ハイパフォーマンス・ソフトウェア・カンファレンス

コードの現代化と最適化

OpenMP* プログラミング・モデルによるスレッド化とベクトル化でパフォーマンスを改善

2017 年 12 月
iSUS 編集長
すがわら　きよふみ

ハードウェアの進化はソフトウェア開発に影響

コア数の増加とベクトルレジスター幅の拡大は、
より多くのスレッド数と最大パフォーマンスの向上を意味します
… しかし、ソフトウェア開発者は、これらの利点を活用するように
ソフトウェアを記述する必要があります

スレッド数の増加はスピードアップの可能性を意味します

インテル® Xeon® プロセッサー	64 ビット	5100 番台	5500 番台	5600 番台	E5-2600	E5-2600 v2	E5-2600 v3	E5-2600 v4	Platinum 8180
コア数	1	2	4	6	8	12	18	22	28
スレッド数	2	2	8	12	16	24	36	44	56
SIMD 幅	128	128	128	128	256	256	256	256	512

© 2017 iSUS *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® 開発ツールはコードの現代化を支援

- **最適化ツールとライブラリーの活用:** チューニングを始めましょう!
  - ビルド: 最適なコードを生成するため、インテル® コンパイラー (ホットスポットで) を使用します
  - ビルド: 利用可能であれば、インテル® マス・カーネル・ライブラリー (インテル® MKL)、インテル® スレッディング・ビルディング・ブロック (インテル® TBB) そしてインテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) などの最適化されているパフォーマンス・ライブラリーを使用します
  - プロファイル: インテル® VTune™ Amplifier でホットスポットを特定します
    インテル® Advisor でベクトル化とスレッド化の可能性を調査します
- **スカラー、シリアルの最適化:**
  コンパイラーとライブラリーの適切な精度、オプション、そして適切な機能を使用します
- **ベクトル化:**
  データレイアウトの最適化とともにコンパイラーとライブラリーの自動ベクトル化と SIMD 機能を使用します
- **並列化パート 1 - スレッド (共有メモリー) 並列:**
  インテル® Inspector、インテル® VTune™ Amplifier およびインテル® コンパイラーを使用してスレッドのスケーリングをプロファイルし、スレッドのコアへのアフィニティーを制御します
- **並列化パート 2 - マルチコアからメニーコアへ (分散メモリー並列):**
  インテル® Trace Analyzer & Collector とインテル® MPI ライブラリーを使用して、アプリケーションをプロファイルし分散メモリーにスケールさせます

© 2017 iSUS *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

- サンプルカーネルの最適化
  - プロセッサー・タイプ
  - メモリーアクセス
  - スレッド化
    - 同期
  - スレッド・アフィニティー
  - 明示的なベクトル化
  - その他

© 2017 iSUS *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。

サンプルプログラム

- 部分的なサブイメージを大きなイメージから検索します
  - 力作業ですべての位置をテスト
  - 絶対差の最小和を探す
    - 強度と色指数において
- テストには、ランダムに背景を重ね合わせたランダムパターンを使用します
- C と Fortran のソースを用意しました
  - 両者における問題と最適化はほぼ同じです
  - また、Windows* と Linux* バージョンも同様です (ここでは Windows* の結果を示します)
- ここでは、インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) をサポートする、インテル® Core™ i7-7800X X シリーズ・プロセッサー、6 コア、3.50GHz を使用しています

入手方法と構成

© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;   // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
    & feature, S_TYPE & sabdmini, S_TYPE & sabdminc)
{
        A_TYPE x, y, u, v, result;
        S_TYPE sabdmin, sabdi, sabdc;
        sabdminc = sabdmini = sabdmin =
                (S_TYPE) 2000000000;
// イメージの可能な位置をループで処理
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-
feature.ymax+1); y++) {
                        sabdi = 0;
                        sabdc = 0;
// 特徴画素を処理するループ
  for(u = 0; u < feature.xmax; u++) {
// 強度と色指数に対する絶対差の和を計算
      for(v = 0; v < feature.ymax; v++) {
          A_TYPE uv = u + v*feature.xmax;
          A_TYPE xy = x+u + (y+v)*image.xmax;
          sabdi = sabdi + abs(int(feature.intens[uv] -
          image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
          if(feature.intens[uv] > IMIN) {
                  if(image.intens[xy] < IMAX)
                          sabdc = sabdc + abs(int(feature.color[uv] -
                          image.color[xy]));
                  else
                          sabdc = sabdc + CMAX;
                  }
          } // v ループの終わり
      } // u ループの終わり
// 最小値の検出と記録
                  if (sabdi + sabdc < sabdmin) {
                                  result = x + y*image.xmax;
                                  sabdmin = sabdi + sabdc;
                                  sabdmini = sabdi;
                                  sabdminc = sabdc;
                  }
          }
    } // y ループの終わり
  } // x ループの終わり
          return result;
}
```


© 2017 iSUS *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。

ベースライン

$ icl /Qstd=c++14 /O2 set_seed.cpp init.cpp driver.cpp match.cpp /Qopt-report3 /Qansi-alias /Fesample1

sample1

