAT%AU(9*j-8)*X1 + hecMAT%AU(9*j-7)*X2  &
                                             + hecMAT%AU(9*j-6)*X3
    Y(3*i-1)= Y(3*i-1) + hecMAT%AU(9*j-5)*X1 + hecMAT%AU(9*j-4)*X2  &
                                             + hecMAT%AU(9*j-3)*X3
    Y(3*i  )= Y(3*i  ) + hecMAT%AU(9*j-2)*X1 + hecMAT%AU(9*j-1)*X2  &
                                             + hecMAT%AU(9*j  )*X3
  enddo
enddo
```

計算対象	オリジナル	改良	改善率
Hinge	373.6	292.6	1.28
V6engine	246.5	216.5	1.14
Cap	757.1	665.7	1.14

コンパイルオプション-xAVX
の効果はなし

大規模並列処理性能

解析対象
静応力解析（四面体2次要素）



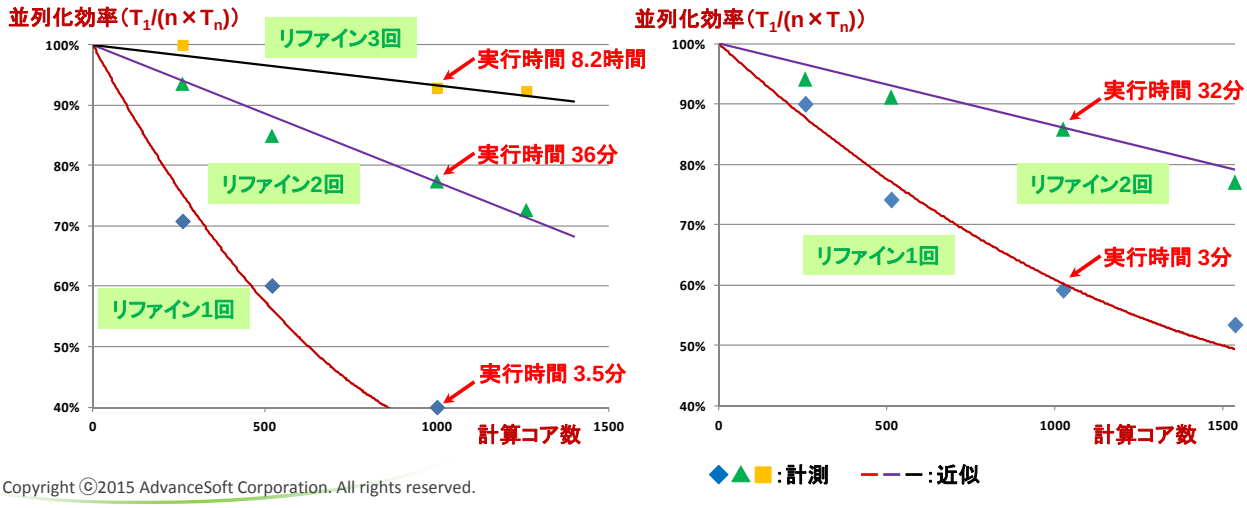
リファイン	要素数	節点数
なし	684,807	1,008,911
1回	5,478,456	7,707,758
2回	43,827,648	60,089,084
3回	350,621,184	474,183,032

JAMSTEC UV2000

Xeon E5-4650v2(2.4GHz)
×1CPU(10コア)／ソケット

東京大学FX10

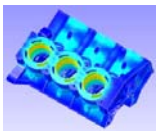
SPARC 64Ixfx(1.848GHz)
×1CPU(16コア)／ノード



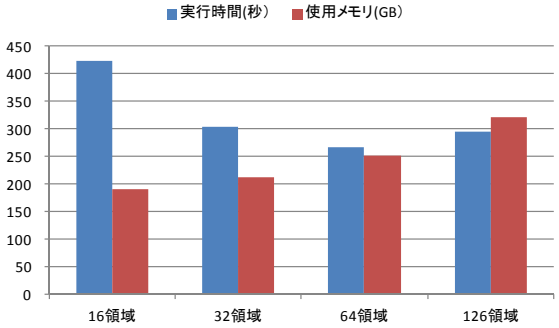
直接法ソルバーによる大規模解析

ケースA

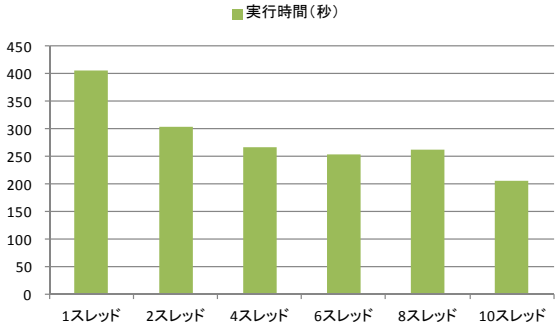
静応力解析（四面体2次要素）
要素数：2,257,024
節点数：3,338,817



プロセス並列性能（4スレッド）

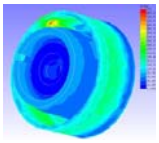


スレッド並列性能（64プロセス）



