インテル® Fortran コンパイラーによる効果的な並列処理の最適化

Martyn Corden
インテル コーポレーション
2016 年 3 月

内容

SIMD 並列処理

- 自動ベクトル化
- OpenMP* 4.5 による明示的な SIMD プログラミング

マルチコア並列処理

- 自動並列化
- OpenMP*

簡潔にするため、ほとんどの例では Linux*/ OS X* 用のコンパイラー・オプションを使用しています。Windows* 用のオプションはインテル® コンパイラー・ユーザー・リファレンス・ガイドを参照してください。

重要だがコンパイラーの一部ではなく今回は取り上げないもの

- インテル® MKL
- MPI

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

SIMD: Single Instruction Multiple Data

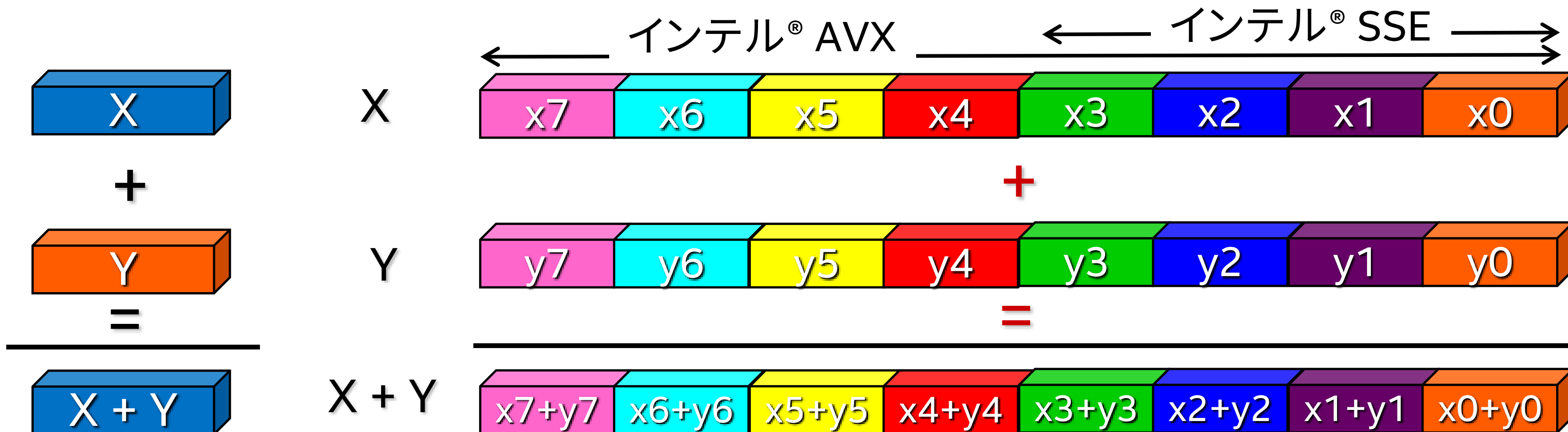
スカラーモード

for (i=0; i<n; i++) z[i] = x[i] + y[i];

- 1 つの命令で 1 つの結果
- 例: vaddss (vaddsd)

ベクトルモード

- インテル® AVX (または インテル® SSE) 命令
- 1 つの命令で複数の結果
- 例: vaddps (vaddpd)



最適化に関する注意事項

© 2016 Intel Corporation. 無断での引用、転載を禁じます。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

SIMD 並列処理

- 自動ベクトル化
- OpenMP* 4.5 による明示的な SIMD プログラミング

マルチコア並列処理

- 自動並列化
- OpenMP*

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

自動ベクトル化に影響を与える要因

ループ伝播の依存性

```
DO I = 1, N
   A(I+IOFFSET) = A(I) + B(I)
ENDDO
```

不明な反復回数 / 複数の exit

```
DO I = 1, N
   IF(A(I) < 0.)EXIT
   A(I) = SQRT(A(I))
ENDDO
```

関数呼び出しやサブルーチン呼び出し

```
DO I = 1, N
   SUMX = SUMX + FUNC(X(I), X0)
ENDDO
```

間接メモリーアクセス / 非ユニットストライド

```
 DO I = 1, N
      A(I)= B(I)*C(INDEX(I)) )
 ENDDO
```

ポインター・エイリアシング

```
REAL,POINTER,DIMENSON(:) :: A,B,C
   DO I = 1, N
      A(I) = B(I) + 2.*C(I)
   ENDDO
```