```
  LOOP BEGIN at match.cpp(35,5)
      <Distributed chunk1>
        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。
analyze 1 images
SUCCESS image 0 match at (x,y) 387709 ref 387709 sabd=32350.000000 7064.000000
time per image  6.79
```

- “最小公約数” 命令セットはデフォルト (インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)) になります
  - しかし、ここで使用するインテル® Core™ i7-7800X X シリーズ・プロセッサー (開発コード名: “Skylake-X”) は、インテル® AVX2 をサポート

$ icl /Qstd=c++14 /O2 /Qxcore-avx2 set_seed.cpp init.cpp driver.cpp match.cpp /Qopt-report3 /Qansi-alias /Fesample2

sample2

```
  LOOP BEGIN at match.cpp(35,5)
      <Distributed chunk1>
        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。
analyze 1 images
SUCCESS image 0 match at (x,y) 91526 ref 91526 sabd=31112.000000 6578.000000
time per image 6.87
```

- ループがベクトル化された部分と、自動ベクトル化ではベクトル化できない部分に分割されたため、“部分的な (Partial)” ベクトル化が適用されています

© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;   // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
    & feature, S_TYPE & sabdmini, S_TYPE & sabdminc)
{
        A_TYPE x, y, u, v, result;
        S_TYPE sabdmin, sabdi, sabdc;
        sabdminc = sabdmini = sabdmin =
                (S_TYPE) 2000000000;
// イメージの可能な位置をループで処理
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-
feature.ymax+1); y++) {
                        sabdi = 0;
```

```
// 特徴画素を処理するループ
   for(u = 0; u < feature.xmax; u++) {
// 強度と色指数に対する絶対差の和を計算
      for(v = 0; v < feature.ymax; v++) {
         A_TYPE uv = u + v*feature.xmax;
         A_TYPE xy = x+u + (y+v)*image.xmax;
         sabdi = sabdi + abs(int(feature.intens[uv] -
         image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
         if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] -
                        image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
                }
         } // v ループの終わり
      } // u ループの終わり
// 最小値の検出と記録
                if (sabdi + sabdc < sabdmin) {
                        result = x + y*image.xmax;
                        sabdmin = sabdi + sabdc;
                        sabdmini = sabdi;
                        sabdminc = sabdc;
                }
}
```

```
ループの開始 C:¥2017_c¥avx512¥match.cpp(35,4)
    リマーク #25451: アドバイス: ループ交換は (可能であれば) ループの入れ子に適用できます。推奨する順列 : ( 1 2 ) --> ( 2 1 )
    リマーク #15541: 外部 ループ は自動ベクトル化されませんでした: SIMD ディレクティブの使用を検討してください。
```


© 2017 iSUS *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。

メモリーアクセス

“ストライドロード” は、メモリーへのアクセス効率が悪いことを意味します

```
for(u = 0; u < feature.xmax; u++) {
…
        for(v = 0; v < feature.ymax; v++) {
                A_TYPE uv = u + v*feature.xmax;
                A_TYPE xy = x+u + (y+v)*image.xmax;
                sabdi = sabdi + abs(int(feature.intens[uv] - image.intens[xy])); // 行 37
```

- u で連続したメモリーアクセスを行うため、u と v ループを入れ替えます

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 set_seed.cpp init.cpp driver.cpp match1.cpp /Qopt-report3 /Qansi-alias /Fesample3
```

sample3

