SS研
ポストペタアプリ性能WG資料

FUJITSU
shaping tomorrow with you

3.1 フラグメント分子軌道法「ABINIT-MP」d関数積分ルーチンの結果異常について

2015年3月17日
富士通株式会社
千葉　修一

Copyright 2015 FUJITSU LIMITED

サマリー（１）

FUJITSU

- 課題
  - 最適化オプション -O3により結果異常となる
- 対象コード
  - フラグメント分子軌道法プログラム「ABINIT-MP」
  - d関数積分ルーチン
- 事象
  - d関数積分ルーチンの影響で以降の処理に問題が発生
  - FMO計算の1stステップである Monomer-SCCで収束せず
  - 最適化オプション-O2では問題なし

Copyright 2015 FUJITSU LIMITED

サマリー（２）

FUJITSU

- 回答
  - 結果異常の原因
    - コンパイラの障害
  - 対処
    - 修正提供済み、保守日などの関係で次期に違いあり
    - 「京」16版　　(2014年3月)
    - FX10　V1.0L30　(2013年9月)
  - コンパイラへのフィードバック
    - コンパイラのバージョンの判り難さを改善（中期課題）

Copyright 2015 FUJITSU LIMITED

障害内容

FUJITSU

- 障害番号 PH04880
- 発生条件

  以下の要因の複合条件により発生する場合がある
  - Fortran/C/C++において、最適化オプション -O3を利用
  - ループ内に子ループが存在し、制御変数以外に変数(以降,n)が存在
  - 子ループでnが更新、またはnを添え字とする配列(以降,b)を定義
  - ループ外の分岐においてbが参照

サンプルコード

```
do i=1,m
n = 0
  do j=0,1
    n = n + 1
    b(n) =  1
  enddo
  if (c(1) < 10) then
    a(1) =  b(1)
  endif
enddo
```

- 内容

  配列の添え字の更新処理を誤って最適化
- 回避方法

  最適化オプション -O2を利用

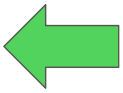
Copyright 2015 FUJITSU LIMITED

分析

FUJITSU

- コンパイラの調査
  - FX10向け最新コンパイラで結果異常がないことを確認
  - 「京」向け標準コンパイラで結果異常を確認
  - Technical Computing Suite V1.0L30相当で修正された障害
- コンパイラのバージョン情報(frtpx -V)
  - 最新コンパイラ / FX10　－　問題なし

frtpx: Fujitsu Fortran Driver Version 1.2.1
P-id: T01641-04 (Jul 10 2014 14:29:18)



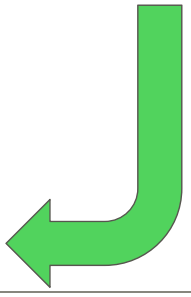
バージョン表記が異なり判断がつきにくい

  - 標準コンパイラ / 「京」　－　問題あり

frtpx: Fujitsu Fortran Driver Version 1.2.0
P-id: L30000-07 (Jul 30 2013 08:57:22) K-1.2.0-15

  - 標準コンパイラ / 「京」　－　問題なし

frtpx: Fujitsu Fortran Driver Version 1.2.0
P-id: L30000-08 (Dec 5 2013 15:09:10) K-1.2.0-16-3



Copyright 2015 FUJITSU LIMITED

対処

FUJITSU

- コンパイラの変更
  - 「京」においては、FX10と同等バージョンのコンパイラを利用することで対処
  - 「京」とFX10 の対応付けについては利用手引書で開示

    「利用手引書(1.35版)」では「17.「 京」と FX10 の言語環境」
  - 環境を変更させることで対応

    「利用手引書(1.35版)」では「4.3.コンパイル環境設定」

```
% source /home/system/Env_base_1.2.0-16-3
```


Copyright 2015 FUJITSU LIMITED

コンパイラへのフィードバック

FUJITSU

- バージョン表記の統一
  - バージョンを「京」、一般展開機でバージョン情報を統一
  - バージョン表記で環境の一致性を判断
  - 開発環境のポータビリティの向上へつなげる
- チェックコンパイラの利用促進
  - 結果異常障害については、チェックコンパイラを提供済み
  - 利用を促進することで早期の自己解決へ