OK: IF 文、数学関数、スカラーの代入

```
DO I=1,N
  S = B(I)**2 - 4.*A(I)*C(I)
  IF ( S .GT. 0 ) X(I) =
       (-B(I)+SQRT(S))/(2.*A(I))
ENDDO
```

など ……

最適化に関する注意事項

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ベクトル化可能なコードを記述するためのガイドライン

単純な DO ループまたは for ループを使用する

分かりやすいコードを記述し、次の表記は (できるだけ) 避ける

- 関数呼び出しやサブルーチン呼び出し
- マスク付きの代入として処理できない分岐

ループ反復間の依存性は避ける

- 少なくともリードアフターライト (RAW) 依存は避ける

ポインターや結合の代わりに配列を使用する

- 多くの場合、コンパイラーはポインターを含むコードを安全にベクトル化できるかどうか判断できない
- カウンターをインクリメントして配列アドレスに使用する代わりに、ループ・インデックスを配列インデックスで直接使用する

効率良いメモリーアクセスを使用する

- 内部ループとユニットストライドを使用する
- 間接アドレス指定は最小限に抑える
- できるだけ一貫した方法でデータをアライメントする　　例: -align array64byte

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

その他の一般的なアドバイス

ソースの手動アンロール (古いコードでは一般的) を回避する

- 単純な DO ループまたは for ループとして記述する
- コンパイラーが最適化やアライメントを行いやすいようにする
- プラットフォームに依存しないようにする
- より分かりやすく記述する

ループのインダクション変数をローカルにする (ループの制限を含む)

- コンパイラーはエイリアスできないことを知っている

Fortran ポインターと形状引き継ぎ配列の引数に注意する

- コンパイラーはユニットストライドであると仮定できない
  - 必要に応じて **CONTIGUOUS** を宣言する
- 可能であればポインターではなく割付け配列を使用する

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

形状引き継ぎ配列の仮引数を ...

```
module mod2
contains
  subroutine sub2(a,b,c)
    real, dimension(:) :: a,b,c

    a = b + abs(c)

  end subroutine sub2
end module mod2
```

ifort -c -qopt-report=4 sub2.f90

```
…
ループの開始 sub2.f90(6,5)
  リマーク #15335: ループはベクトル化されませんでした: ベクトル化は可能ですが非効率です。...
  リマーク #15329: ベクトル化のサポート: 分散 (scatter) がエミュレートされました (変数 a: 非定数値によるストライド)
  リマーク #15328: ベクトル化のサポート: 集約 (gather) がエミュレートされました (変数 b: 非定数値によるストライド)
  リマーク #15328: ベクトル化のサポート: 集約 (gather) がエミュレートされました (変数 c: 非定数値によるストライド)
```

```
module mod1
contains
  subroutine sub1(a,b,c)
    use mod2
    real, dimension(:) :: a,b,c

      call sub2(a,b,c)

  end subroutine sub1
end module mod1
```

ifort -c -qopt-report=4 sub2.f90

(メッセージなし)

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

... 部分配列にする

コンパイラーは保守的で、引数が非ユニットストライドであると仮定する

- 例: CALL SUB2 (A(1:1000), B(1:2000:2), C(1:4000:4))
  - コンパイラーは SIMD (ベクトル) ロードを生成しない (→ 個々のロードのギャザー)
  - この例ではベクトル化を行うメリットはない
    - より多くのワークを含むループもベクトル化できるが遅くなる

解決方法: 配列がユニットストライドの場合、コンパイラーに知らせる

- REAL, DIMENSION(:), **CONTIGUOUS** :: A,B,C (Fortran 2008 の新機能)

```
リマーク #15300: ループがベクトル化されました。
リマーク #15448: マスクなしアライン・ユニット・ストライド・ロード: 1
...
```

または、大きさ引き継ぎ配列や可変長配列を使用する

```
Subroutine sub2(a, b, c, n)
    real, dimension(n) :: a, b, c
```

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

呼び出し元の CONTIGUOUS 属性の影響

sub1 の sub2 へのインターフェイスが変更された

- 多くのベクトル化メッセージが表示されるようになった

```
ループの開始 sub1.f90(7,10)
<マルチバージョン v1>
  リマーク #15300: ループがベクトル化されました。
ループの終了

ループの開始 sub1.f90(7,10)
<マルチバージョン v2>
  リマーク #15335: ループはベクトル化されませんでした: ベクトル化は可能ですが非効率です。...
ループの終了
```