```
  LOOP BEGIN at match1.cpp(34,5)
        リマーク #15344: ループはベクトル化されませんでした: ベクトルの依存関係がベクトル化を妨げています…
        リマーク #15346: ベクトル依存関係: FLOW の依存関係が sabdc (44:8) と sabdc (42:8) の間に仮定されました。
analyze 1 images
SUCCESS image 0 match at (x,y) 305101 ref 305101 sabd=32316.000000 6711.000000
time per image 4.90
```


© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;  // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
    & feature, S_TYPE & sabdmini, S_TYPE & sabdminc)
{
        A_TYPE x, y, u, v, result;
        S_TYPE sabdmin, sabdi, sabdc;
        sabdminc = sabdmini = sabdmin =
                (S_TYPE) 2000000000;
// イメージの可能な位置をループで処理
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
```

```
// 特徴画素を処理するループ
 for(v = 0; v < feature.ymax; v++) {
// 強度と色指数に対する絶対差の和を計算
   for(u = 0; u < feature.xmax; u++) {
        A_TYPE uv = u + v*feature.xmax;
        A_TYPE xy = x+u + (y+v)*image.xmax;
        sabdi = sabdi + abs(int(feature.intens[uv] -
        image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
        if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] -
                        image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
                }
        } // u ループの終わり
    } // v ループの終わり
// 最小値の検出と記録
                if (sabdi + sabdc < sabdmin) {
                        result = x + y*image.xmax;
                        sabdmin = sabdi + sabdc;
```

```
ループの開始 C:¥2017_c¥match1.cpp(34,5)
        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。最初の依存関係を以下に示します。詳細については、レベル 5 のレポートを使用してください。
        リマーク #15346: ベクトル依存関係: ANTI の依存関係が sabdc (42:8) と sabdc (42:8) の間に仮定されました。
        リマーク #25439: 剰余ありアンロール - 2
    ループの終了
```


© 2017 iSUS *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。

ループ分散

コンパイラーは改善された内部ループが、安全にベクトル化できることを判断できません

- これは、複数の if 文でリダクションが行われているためです

そして、ループ分散の有効性を判断できませんでした

- ”distribute point” プラグマを使用して、リダクションを 2 つの異なるループに分割することを指示できます

```
  sabdi = sabdi + abs(feature.intens[uv] - image.intens[xy]);        // チャンク 1 の終わり
#pragma distribute point
  if(feature.intens[uv] > IMIN) {                                    // チャンク 2 の始まり
        if(image.intens[xy] < IMAX)
              sabdc = sabdc + abs(int(feature.color[uv] - image.color[xy])); // 行 42
```

- 最初のチャンク、sabdi = sabdi + … は単純なリダクションであり、ユニットストライドでベクトル化可能です

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 set_seed.cpp init.cpp driver.cpp match1a.cpp /Qopt-report3 /Qansi-alias /Fesample4
  LOOP BEGIN at match1a.cpp(34,5)
      <Distributed chunk1>
        リマーク #15301: 部分ループがベクトル化されました。
        リマーク#15450: マスクなしの非アライメント ユニット・ストライド・ロード: 2
analyze 1 images
SUCCESS image 0 match at (x,y) 526873 ref 526873 sabd=31868.000000 7081.000000
time per image 4.09
```

sample4

© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;  // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
```

```
ループの開始 C:¥2017_c¥match1a.cpp(34,5)
    <分散チャンク1>
      リマーク #25426: ループ分散 (2 ウェイ)
      リマーク #25428: 行 34 で大規模な ii 用に分散されました。
[ C:¥2017_c¥match1a.cpp(34,5) ]
      リマーク #15301: 部分ループ がベクトル化されました。
      リマーク #15442: すべてのループは剰余ループとして実行されます。
      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: 2
      リマーク #15475: --- ベクトルのコストサマリー開始 ---
      リマーク #15476: スカラーのコスト: 14
      リマーク #15477: ベクトルのコスト: 1.250
      リマーク #15478: スピードアップの期待値: 7.420
      リマーク #15487: 型変換: 1
      リマーク #15488: --- ベクトルのコストサマリー終了 ---
    ループの終了
```

```
// 特徴画素を処理するループ
 for(v = 0; v < feature.ymax; v++) {
// 強度と色指数に対する絶対差の和を計算
   for(u = 0; u < feature.xmax; u++) {
        A_TYPE uv = u + v*feature.xmax;
        A_TYPE xy = x+u + (y+v)*image.xmax;
        sabdi = sabdi + abs(int(feature.intens[uv] -
        image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
#pragma distribute point
        if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] -
                        image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
                }
u ループの終わり
ープの終わり
出と記録
                if (sabdi + sabdc < sabdmin) {
                                result = x + y*image.xmax;
                                sabdmin = sabdi + sabdc;
                                sabdmini = sabdi;
                                sabdminc = sabdc;
                }

プの終わり
プの終わり
n result;
```