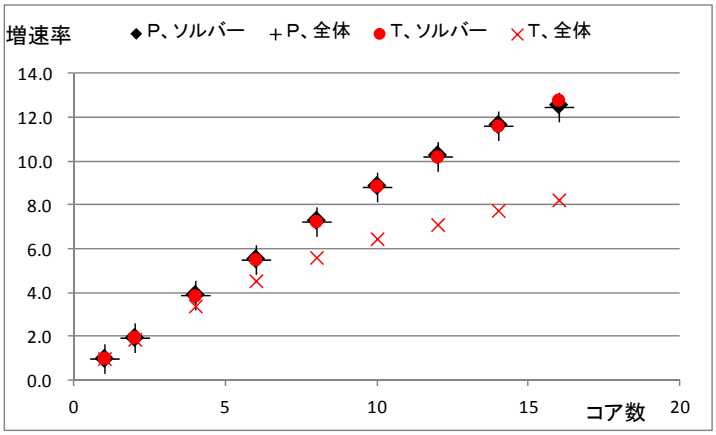
ケースB

静応力解析（四面体2次要素）
要素数：5,478,456　　節点数：7,707,758
MUMPSの必要メモリ量予測では16.4TB
150ソケット程度（max256）で実行可能　＝＞　今後、ジョブ申請



「京」（FX10)における最適化項目

- 負荷が高い反復法ソルバーの高効率なハイブリッド並列
- ハイブリッド並列時のソルバー以外の処理の並列化



CG法内部サブルーティンの負荷バランス

サブルーティン機能	負荷率（%） ー前処理：対角ー	負荷率（%） ー前処理：ILUー
行列ベクトル積	80.4 ⬅	39.4 ⬅
残差総和計算（通信）	7.4	3.5
前処理	4.2	54.0 ⬅
ベクトル更新／収束判定	3.6	0.9
残差内積計算	2.4	0.5
ベクトル交換（通信）	1.7	1.4
その他	0.3	0.3

行列ベクトル積サブルーティンの分析

```
do i = 1, hecMAT%N                    ←―― 節点数による行ループ
  X1 = X(3*i-2)
  X2 = X(3*i-1)        Xは変数ベクトル
  X3 = X(3*i  )
YV1 = hecMAT%D(9*i-8)*X1 + hecMAT%D(9*i-7)*X2 + hecMAT%D(9*i-6)*X3
YV2 = hecMAT%D(9*i-5)*X1 + hecMAT%D(9*i-4)*X2 + hecMAT%D(9*i-3)*X3
YV3 = hecMAT%D(9*i-2)*X1 + hecMAT%D(9*i-1)*X2 + hecMAT%D(9*i  )*X3

  jS = hecMAT%indexL(i-1) + 1
  jE = hecMAT%indexL(i  )
  do j = jS, jE                       ←―― 非零要素数による列ループ
    in  = hecMAT%itemL(j)
    X1 = X(3*in-2)
    X2 = X(3*in-1)
    X3 = X(3*in  )
    YV1= YV1 + hecMAT%AL(9*j-8)*X1 + hecMAT%AL(9*j-7)*X2 + hecMAT%AL(9*j-6)*X3
    YV2= YV2 + hecMAT%AL(9*j-5)*X1 + hecMAT%AL(9*j-4)*X2 + hecMAT%AL(9*j-3)*X3