Copyright 2015 FUJITSU LIMITED

FUJITSU

shaping tomorrow with you

SS研
ポストペタアプリ性能WG資料

FUJITSU
shaping tomorrow with you

3.2 核融合プラズマ流体解析「GKV」SSL2の性能改善について

2015年3月17日
富士通株式会社
千葉　修一

Copyright 2015 FUJITSU LIMITED

サマリー（１）

FUJITSU

- 課題
  - SSL2とFFTWで性能が大幅に異なる
- 対象コード
  - 核融合プラズマ流体解析「GKV」
  - ２次元配列(256x256)に対するマルチストリーム1DFFT
- 検討事項
  - 特に実変換の性能が大幅に異なる
  - 実変換を複素変換で実装することで性能改善か可能

Copyright 2015 FUJITSU LIMITED

サマリー（２）

FUJITSU

■回答

■対応

- DVMRF2について改善を検討
- 現時点では、DVCFM1で扱う形式に帰着させる方式とした方 が、DVMRF2を利用する形式より高い性能

■DVMRF2 の実装状況

- FFT計算カーネル部分は、任意のFFT長さに対応
- SIMD化などのチューニングは、未対応

■DVCMF1 の実装状況 ※ 2,3,5,7混合基底

- FFT計算カーネル部分は、任意のFFT長さに未対応
- SIMD化などのチューニングは、対応済み

Copyright 2015 FUJITSU LIMITED

性能データ

FUJITSU

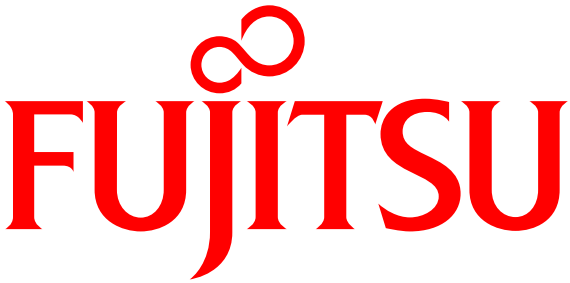
■演算ピーク性能

- SSL2　　：DVCFM1(複素－複素変換)、DVMRF2(実－複素変換)
- SSL2_rev：DVCFM1(複素－複素変換)、DVCFM1(実－複素変換)

演算ピーク比(%)	SSL2	SSL2_rev	FFTW3_in	FFTW3_out
複素ー複素逆変換	30.04	30.04	20.91	33.23
複素ー複素順変換	29.07	29.07	18.88	29.29
複素ー実逆変換	7.74	22.30	16.39	25.20
実ー複素変換	5.04	27.52	19.32	26.35

「日本原子力研究開発機構) 井戸村様 SS研 ポストペタアプリ性能WG 第４回 報告資料」より

Copyright 2015 FUJITSU LIMITED



shaping tomorrow with you

SS研
ポストペタアプリ性能WG資料

FUJITSU
shaping tomorrow with you

3.3 圧縮流体解析「UPACS-L」 tune1コード:プレ処理の性能劣化について

2015年3月17日
富士通株式会社
千葉　修一

Copyright 2015 FUJITSU LIMITED

サマリー（１）

FUJITSU

- 課題
  - 構造体の変更(tune1)により構造体の初期化処理の性能が劣化
- 対象コード
  - 圧縮流体解析「UPACS-L」
  - viscous-pre（プレ処理：構造体の初期化）
- チューニング
  - 構造体×３次元　→　構造体の各メンバ×３次元
    - type cellFaceType
    - type visCellFaceType

Copyright 2015 FUJITSU LIMITED

サマリー（２）

FUJITSU

■回答

■性能低下の原因

- 配列宣言の変化により並列化される次元が変化
- 並列化された次元が回転数が少なく、ロードインバランスが発生

■対処

- オプションで回転数不足のループを並列化の対象から除外
- 内側の次元を並列化することでオリジナルと同等性能を達成
- ただし最内の回転数が少ないためxfillの効果が低い

区間(sec)	original	tune1	対処済
Viscous_pre	0.51	1.66	0.50

※社内機での実測値

Copyright 2015 FUJITSU LIMITED

分析

FUJITSU

■対象コード

original

```
!omp parallel do private(i,j,k)
 do k=-1,blk%kn
  do j=-1,blk%kn
   do i=-1,blk%kn
    ...
    cface(i,j,k)%nv(:) =0.0
    ...
```