© 2017 iSUS *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。

スレッド化

OpenMP* プラグマを使用して簡単にスレッド化を実装できます

- 最も上位レベル (最外のループ) でスレッド化するのが最も効果的です

```
// イメージの可能な位置をループで処理
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)          // 行 26
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-feature.ymax+1); y++) {…
```

何が起こるか見てみましょう ....

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cp driver.cpp match2.cpp /Qopt-report3 /Qansi-alias /Fesample5
match2.cpp(26:1-26:1):OMP:_Z5matchR10image_dataS0_RfS1_: OpenMP 定義ループが並列化されました。
…
analyze 1 images
FAILED image 0 match at (x,y) 435900 ref 340101 sabd=285190.000000 36914.000000
time per image 0.68
```

sample5

© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;  // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
    & feature, S_TYPE & sabdmini, S_TYPE & sabdminc)
{
        A_TYPE x, y, u, v, result;
        S_TYPE sabdmin, sabdi, sabdc;
        sabdminc = sabdmini = sabdmin =
                (S_TYPE) 2000000000;
// イメージの可能な位置をループで処理
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-
feature.ymax+1); y++) {
                        sabdi = 0;
                        sabdc = 0;
// 特徴画素を処理するループ
 for(v = 0; v < feature.ymax; v++) {
// 強度と色指数に対する絶対差の和を計算
  for(u = 0; u < feature.xmax; u++) {
        A_TYPE uv = u + v*feature.xmax;
        A_TYPE xy = x+u + (y+v)*image.xmax;
        sabdi = sabdi + abs(int(feature.intens[uv] -
        image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
#pragma distribute point
        if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] -
                        image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
                }
        } // u ループの終わり
    } // v ループの終わり
// 最小値の検出と記録
                if (sabdi + sabdc < sabdmin) {
                        result = x + y*image.xmax;
                        sabdmin = sabdi + sabdc;
                        sabdmini = sabdi;
                        sabdminc = sabdc;
                }
        }
  } // y ループの終わり
 } // x ループの終わり
        return result;
}
```