- ループが含まれていないソースコードでも有効
- コンパイラーは sub2 に必要な引数の連続コピーを行う
  - 問題をアップストリームに戻しただけ
  - **CONTIGUOUS** がコールスタックをバックアップするように配列を宣言する
  - sub1 のループがなくなる

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

変更後のソースコード

```
module mod2
contains
  subroutine sub2(a,b,c)
    real, dimension(:), contiguous :: a,b,c

    a = b + abs(c)

  end subroutine sub2
end module mod2
```

```
ifort -c -qopt-report=4 sub2.f90
```

```
…
ループの開始 sub2.f90(6,5)
  リマーク #15300: ループがベクトル化されました。
  リマーク #15448: マスクなしアライン・ユニット・ストライド・ロード: 1
  リマーク #15449: マスクなしアライン・ユニット・ストライド・ストア: 1
  リマーク #15450: マスクなし非アライン・ユニット・ストライド・ロード: 1
```

```
module mod1
contains
  subroutine sub1(a,b,c)
    use mod2
    real, dimension(:), contiguous :: a,b,c

      call sub2(a,b,c)

  end subroutine sub1
end module mod1
```

```
ifort -c -qopt-report=4 sub2.f90
```

(メッセージなし)

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

パフォーマンスへの影響

(配列サイズ 1,000、2,000,000 回実行)

Sub1 contiguous	Sub2 contiguous	スピードアップ
×	×	1.0
×	○	0.85
○	×	1.0
○	○	3.8

性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。
システム構成: 第 4 世代インテル® Core™ i7-4790 プロセッサー 3.60GHz、Red Hat* Enterprise Linux* 7.0、インテル® Fortran コンパイラー 16.0.2。

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

SIMD 並列処理

- 自動ベクトル化
- OpenMP* 4.5 による明示的な SIMD プログラミング

マルチコア並列処理

- 自動並列化
- OpenMP*

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* による明示的なベクトル・プログラミング

ベクトル化は非常に重要 ➔ 明示的なベクトル・プログラミングを検討すべき

OpenMP* によるスレッド化 (明示的な並列プログラミング) をモデルにしている

- コンパイラーが自動ベクトル化できない複雑なループをベクトル化
  - 例: 外部ループ (以前の Web セミナーを参照):
  - https://software.intel.com/videos/new-vectorization-features-of-the-intel-compiler
- コンパイラーにとってディレクティブはヒントではなく命令
  - プログラマーが正当性を保証する必要がある (OpenMP* によるスレッド化と同様)
    - 例: PRIVATE 節、FIRSTPRIVATE 節、REDUCTION 節
- OpenMP* 4.0 で採用 ⇒ 移植可能
  - -qopenmp または -qopenmp-simd で有効にする

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

!$OMP SIMD ディレクティブ

-qopenmp (/Qopenmp) または -qopenmp-simd (/Qopenmp-simd) で有効にする

```
subroutine add(A, N, X)
   integer N, X
   real    A(N)
 !$ OMP SIMD
   DO I=X+1, N
       A(I) = A(I) + A(I-X)
   ENDDO
end
```

ifort -c -qopt-report-file=stderr add.f90

...

ループはベクトル化されませんでした: ベクトル依存関係が存在しています。

ifort -c -qopt-report-file=stderr **-qopenmp-simd** add.f90

...

SIMD ループがベクトル化されました。

ループをベクトル化しても安全であると**知っている**ときに使用する

インテル® コンパイラーはベクトル化できる場合は (依存性や効率を無視して) ベクトル化する
http://www.isus.jp/products/psxe/requirements-for-vectorizable-loops/
ベクトル化を行うために必要なソースコードの変更を最小限に抑える

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。
intel

サブルーチン呼び出しを含むループ

関数呼び出しを行うと自動ベクトル化を妨げるループ伝播の依存性が発生する場合の解決方法:

- インライン展開
  - 小さなプロシージャーに最適
  - 同じソースコードで展開するか、-ipo を使用する (Windows* の場合は /Qipo)
- SIMD 対応関数またはサブルーチン
  - 大規模ルーチン、複雑なルーチン、インライン展開が困難な場合に適切
  - 通常の DO ループから呼び出す
  - ELEMENTAL キーワードを追加すると SIMD 対応プロシージャーを部分配列引数で呼び出すことができる
- !$OMP SIMD ディレクティブ (最後の手段)
  - プロシージャーに対するスカラー呼び出しを維持しつつループの剰余をベクトル化する