tune1

```
...
cface%nv = 0.0
...
```

doループを
明示的に書いても
性能は変化せず

■性能データ

区間(sec)	original	tune1
Viscous_pre	0.5870	1.720

「宇宙航空研究機構) 高木様 SS研 ポストペタアプリ性能WG 第３回 報告資料」より

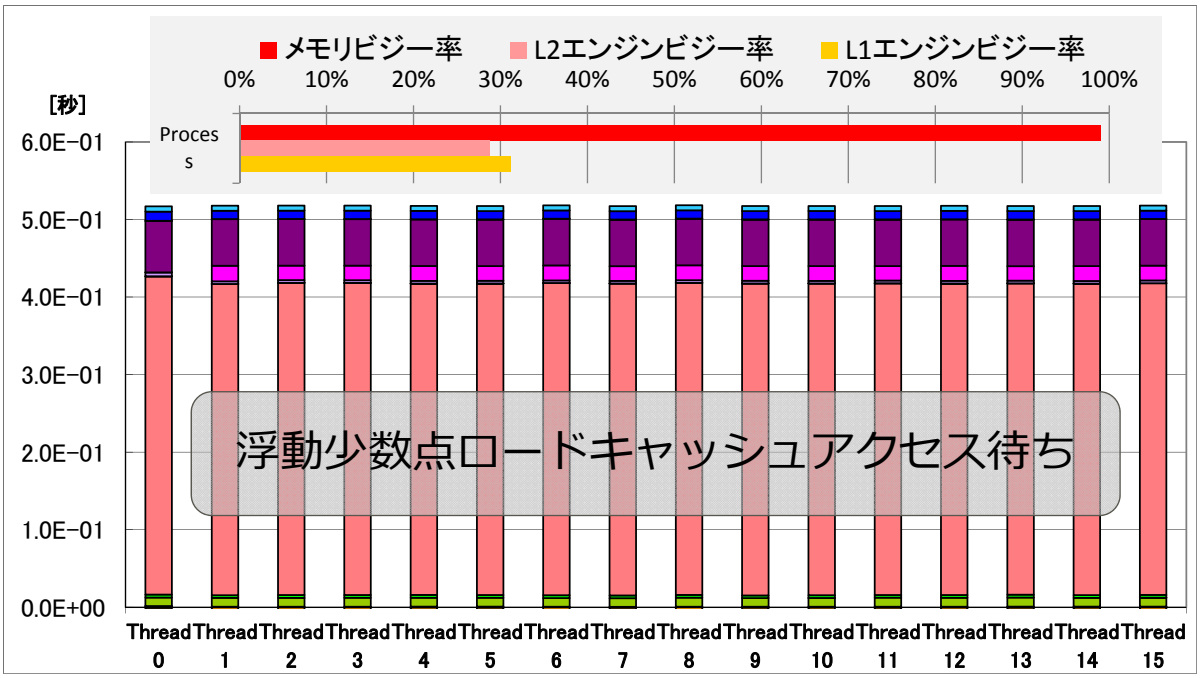
Copyright 2015 FUJITSU LIMITED

分析（１）

FUJITSU

originalの詳細PA情報

各スレッドで均等に処理が実行



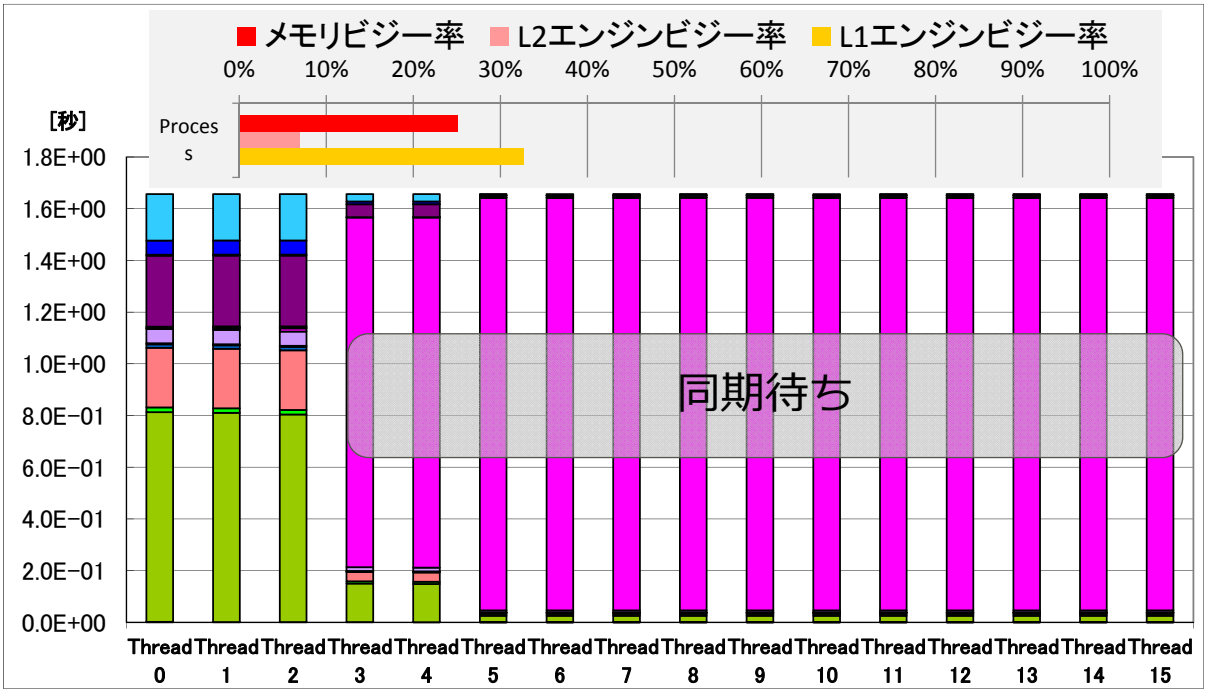
Copyright 2015 FUJITSU LIMITED

分析（２）

FUJITSU

tune1の詳細PA情報

ロードインバランスが発生している



Copyright 2015 FUJITSU LIMITED

分析（３）

FUJITSU

■コードの分析

- ■1から3次元目までは 161要素
- ■4次元目以降の最終次元は 3から5要素、特に3が多い

```
type visCellFaceType
  ...
  real(8), pointer, dimension(:,:,:,:)    :: nv
  ...
end type
type(visCellFaceType) :: cface
allocate( cface%area(-1:blk%in,-1:blk%jn,-1:blk%kn ))
allocate( cface%nv   (-1:blk%in,-1:blk%jn,-1:blk%kn,3 ))
allocate( cface%flux(-1:blk%in,-1:blk%jn,-1:blk%kn,5 ))
allocate( cface%mu   (-1:blk%in,-1:blk%jn,-1:blk%kn ))
allocate( cface%dTdx(-1:blk%in,-1:blk%jn,-1:blk%kn,3 ))
allocate( cface%u    (-1:blk%in,-1:blk%jn,-1:blk%kn,3 ))
allocate( cface%dudx(-1:blk%in,-1:blk%jn,-1:blk%kn,3,3 ))
```


Copyright 2015 FUJITSU LIMITED