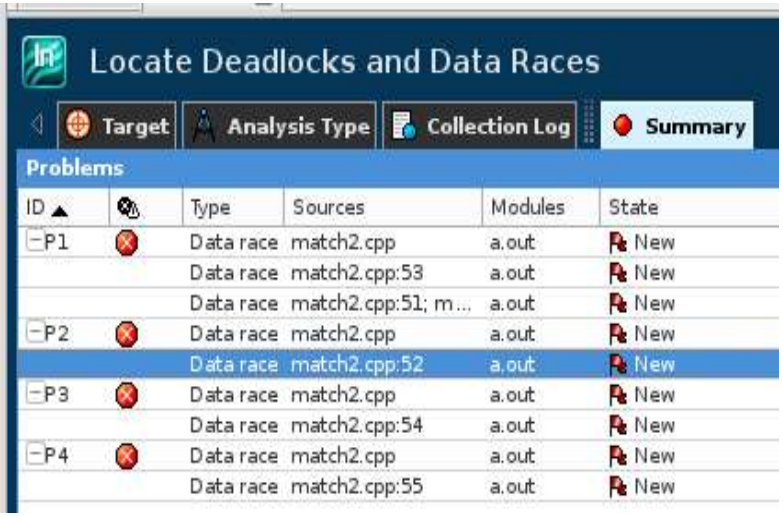

© 2017 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

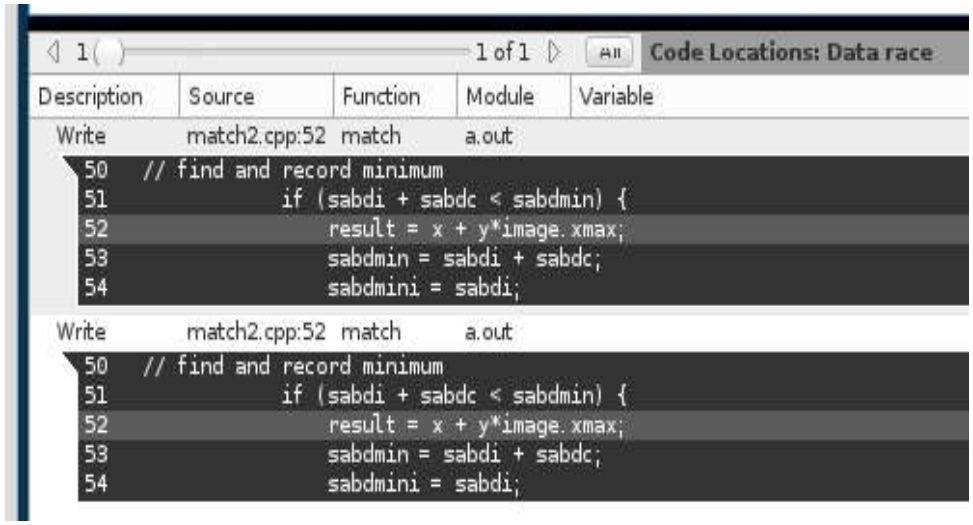
スレッドのエラーを検出

インテル® Inspector を実行: inspxe-cl -collect=ti3 -- ./a.out

Warning: One or more threads in the application accessed the stack of another thread.
(アプリケーション中の複数のスレッドがほかのスレッドのスタックをアクセスしています)
4 Data race problem(s) detected (4 つのデータ競合問題が検出されました)

GUI で確認します:





© 2017 iSUS *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。

スレッド化の問題を解決

OpenMP* の critical 句を使用して保護

- 一度に 1 スレッドのみを実行することができます
- この部分は低速ですが、それ以外の領域は並列に実行されます

```
// 最小値の検出と記録
#pragma omp critical
{
        if (sabdi + sabdc < sabdmin) {
                result = x + y*image.xmax;
                sabdmin = sabdi + sabdc;
                sabdmini = sabdi;
                sabdminc = sabdc;
        }
}
```

```
$ icl -std=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cpp driver.cpp match2a.cpp
                                                        /Qansi-alias /Qopt-report3 /Fesample6
match2a.cpp(51:1-51:1):OMP:_Z5matchR10image_dataS0_RfS1_: CRITICAL の OpenMP マルチスレッド･コードが正常に生成されました。
match2a.cpp(26:1-26:1):OMP:_Z5matchR10image_dataS0_RfS1_: OpenMP 定義ループが並列化されました。
…
analyze 10 images
SUCCESS image 0 match at (x,y) 87916 ref 87916 sabd=31800.000000 6563.000000 …
time per image 0.68
```

sample6

Critical 句 (クリティカル･セクション) がパフォーマンスに影響

- しかし、スレッド化されていないバージョンよりも高速

© 2017 iSUS *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。

SAD カーネル (C)

```
struct image_data {
        P_TYPE *intens; // intensity (x, y)
        P_TYPE *color;  // color (x, y)
        A_TYPE xmax;    // x のイメージサイズ (ピクセル数)
        A_TYPE ymax;    // y のイメージサイズ (ピクセル数)
};

int match(struct image_data & image, struct image_data
    & feature, S_TYPE & sabdmini, S_TYPE & sabdminc)
{
        A_TYPE x, y, u, v, result;
        S_TYPE sabdmin, sabdi, sabdc;
        sabdminc = sabdmini = sabdmin =
                (S_TYPE) 2000000000;
// イメージの可能な位置をループで処理
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)
        for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-
feature.ymax+1); y++) {
                        sabdi = 0;
                        sabdc = 0;
// 特徴画素を処理するループ
 for(v = 0; v < feature.ymax; v++) {
// 強度と色指数に対する絶対差の和を計算
  for(u = 0; u < feature.xmax; u++) {
        A_TYPE uv = u + v*feature.xmax;
        A_TYPE xy = x+u + (y+v)*image.xmax;
        sabdi = sabdi + abs(int(feature.intens[uv] -
        image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
#pragma distribute point
        if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] -
                        image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
                }
        } // u ループの終わり
    } // v ループの終わり
// 最小値の検出と記録
#pragma omp critical
                if (sabdi + sabdc < sabdmin) {
                        result = x + y*image.xmax;
                        sabdmin = sabdi + sabdc;
                        sabdmini = sabdi;
                        sabdminc = sabdc;
                }
        }
  } // y ループの終わり
 } // x ループの終わり
        return result;
}
```


© 2017 iSUS *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® VTune™ Amplifier

Top Hotspots

This section lists the most active functions in your application.