    YV3= YV3 + hecMAT%AL(9*j-2)*X1 + hecMAT%AL(9*j-1)*X2 + hecMAT%AL(9*j  )*X3
  enddo
  Upper partで同様の計算
  B(3*i-2) = YV1
  B(3*i-1) = YV2        Bは積ベクトル
  B(3*i  ) = YV3
enddo
```

D part

Lower part

行列の要素は連続アドレス
プリフェッチ128byteに対して、72byte/ループ使用

Xは飛びアドレスアクセスになるが、再利用される
＝＞ セクターキャッシュに格納

X以外に、再利用される変数は
X1,X2,X3,in,YV1,YV2,YV3およびプリフェッチ未使用分
＝＞ ノーマルキャッシュ容量はわずかでよい

列ループの回数は行ごとに異なる
＝＞ 列ループの総和が均等になるように行ループを分割

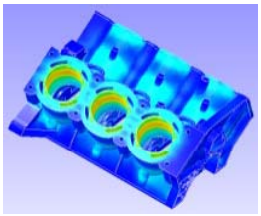
ハイブリッド並列の実装要領

- 行列ベクトル積計算のコーディング
  - !OCL CACHE_SECTOR_SIZE(1,23)
  - !OCL CACHE_SUBSECTOR_ASSIGN(X)
  - !$OMP PARALLEL
  - !$ threadNum = omp_get_thread_num() + 1
  - do i = OMPstart(threadNum), OMPend(threadNum)
    中略
  - enddo
  - !$OMP END PARALLEL
  - !OCL END_CACHE_SUBSECTOR
  - !OCL END_CACHE_SECTOR_SIZE
- 他のサブルーティンのdoループは、単純なブロック分割

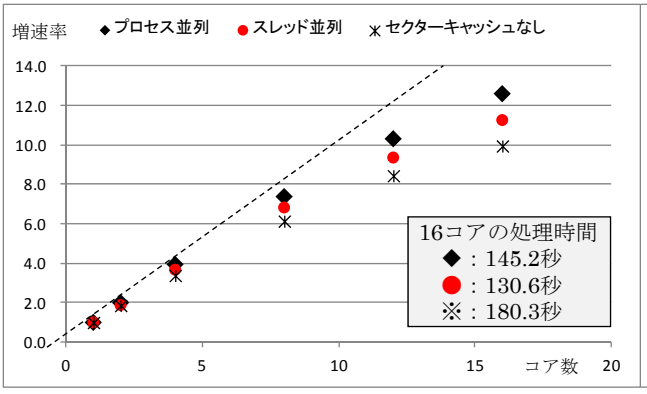
行ループ分割の節点数は予め計算

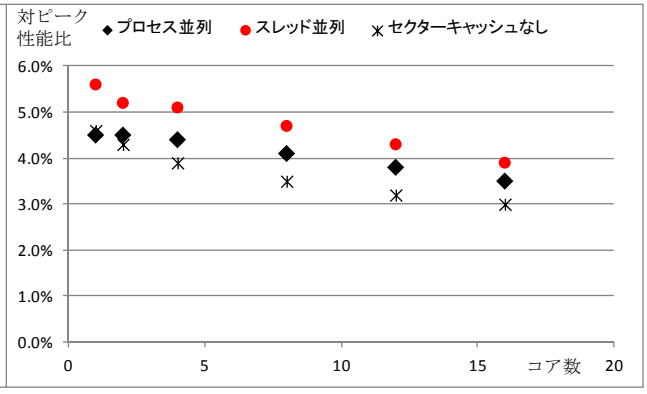
L2キャッシュ12MB・24wayを1:23に分割
セクターキャッシュ23wayを配列変数Xで使用
＝＞最大50万節点の変位ベクトルを格納

最適化されたハイブリッド並列性能



メッシュ緒元
四面体2次要素
要素数：282,128
節点数：459,292





ハイブリッド並列機能の補完