分析（３）

FUJITSU

■コードの分析

- ■最適化情報では並列化を確認
- ■並列化次元は、イメージコードの最外次元"l"で並列化

```
37                    call start_collection("tune1A")
38      pp    v       cface%dudx = 0.0
39      pp    v       cface%dTdx = 0.0 ; cface%u    = 0.0
40      pp    v       cface%ar                ; cface%nv   = 0.0
41      pp    v       cface%r      = 0.0 ; cface%flux = 0.0
42                    call sto collection("tune1A")
```

```
pp   do l=1,3
       do k=-1,blk%kn
         do j=-1,blk%kn
           do i=-1,blk%kn
             cface%nv(i,j,k,l) =0.0
             ...
```

1から3スレッドに処理が集中

Copyright 2015 FUJITSU LIMITED

対処（１）

FUJITSU

■翻訳オプション -Kprallel_iteration による対処

```
% fccpx -Kparallel_iteration=16 ...
```

[-Kparallel_iteration=N]
翻訳時にN回転未満のループは自動並列化の対象外とし、内側のループで並列化を行う。

```
pp   do l=1,3
       do k=-1,blk%kn
         do j=-1,blk%kn
           do i=-1,blk%kn
             ...
```

```
       do l=1,3
pp       do k=-1,blk%kn
           do j=-1,blk%kn
             do i=-1,blk%kn
               ...
```

■イメージコードでは、内側次元"k"で並列化

■originalコードとほぼ同等の実行時間、1.66 → **0.50**(sec)