Function	Module	CPU Time
__kmpc_critical_with_hint	libiomp5.so	336.975s
match_ompparallel_for@28	a.out	86.990s
__kmp_fork_barrier	libiomp5.so	2.747s
__kmp_release_queuing_lock	libiomp5.so	2.690s
__kmpc_end_critical	libiomp5.so	0.180s

Elapsed Time: 8.003s

CPU Time:	429.902s
Effective Time:	87.170s
Spin Time:	342.482s
Imbalance or Serial Spinning:	25.917s
Lock Contention:	306.642s
Other:	9.923s
Overhead Time:	0.250s
Total Thread Count:	56
Paused Time:	0s

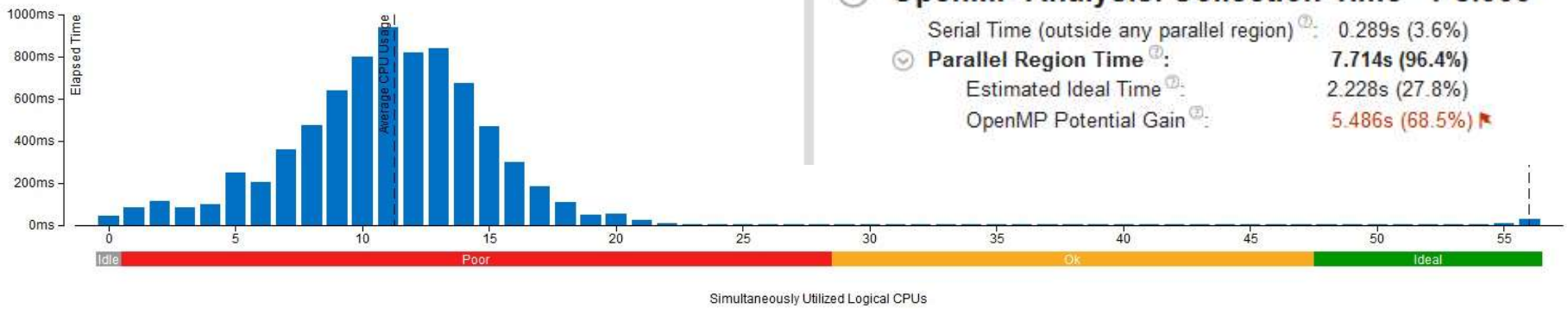
OpenMP Analysis. Collection Time: 8.003

Serial Time (outside any parallel region):	0.289s (3.6%)
Parallel Region Time:	7.714s (96.4%)
Estimated Ideal Time:	2.228s (27.8%)
OpenMP Potential Gain:	5.486s (68.5%)

OpenMP Region CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously in an OpenMP region (Elapsed Time) so it is recommended to start exploration from the top.

OpenMP Region: match_ompparallel:56@unknown:28:63



さらに効率良く: ループを分散 (分割)

critical 句 (クリティカル･セクション) にはそれほど多くの処理は含まれていません

- シリアル実行よりは高速
- スレッド間の同期を排除します

sample7

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cpp
driver.cpp match3.cpp /Qqopt-report3 /Qansi-alias /Fesample7
match3.cpp(29:1-29:1): OMP:_Z5matchR10image_dataS0_RfS1_:
        OpenMP 定義ループが並列化されました。

analyze 10 images
SUCCESS image 0 match at (x,y) 648425 ref 648425 …
…
time per image 0.67
```

- critical 句 (クリティカル･セクション) よりも高速

