3. 最適化レポート

Revision 1.0

エクセルソフト株式会社
黒澤 一平

ベクトル化の結果表示

Linux*
-vec_report*n*

Windows*
/Qvec_report*n*

•ベクトル化レポートの診断レベルを *n* で設定する

n=*0*: 診断情報を表示しない
n=1: (デフォルト)ベクトル化されたループのみを表示する
n=*2*: ベクトル化されなかったループとその理由も表示する
n=3: 依存情報も表示する
n=4: ベクトル化されなかったループのみを表示する
n=5: ベクトル化されなかったループのみを表示し、
依存情報も表示する

XLSOFT

ベクトル化レポート

ループがベクトル化されない理由

- Existence of vector dependence
  "ベクトルの依存関係が存在する"
- Nonunit stride used
  "ユニットストライドではないアクセス"
- Mixed Data Types
  "データ型が混合している"
- Condition too Complex
  "条件が複雑すぎる"
- Condition may protect exception
  "分岐条件の変更が例外を招く"
- Low trip count
  "ループ回数が少なすぎる"
- Subscript too complex
  "添字が複雑すぎる"
- Unsupported Loop Structure
  "サポートされないループ構造"
- Contains unvectorizable statement at line XX
  "行 XX にベクトル化できない文が含まれている"
- Not Inner Loop
  "内部ループでない"
- Vectorization possible but seems inefficient
  "ベクトル化しても非効率"
- Operator unsuited for vectorization
  "演算子がベクトル化に適していない"

"ユニットストライドではないアクセス"

```
for (i=0;i<=MAX;i++)
        for (j=0;j<=MAX;j++) {
                c[i][j]+=1;    // ユニットストライド
                c[j][i]+=1;    // 非ユニットストライド
                A[j*j]+=1;     // 非ユニットストライド
                A[B[j]]+=1;    // 非ユニットストライド
                if (A[MAX-j]=1) last1=j;  // 非ユニットストライド
        }
```

- 非ユニットストライドはベクターデータの読み込み時間が演算時間を上回り、データ読み込み待ちが多発する

- コンパイラーは非効率な演算をベクトル化しない

C/C++、Fortran 混在言語プログラミングのTips
C から Fortran を呼び出す場合

- 呼び出す Fortran 関数を大文字にする
- C は行インデックスだが、Fortran は列インデックスであるため配列をインデックス順にスワップする

C[I][J] = C[I][J] + A[I][K] * B[K][J]

の場合、

C(J,I) = C(J,I) + A(K,I) * B(J,K) を使用する

C(I,J) = C(I,J) + A(I,K) * B(K,J) は使用しない

"分岐条件の変更が例外を招く"

SIMD 演算では条件比較はビット・マスクに変更されるが
例外が起きる場合にはベクトル化されない

```
int A[MAX-50];
for (i = 0; i < MAX; i++) {
    if (i < MAX-50) A[i] = 0;
    ...
}
```

"ループ回数が少なすぎる"

ベクトル化はオーバーヘッドが発生する ため
反復回数が少ないループはベクトル化されない

反復回数が分かっている場合には、コンパイラーに
大きさを知らせ、最適化を補助して対応する

```
#pragma loop count(5)
    for (i = 0; i < MAX1; ++i) {
         a1[i] = b1[i] * c1[i] + d1[i];
    }

#pragma loop count(1000)
    for (i = 0; i < MAX2; ++i) {
        a2[i] = b2[i] * c2[i] + d2[i];
    }
```

XLSOFT

"ベクトル化しても非効率"

ループが大きいと、多くのレジスターが必要になる
以下の場合、32bitアプリケーションはレジスターが不足する

例えば、ループの分割をして対応する
#pragma distribute point – ループ分割機能の使用例

```
for (i = 0; i < N; i++) {
  A[i] = B[i] * max(C[i], D[i]) + F[i] - E[i];
  #pragma distribute point
  G[i] = H[i] * sin(I[i]) + J[i] / K[i];
}
```

ただし、他の問題によってもこの警告が出力されることがある

ベクトル化を制御する宣言子

後に続くループのベクトル化を制御する

#pragma novector – ループのベクトル化が有効な場合でもループをベクトル化しない

#pragma vector always – メリットと例外検出に関する判別を無視し、常にループをベクトル化する

```
#pragma vector always
   for (i = 0; i < MAX; ++i) {
       a[i] = b[i] * c[i] + d[i];
   }
```

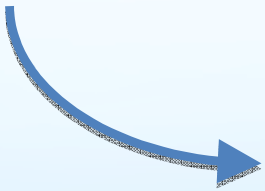

XLSOFT

"サポートされないループ構造"

コンパイラーがコンパイル時にループの反復数をカウントできない場合や、コンパイラーがループ反復数をランタイム時に判定できない場合に最適化を断念してしまう

```
struct _xx { int data; int bound; };

doit1(int *a, struct _xx *x) {
for (int i = 0; i < x->bound; i++)
  a[i] = 0;
}
```



```
struct _xx { int data; int bound; };

doit1(int *a, struct _xx *x) {
int MAX = x->bound;
for (int i = 0; i < MAX; i++)
  a[i] = 0;
}
```