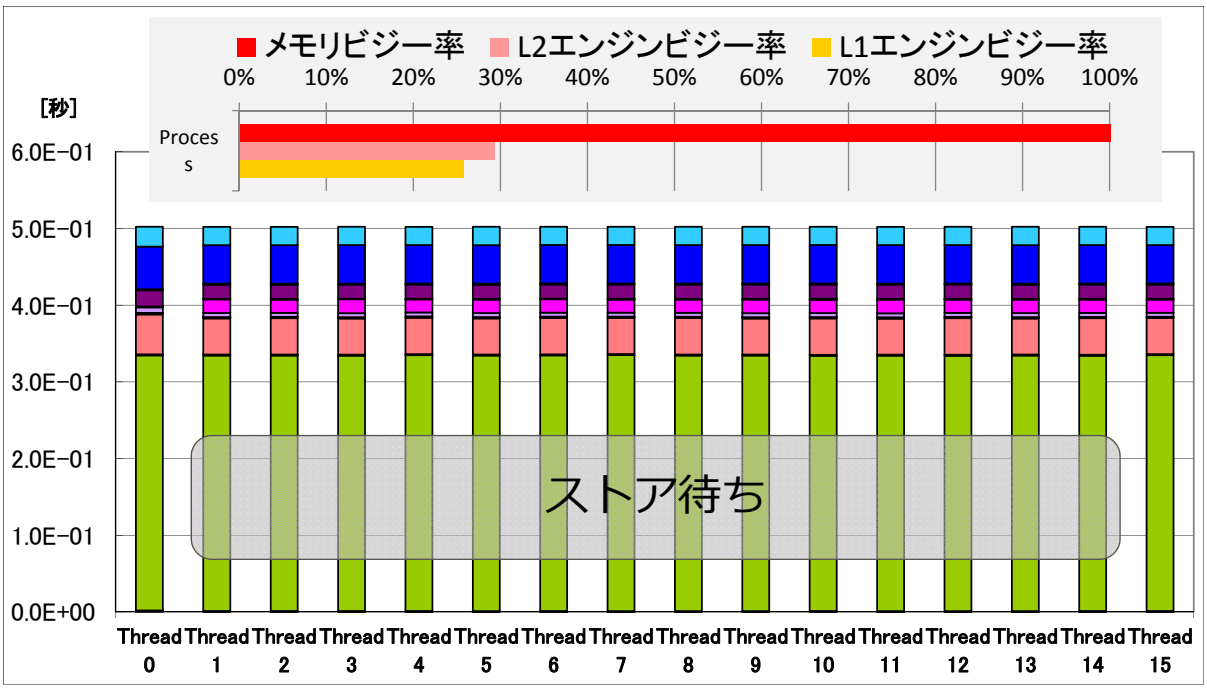
Copyright 2015 FUJITSU LIMITED

対処（２）

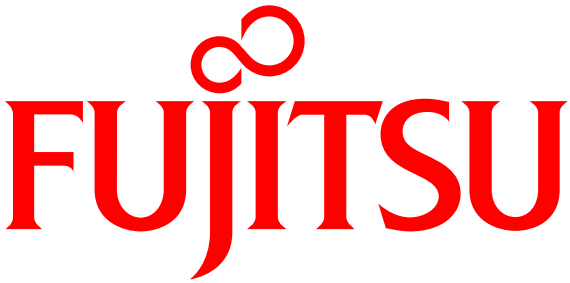
FUJITSU

■-Kparallel_iterationによる性能の変化

回転数が少なくxfillの効果が弱い、ループの一重化などが必要



Copyright 2015 FUJITSU LIMITED



shaping tomorrow with you

SS研
ポストペタアプリ性能WG資料

FUJITSU
shaping tomorrow with you

3.4 電解磁場解析「ADVENTURE_Magnetic」DAXPYのSIMD化について

2015年7月1日
富士通株式会社
千葉　修一

Copyright 2015 FUJITSU LIMITED

サマリー（１）

FUJITSU

- 課題

  DAXPYコードなどSIMD化されない理由が不明確

- 対象コード
  - 電解磁場解析「ADVENTURE_Magnetic」
  - DAXPY：aij_solver_cg.hの367行目のループ
- メッセージ

  コンパイラの世代によってメッセージが異なる

  ① jwd6101s-i "aij_solver_cg.h", line 367: ループ内にSIMD化の制約となる文が存在します。

  ② jwd6202s-i "aij_solver_cg.h", line 369: ループ中で変数'aq'を定義する順序が逐次実行と異なるため、このループはSIMD化できません。

Copyright 2015 FUJITSU LIMITED

サマリー（２）

FUJITSU

■回答

■SIMD化されない原因

- ポインタアクセスにより、依存関係が無いことが判断できないため
- コンパイラに対する指示によりSIMD化が可能

■メッセージが異なる原因

- メッセージの改善により以下の版から出力内容が変更
- 「京」16版　　(2014年3月)
- FX10　V1.0L30（2013年9月)