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

SIMD 対応サブルーチン

コンパイラーはベクトル化されたループから呼び出すことができるスカラー・サブルーチンの SIMD 対応 (ベクトル) バージョンを生成する:

```
subroutine test_linear(x, y)
!$omp declare simd (test_linear) linear(ref(x, y))
   real(8),intent(in)  :: x
   real(8),intent(out) :: y
   y = 1. + sin(x)**3
end subroutine test_linear
…
interface
…
do j = 1,n
    call test_linear(a(j), b(j))
enddo
```

Fortran では引数は参照渡しのため重要

リマーク #15301: 関数がベクトル化されました。

リマーク #15300: ループがベクトル化されました。

```
ifort -xavx -qopenmp-simd -qopt-report-file=stderr test_linear.f90
```

SIMD 対応ルーチンは明示的なインターフェイスにする (単純なケースでは !$omp simd は必要ない)

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

SIMD 対応サブルーチン - パフォーマンス

LINEAR(REF()) 節は非常に重要 (コンパイラー 16.0.1 の新機能)

- C では、コンパイラーはベクトルレジスターに連続する引数値を配置する
- しかし Fortran では引数は参照渡し
  - デフォルトでは、コンパイラーはベクトルレジスターに連続するアドレスを配置する
  - 4 つのアドレスのギャザーを行う (遅い)
  - LINEAR(REF(X)) はアドレスが連続していることをコンパイラーに知らせる;
一回のみ逆参照してベクトルレジスターに連続する値をコピーする必要がある
- 同じ方法を参照渡しの C 引数にも使用可能

要素 100 万の倍精度配列のスピードアップ	
DECLARE SIMD なし	1.0
DECLARE SIMD あり、LINEAR(REF) 節なし	0.9
DECLARE SIMD あり、LINEAR(REF) 節あり	3.6

性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。
システム構成: インテル® Xeon® プロセッサー E7-4850 v3 システム、2.20GHz、Red Hat* Enterprise Linux* 7.1、インテル® Fortran コンパイラー 16.0.1。

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

SIMD 並列処理

- 自動ベクトル化
- OpenMP* 4.5 による明示的な SIMD プログラミング

マルチコア並列処理

- 自動並列化
- OpenMP*

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

マルチコア並列処理 – スレッドセーフ

OpenMP* および自動並列化の両方に適用する

- Co-Array には適用しない

アプリケーション全体を -qopenmp (または -threads -auto) でコンパイルする

- OpenMP* ディレクティブを含まない「親なし」サブルーチンをインクルード
  - ローカル配列をスレッドセーフにする
    - -parallel のみでは十分ではない
    - 多くの場合スタックサイズを増やす必要がある
- メインルーチンをインクルード
  - Fortran RTL のリンクと初期化をスレッドセーフ・モードで行う

(コマンドラインに関係なく) ソースコードでルーチンをスレッドセーフにするには

- 関数を RECURSIVE として宣言する
- -save や SAVE キーワードを使用しない
- グローバル変数を避ける (または同期されていない場合は書き込まない)

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

自動並列化 (自動スレッド化)

自動ベクトル化のように動作する

- -parallel (/Qparallel) で有効にする
- OpenMP* ランタイム・ライブラリーを利用する
- 反復はそれぞれ完全に独立している必要がある
- スレッド化のオーバーヘッド >>> ベクトル化のオーバーヘッド (~10^4 サイクル)
  - 多くの計算ワークを含むループでのみメリットがある
  - 関数呼び出しはインライン展開する必要がある (必要な場合は ipo を使用する)

- DO CONCURRENT (Fortran 2008) が効果的
  - ループは並列実行しても安全であると示す
  - 呼び出されるサブルーチンや関数は PURE である (副作用がない) こと
  - 明示的なインターフェイスが必要
  - スレッドセーフであること (-auto または recursive; -parallel では十分ではない)
  - 関数呼び出しの実装はまだ完全ではない ☹ - dpd200378091 の修正待ち

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

DO CONCURRENT の例

```
ifort -c -parallel test.f90 -qopt-report-file=stderr
...
ループの開始 test.f90(15,3)
  リマーク #17104: ループは並列化されませんでした: 並列依存関係が存在しています。
  リマーク #15344: ループはベクトル化されませんでした: ベクトル依存関係がベクトル化を妨げています。
```