一旦、単純な変数に置き換え、コンパイラーが判断できるようにすることで、回避できることが多い

"混在するデータ型"

```
int howmany_close(double *x, double *y) {
        long long  withinborder=0;
        double dist;
        for(int i=0;i<MAX;i++) {
                dist=sqrtf(x[i]*x[i] + y[i]*y[i]);
                if (dist<5) withinborder++;
        }
}
```

データ型の混合は可能だが、複雑になると、コンパイラーのバージョンや、ターゲットプロセッサーの指定によってはベクトル化できない場合がある。
対応は、できる限りキャストせず、同じ型で宣言するようにする。

"ベクトルの依存関係が存在する"

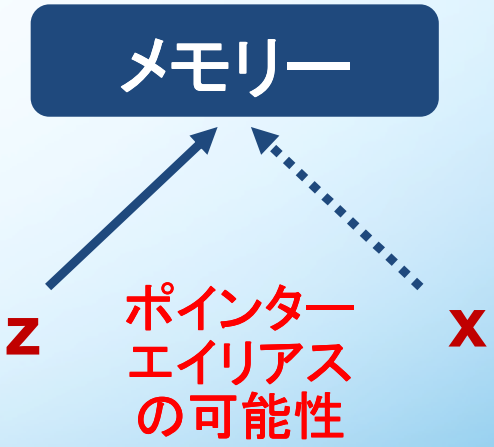
- ループの反復にベクトル化できない依存関係がある場合

または

- ２ つの配列ポインターが重複している可能性のある場合

(コンパイラーは完全に安全な条件でないとベクトル化しない)

```
void scale(float *z, float *x) {
        for (i = 0; i < 100; i++)
                z[i] = A * x[i];
}
```



次ページ解決方法

エイリアス問題をコンパイラーに解決させるための方法

- IPO の使用
  IPO による定数伝播やスタティック解析はポインターを明確化できる

-ipo (Linux)　　/Qipo (Windows)

- C99 の restrict オプションの使用
  ポインターの明確化を有効にする
  -restrict (Linux)　　/Qrestrict (Windows)

```
void mult(int a[][NUM], int b[restrict][NUM]);

void foo(int *x, int *y, int * restrict z) {
    int i;
    for (i = 0; i < 100; i++) z[i] = A * x[i] + y[i];
}
```

エイリアス問題の対応方法つづき

ivdep 宣言子により、ベクトルの依存関係をコンパイラーに無視させる

```
#pragma ivdep
    for (i = 0; i < 100; i++)
            z[i] = A * x[i] + y[i];
```

```
!DEC$ IVDEP
  DO I=1, N
    A(INDARR(I)) = A(INDARR(I)) + B(I)
  END DO
```

エイリアス問題の確認を無視させるオプション

-fno-alias (Linux)　　/Oa (Windows)

コンパイラーは依存関係の確認をせずベクトル化する・最適化していった場合の、最大の性能を確認するために使用する。

ただし、エイリアスが存在する場合には即座に例外が生じてしまうため、実運用には使用しないほうがよい。

最適化レポート

指定された最適化機構のレポートを生成します。

- Linux:

  -opt_report -opt_report_phase=*〈phase〉*

- Windows:

  /Qopt_report /Qopt_report_phase:*〈phase〉*

〈phase〉 =

ipo	プロシージャー間の最適化
hlo	高レベル最適化
hpo	ハイパフォーマンス最適化
pgo	プロファイルに基づく最適化
all	すべての最適化
ilo	中間言語スカラー最適化

使用例:

```
icc -O3 -xHost -opt-report -opt-report-phase=hlo sample.c
```

※ -O3 等の最適化オプションがないと、レポートが生成されない

いろいろな最適化レポート出力

HLO レポートの
アンロール情報

Block, Unroll, Jam Report:
(loop line numbers, unroll factors and type of transformation)

<pardat.f90;157:157;hlo_unroll;parallel_data_mp_exchange_;0>
Loop at line 157 unrolled with remainder by 2
Loop at line 157 completely unrolled by 1

<pardat.f90;187:187;hlo_unroll;parallel_data_mp_exchange_;0>
Loop at line 187 unrolled with remainder by 2
Loop at line 187 completely unrolled by 1

<pardat.f90;217:217;hlo_unroll;parallel_data_mp_exchange_;0>
Loop at line 217 unrolled with remainder by 2
Loop at line 217 completely unrolled by 1

<red_black.f90;11:138;IPO INLINING;MAIN_
INLINING REPORT: (MAIN__) [1/2=50.0%]

-> for_write_seq_lis(EXTERN)
-> for_write_seq_lis_xmit(EXTERN)
-> for_write_seq_lis(EXTERN)
-> for_write_seq_lis_xmit(EXTERN)
-> output_mp_out_(EXTERN)
-> mpi_wtime_(EXTERN)
-> poisson_red_black_(2) (isz = 2278) (sz = 2295 (983+1312))
-> mpi_wtime_(EXTERN)
-> for_check_mult_overflow64(EXTERN)
-> for_alloc_allocatable(EXTERN)
-> for_check_mult_overflow64(EXTERN)

IPO レポートの
インライン情報

最適化に関する注意事項

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Optimization Notice

最適化に関する注意事項

インテル コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル ストリーミング SIMD 拡張命令 2 (インテル SSE2)、インテル ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。

改訂 #20110804

XLSOFT