■コンパイラへのフィードバック

- 判りやすいメッセージへの改善（短期課題）
- ポインタのSIMDの強化（長期課題）

Copyright 2015 FUJITSU LIMITED

分析（１）

FUJITSU

■対象コード

```
#define MTYPE double
#define MPLUS( x, a, b )          ( (x) = (a) + (b) )
#define MMULTI( x, a, b )         ( (x) = (a) * (b) )
```

```
MTYPE  *r ;
MTYPE  *aq ;
 r = (MTYPE *)Malloc( sizeof(MTYPE), nf ) ;
aq = (MTYPE *)Malloc( sizeof(MTYPE), nf ) ;
```

```
for( i=0 ; i<nf ; i++ ) {
  MTYPE  dd ;
  MMULTI( dd, alpha, aq[i] ) ;
  MPLUS( r[i], r[i], dd ) ;
}
```

ループ内のデータに
依存あり

Copyright 2015 FUJITSU LIMITED

分析（２）

FUJITSU

■データ依存の分析

■展開コード

```
for( i=0 ; i<nf ; i++ ) {
  double  dd ;
  dd = alpha * aq[i];
  r[i] = r[i] + dd;
}
```

■コンパイラの判断

- 変数aq と 変数r はポインターであり依存が不明
- SIMD化するとデータアクセス順序が変化すると判断

Copyright 2015 FUJITSU LIMITED

分析（３）

FUJITSU

■SIMD化によるデータアクセス順序

```
MTYPE aq = &temp[i];
MTYPE r  = &temp[i+1];
```

```
LOOP:
 load  aq[i] -> s;
 store  r[i] <- s;
goto LOOP;
```

SIMD化

```
LOOP:
 vload  aq[i:i+1] -> v;
 vstore  r[i:i+1] <- v;
goto LOOP;
```

※ ストア／ロードのみの簡易コードで評価

データアクセス順序

```
-Iteration 1,2-
 load  temp[i]   -> s;
 store temp[i+1] <- s;
 load  temp[i+1] -> s;
 store temp[i+2] <- s;
```

temp[i+1]に対する
アクセス順序が変化

```
-Iteration 1-
 load  temp[i]   -> s;
 load  temp[i+1] -> s;
 store temp[i]   <- s;
 store temp[i+1] <- s;
```


Copyright 2015 FUJITSU LIMITED

対処（１）

FUJITSU

■restrict 修飾子による対処

```
#define MTYPE double
#define MPLUS( x, a, b )          ( (x) = (a) + (b) )
#define MMULTI( x, a, b )         ( (x) = (a) * (b) )
```

```
MTYPE  * restrict r ;
MTYPE  * restrict aq ;
 r = (MTYPE *)Malloc( sizeof(MTYPE), nf ) ;
aq = (MTYPE *)Malloc( sizeof(MTYPE), nf ) ;
```

```
for( i=0 ; i<nf ; i++ ) {
  MTYPE  dd ;
  MMULTI( dd, alpha, aq[i] ) ;
  MPLUS( r[i], r[i], dd ) ;
}
```

[restrict 修飾子]
指定した変数の
ポインタのアクセスが
重ならないことを指示

Copyright 2015 FUJITSU LIMITED

対処（２）

FUJITSU

■最適化指示行 NORECURRENCEによる対処

```
% fccpx -Kocl ...
```

```
#pragma loop norecurrence
for( i=0 ; i<nf ; i++ ) {
  ...
```

[norecurrence 指示子]
ループ内の
ポインタのアクセスが
重ならないことを指示

■翻訳オプション -Krestp による対処

```
% fccpx -Krestp=all ...
```

[-Krestp=all]
ファイル内のポインタのアクセスが重ならないことを指示

[-Krestp=arg]
関数の引数のポインタのアクセスが重ならないことを指示

Copyright 2015 FUJITSU LIMITED

コンパイラへのフィードバック

FUJITSU

■メッセージの改善

メッセージの改善を検討する。

メッセージ例)
jwd*NNNN*s-i "aij_solver_cg.h", line 369: ループ中で変数'aq'と変数'r'の依存が不明確なため、このループはSIMD化できません。