DO を DO CONCURRENT に置換:

```
ifort -c -parallel test.f90 -qopt-report-file=stderr
...
ループの開始 test.f90(16,3)
  リマーク #17109: ループが自動並列化されました。
  リマーク #15527: ループはベクトル化されませんでした: sub_ の関数呼び出しはベクトル化できません   [ test.f90(19,12) ]
ループの終了
```

```
pure recursive subroutine sub (y, z)
  real, intent(in)    :: y
  real, intent(out) :: z
  z = 1. + sin(y)**2
end subroutine sub
...

  do i=1, n
!  do concurrent (i=1:n)
     call sub(b(i), c(i))
  enddo
```

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

SIMD 並列処理

- 自動ベクトル化
- OpenMP* 4.5 による明示的な SIMD プログラミング

マルチコア並列処理

- 自動並列化
- OpenMP*

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® Fortran コンパイラーでの OpenMP* の使用

OpenMP* 標準について学ぶ

- www.openmp.org (英語) を参照

単純だがプログラマーが正当性を保証する必要がある

- PRIVATE、REDUCTION、その他の節

最適化されたシリアルコードで開始する

- ベクトル化された内部ループ
- -O3 -xcore-avx2 -ipo ...

スレッド化可能な最外ループをスレッド化する

同じループをスレッド化およびベクトル化できる

- ワークの量で保証
- 自動ベクトル化または明示的なベクトル化
- $omp parallel do simd ....

```
!$omp parallel do private(a) &
!$omp reduction(+:sum)

 do i=1, n
    a = real(i)
    a = 1. + a + a**2
    call sub(a,  b(i), c(i))
    sum = sum + a
 enddo
```

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

パフォーマンスの考慮事項

キャッシュラインのフォルス・シェアリングを回避する

- 各スレッドは無効にされた A(i,j) のコピーと考える
- A の添字を反転すると各スレッドのデータの局所性が向上する
  - メモリーアクセスを連続にして内部ループのベクトル化を許可する

```
!$OMP parallel do
 do i=1,nthreads
  do j=1,10000
   A(i,j) = A(i,j) + ..
```

スケジュール・オプション

- ループ反復間で作業が均等に分散されていない場合 DYNAMIC や GUIDED を検討する

OMP_PROC_BIND でスレッド/プロセッサーのアフィニティーを設定する

- spread はほとんどのアプリケーションに適していて、すべてのコアとソケットを利用する
- close はスレッドをできるだけ近くに配置して、コアあたりのスレッド数を最大にする
  - 共有キャッシュは最大限に利用されるが、一部のコアやソケットはアイドルになる
- デフォルトはアフィニティーなし - 通常は希望と異なる
- KMP_AFFINITY (インテル固有) でより細かく制御できる

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

高レベルのスレッド

並列領域でより多くのワーク -> より優れたスケーリング;

より簡単にスレッドセーフにできる:

```
!$omp parallel do  private(a,b,c,d,e,f,g,h, i1, i2 ...)   ! 保守が大変
Do  i = 1, n
  .....
!$omp parallel do  private(...)
Do  i = 1, n
  .....
```

変更後:

```
!$omp parallel
  call sub()
   ...
```

保守が容易

```
subroutine sub
Real  :: a,b,c,d,e,f,
Integer :: i1, i2
Real, dimension(ndim) :: g,h
!   -qopenmp を指定してコンパイルすると
!   ローカル宣言は自動的にスレッドセーフになる
...
!$omp do
  Do  i = 1, n
  ...
```

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

一般的な問題 – スタックサイズの不足

最も多くレポートされる OpenMP* 問題

- 症状: (スタック・オーバーフローによる) データ初期化中のセグメンテーション・フォルト
- -qopenmp (または -auto) はスレッドセーフにするためローカル配列をスタックに割り当てる

シェル (マスタースレッド) のスタック制限:

- プログラム全体、共有 + ローカルデータに (スレッド化に関係なく) 適用される
- OS のデフォルト値は小さい (~10MB)
- アドレス空間の制限; 必要な場合はメモリーが割り当てられる
  - 大きい値を設定しても安全
  - Bash: ulimit –s [値 (キロバイト)] または [unlimited] (1 回のみ増加可能)
  - C シェル: limit stacksize [値 (キロバイト)]
  - Windows*: /F:100000000 でリンク (値はバイト)

個々のスレッドのスタック割り当て (スレッドローカルなデータのみ):

- export **OMP_STACKSIZE**=[サイズ] デフォルトは 4m (4MB)
- 実際にはメモリーを割り当てるため大きな値にしないこと

最適化に関する注意事項

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* でスレッド化したアプリケーションのデバッグ

-O0 でデバッグする; SIMD ベクトル化とは異なりスレッド化は -O0 で無効にならない

-qopenmp でビルドし、シングルスレッドで実行する (OMP_NUM_THREADS=1)

- これで動作した場合、インテル® Inspector で競合状態その他を検出する

-qopenmp-stubs -auto でビルドする

- RTL 呼び出しは解決する; スレッド化されたコードは生成されない
- これで動作した場合、FIRSTPRIVATE、LASTPRIVATE が不足していないか確認する

-qopenmp-stubs でビルドする

- これで動作した場合、OpenMP* で変更されたメモリーモデルが影響している
  - スタックサイズが不足している可能性あり (前のスライドを参照)
  - 値が連続するプロシージャー呼び出しで保持されていない可能性あり

PRINT 文でデバッグする場合

- omp_get_thread_num() でスレッド番号を出力する前に OMP_LIB を使用する
- 内部 I/O バッファーはスレッドセーフ
- 異なるスレッドからの PRINT 文の順序は不定

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Co-Array について (Fortran 2008)

長所

- 共有メモリー並列処理と分散メモリー並列処理で同じコード
- スレッドセーフ
- MPI よりも習得が容易

短所

- クラスター環境をセットアップする必要がある
  - パスワードレス ssh、その他
- MPI よりも機能が少ない
- MPI よりも通信オーバーヘッドがかなり大きい
  - コンパイラー 17.0 で大幅に改良

オーバーヘッドを考慮すると多くのワークが必要

```
  real, dimension(n), codimension[*] :: b,c

  nimages = num_images()
  myid    = this_image()
  print *, this_image =', myid, ' /', nimages
! データを最初に初期化する
…
   do i = myid, n, nimages
     a = real(i)
     a = 1. + sin(a)**2
     call sub(a, b(i), c(i))
   enddo

   sync all ! 結果をほかでも利用可能にする

ifort -coarray test_ca.f90;  ./a.out
```

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

参考資料

Web セミナー (英語):

https://software.intel.com/videos/getting-the-most-out-of-the-intel-compiler-with-new-optimization-reports
https://software.intel.com/videos/new-vectorization-features-of-the-intel-compiler
https://software.intel.com/videos/data-alignment-padding-and-peel-remainder-loops

Fortran における明示的なベクトル・プログラミング (英語):

https://software.intel.com/articles/explicit-vector-programming-in-fortran

インテル® Xeon Phi™ コプロセッサーに関する記事 (ほかにも適用可能):

http://www.isus.jp/article/mic-article/vectorization-essential/

http://www.isus.jp/article/mic-article/fortran-array-data/

インテル® コンパイラー・ユーザー・リファレンス・ガイド (英語):

https://software.intel.com/intel-fortran-compiler-16.0-user-and-reference-guide

ベクトル化ガイド: http://www.isus.jp/article/compileroptimization/guide-to-auto-vectorization/

ユーザーフォーラム (英語): https://software.intel.com/forums/intel-visual-fortran-compiler-for-windows
https://software.intel.com/en-us/forums/intel-fortran-compiler-for-linux-and-mac-os-x

最適化に関する注意事項
© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

法務上の注意書きと最適化に関する注意事項

本資料の情報は、現状のまま提供され、本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他、知的財産権の侵害への保証を含む) をするものではありません。

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。

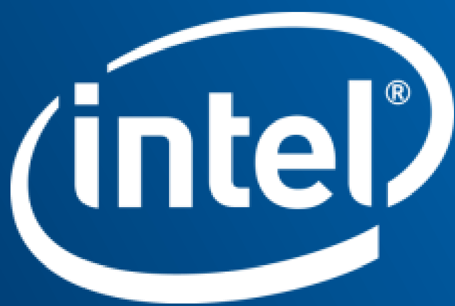
© 2016 Intel Corporation. 無断での引用、転載を禁じます。Intel、インテル、Intel ロゴ、Intel Core、Intel Xeon Phi、Xeon は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

最適化に関する注意事項

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。

注意事項の改訂 #20110804

© 2016 Intel Corporation. 無断での引用、転載を禁じます。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。


intel