■コンパイラの解析能力の強化

C/C++コンパイラにおいて、ポインタ変数が指す領域の重なりを判断できるように改善します。

→ restrict修飾子なしでもSIMD化を適用

Copyright 2015 FUJITSU LIMITED

FUJITSU

shaping tomorrow with you

SS研
ポストペタアプリ性能WG資料

FUJITSU
shaping tomorrow with you

3.5 圧縮流体解析「UPACS-L」
レジスタの不足、大きなループボディに対する最適化の適用

2015年12月3日
富士通株式会社
千葉 修一

Copyright 2015 FUJITSU LIMITED

調査事項

FUJITSU

- 課題
  - レジスタの不足によりSIMD化/SWPLが適応されない
  - 現象の説明と回避方法の明確化
- 対象コード
  - 圧縮流体解析「UPACS-L」
  - cellFaceVariables（粘性項）
- コードの特徴
  - ステンシル系のアクセス、大きなループボディ
  - 配列記述を手で展開
  - 一時配列のスカラー化

Copyright 2015 FUJITSU LIMITED

対象コード

FUJITSU

■粘性項コード

blk_rhsViscous.f90：199行目～466行目

```
199   do k = isrt(3),iend(3)
200     do j = isrt(2),iend(2)
201       do i = isrt(1),iend(1)
202
203         i1 = i  + idelta(1) ; j1 = j  + idelta(2) ; k1 = k  + idelta(3)
204
205         !     xi derivative
206
207         !u_0(1:3) = blk%q(i ,j ,k ,2:4) / blk%q(i ,j ,k ,1)
208         !u_1(1:3) = blk%q(i1,j1,k1,2:4) / blk%q(i1,j1,k1,1)
209         u0_1 = blk%q(2,i ,j ,k ) / blk%q(1,i ,j ,k )
210         u0_2 = blk%q(3,i ,j ,k ) / blk%q(1,i ,j ,k )
211         u0_3 = blk%q(4,i ,j ,k ) / blk%q(1,i ,j ,k )
212         u1_1 = blk%q(2,i1,j1,k1) / blk%q(1,i1,j1,k1)
213         u1_2 = blk%q(3,i1,j1,k1) / blk%q(1,i1,j1,k1)
214         u1_3 = blk%q(4,i1,j1,k1) / blk%q(1,i1,j1,k1)
～
```

「宇宙航空研究開発機構) 高木様 圧縮性流体解析プログラム UPACS-Lコード」より

Copyright 2015 FUJITSU LIMITED

調査結果

FUJITSU

■回答

■レジスタ不足の原因となる二つの要因

1. 配列のアドレス計算を実施するために多くのレジスタを利用
2. 最内ループに対する「Invariant Code Motion」最適化の適用
   a. ループ内不変となる計算式をループ外へ移動し、ループ中の演算を削減
   b. 最内ループが動作中は、移動した演算が利用するレジスタを占有
   c. 最内ループで利用できるレジスタが少なくなる

■対処

- ループ分割することで最適化を促進することが可能
- ただし、データのローカリティ（局所性）を低下させない考慮が必要

Copyright 2015 FUJITSU LIMITED

詳細情報 (1)

FUJITSU

FX100におけるレジスタ

- HPC-ACEにより整数レジスタが拡張
- 整数レジスタは、グローバルレジスタを32→64
- 浮動小数点レジスタは、SIMD時に128本

	FX1	京/FX10/FX100
浮動小数点レジスタ	32本	256本
整数レジスタ	160本	192本

Copyright 2015 FUJITSU LIMITED

詳細情報 (2)

FUJITSU

アドレス計算

- 配列次元が増えるほど多くのレジスタが必要

```
do i = isrt(1),iend(1)
  ...
  blk%q(2,i ,j ,k ) / blk%q(1,i ,j ,k )
```

アドレス計算式

```
blk%q(2,i ,j ,k )アドレス
 = blkアドレス
 + qのオフセット
 + ( 2
     + 1次サイズ * i
     + 2次サイズ * j
     + 3次サイズ * k
   ) * 配列の型サイズ
```

命令イメージ

```
load blk -> r1
add r1 + qのオフセット -> r2
mult 1次サイズ * i -> r3
mult 2次サイズ * j -> r4
mult 3次サイズ * k -> r5
add  2  + r3 -> r6
add  r4 + r5 -> r7
add  r6 + r7 -> r8
mult r8 * 配列の型サイズ -> r9
add  r2 + r9 -> アドレス
```


Copyright 2015 FUJITSU LIMITED

詳細情報 (3)

FUJITSU

■Invariant Code Motion

- ■ループの制御変数に依存しない演算をループ外へ移動
- ■演算は削減、ただし占有できるレジスタも減少

```
do i = isrt(1),iend(1)
  i1 = i  + idelta(1)
  j1 = j  + idelta(2)
  k1 = k  + idelta(3)
```

命令イメージ（最適化前）

```
LOOP_i:
  load idelta(1) -> r1
  load idelta(2) -> r2
  load idelta(3) -> r3
  add  i + r1 -> i1
  add  j + r2 -> k1
  add  k + r3 -> k1
  ...
  ++i
  条件中 LOOP_Iへ
```

命令イメージ（最適化後）

```
load idelta(1) -> r1
load idelta(2) -> r2
load idelta(3) -> r3
add  j + r2 -> k1
add  k + r3 -> k1
LOOP_i:
  add  i + r1 -> i1
  ...
  ++i
  条件中 LOOP_Iへ
```


Copyright 2015 FUJITSU LIMITED

詳細情報 (4)

FUJITSU

■アドレス計算に対する最適化

- ■Strength reduction

命令イメージ(最適化前)

```
LOOP_i:
  load blk -> r1
  add r1 + qのオフセット -> r2
  mult 1次サイズ * i -> r3
  mult 2次サイズ * j -> r4
  mult 3次サイズ * k -> r5
  add  2  + r3 -> r6
  add  r4 + r5 -> r7
  add  r6 + r7 -> r8
  mult r8 * 配列の型サイズ -> r9
  add  r2 + r9 -> アドレス
  ...
  ++i
  条件中 LOOP_Iへ
```

命令イメージ（最適化後）

```
LOOP_i:
  load blk -> r1
  add r1 + qのオフセット -> r2
  mult 1次サイズ * i -> r3
  mult 2次サイズ * j -> r4
  mult 3次サイズ * k -> r5
  add  2  + r3 -> r6
  add  r4 + r5 -> r7
  add  r6 + r7 -> r8
~~mult r8 * 配列の型サイズ -> r9~~
  add  r2 + r8 -> アドレス
  ...
  i+=配列の型サイズ
  条件中 LOOP_Iへ
```


Copyright 2015 FUJITSU LIMITED

詳細情報 (4)

FUJITSU

■アドレス計算に対する最適化

■Invariant Code Motion

命令イメージ(最適化前)

```
LOOP_i:
  load blk -> r1
  add r1 + qのオフセット -> r2
  mult 1次サイズ * i -> r3
  mult 2次サイズ * j -> r4
  mult 3次サイズ * k -> r5
  add  2  + r3 -> r6
  add  r4 + r5 -> r7
  add  r6 + r7 -> r8
  mult r8 * 配列の型サイズ -> r9
  add  r2 + r9 -> アドレス
  ...
  ++i
  条件中 LOOP_Iへ
```

命令イメージ（最適化後）

```
  load blk -> r1
  add r1 + qのオフセット -> r2
  mult 2次サイズ * j -> r4
  mult 3次サイズ * k -> r5
  add  r4 + r5 -> r7
  add  2 + r7 -> r6
LOOP_i:
  mult 1次サイズ * i -> r3
  add  r6 + r3 -> r8
  mult r8 * 配列の型サイズ -> r9
  add  r8 + r9 -> アドレス
  ...
  i+=配列の型サイズ
  条件中 LOOP_Iへ
```

※ 移動した演算は共通で利用可能

Copyright 2015 FUJITSU LIMITED

詳細情報 (5)

FUJITSU

■最適化とレジスタの関係

■ソフトウェアパイプラインニングはさらにレジスタが必要

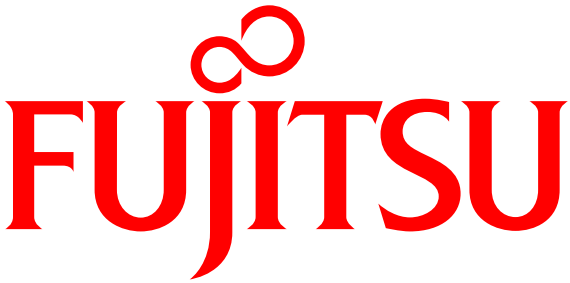
■演算とレジスタのバランスが重要

■コンパイラへのフィードバック

■演算／レジスタの見積もり精度を向上

- 大局的なレジスタ見積もりを検討
- オプション、指示行などによる選択を検討
- アドレッシング方式の選択を検討

Copyright 2015 FUJITSU LIMITED



shaping tomorrow with you