```
// イメージの可能な位置をループで処理
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)
  for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
    for (y = 0; y < (image.ymax-feature.ymax+1); y++) {
      …
      for(v = 0; v < feature.ymax; v++) {
        for(u = 0; u < feature.xmax; u++) {
        #pragma distribute point
          sabdi = sabdi + …
          …
          sabdc = sabdc + …
            }
      }
      A_TYPE xy = x + y*(image.xmax-feature.xmax+1);
      sabdi_xy[xy] = sabdi;
      sabdc_xy[xy] = sabdc;
    }
  }
// 最小値の検出と記録
for(x = 0; x < (image.xmax-feature.xmax+1); x++) {
  for (y = 0; y < (image.ymax-feature.ymax+1); y++) {
    int xy = x + y*(image.xmax-feature.xmax+1);
    if (sabdi_xy[xy] + sabdc_xy[xy] < sabdmin) {
      sabdmin  = sabdi_xy[xy] + sabdc_xy[xy];
      result   = x + y*image.xmax;
      …
    }
  }
}
```


© 2017 iSUS *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。

スレッド･アフィニティー

デフォルトで、OS はスレッドを 1 つのコアから他のコアへ移動できます

- 可変なワークロードとスレッド数には向いていますが、対称的なワークロードには向いていません
- 実行時間の変動につながります。例えば: .26, .20, .22, .24, .25

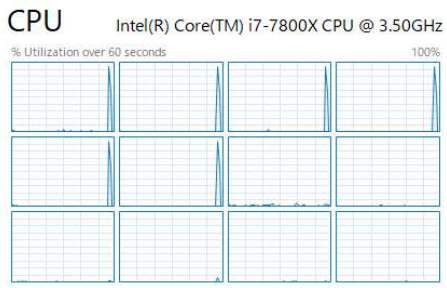
スレッドをコアにバインドすることで改善できます
例えば: set OMP_PROC_BIND=close または set KMP_AFFINITY=compact

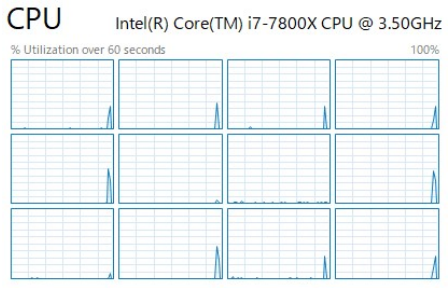
同じバイナリーを実行:

time per image .19
time per image .19
time per image .19
time per image .23

- より高速で一貫性があります

KMP_AFFINITY=compact	KMP_AFFINITY=scatter
1.30 (6 スレッド)	0.74 (6 スレッド)





© 2017 iSUS *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。

「ベクトル化レポート」に戻る

sample8

icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp match3.cpp /Qopt-report2 /c
/Qansi-alias /Qqopt-report-phase=loop,vec

```
LOOP BEGIN at match3.cpp(30,2)
  リマーク #15542: ループはベクトル化できませんでした:内部ループはベクトル化されています。
  LOOP BEGIN at match3.cpp(31,3)
    リマーク #15542: ループはベクトル化できませんでした: 内部ループはベクトル化されています。
    LOOP BEGIN at match3.cpp(36,4)
      リマーク #15542: ループはベクトル化できませんでした: 内部ループはベクトル化されています。
      …
      LOOP BEGIN at match3.cpp(38,5)
      <Distributed chunk1>
        リマーク #15301: 部分ループがベクトル化されました。
      LOOP END
      …
      LOOP BEGIN at match3.cpp(38,5)
      <Distributed chunk2>
        リマーク #15344: ループはベクトル化されませんでした: ベクトルの依存関係がベクトル化を妨げています最初の依存関係を以下に示します。
        さらに詳しい情報を取得するため、レベル 5 のレポートを生成します。
        リマーク #15346: ベクトル依存関係: FLOW の依存関係が sabdc (48:8) と sabdc (46:8) の間に仮定されました。
      LOOP END
…
    LOOP END
  LOOP END
LOOP END
```

最初のループのみがベクトル化されたため、それほどスピードアップはありません。
インテル® Advisor も利用できますが、重要な情報は最適化レポートに含まれています。
この場合、どのループに注目すべきかインテル® Advisor がなくても分かります。

© 2017 iSUS *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。

ベクトル化を再考察

コンパイラーは、赤字のコードが複数の慣用句を結合しているためベクトル化できませんでした

チャンク 2:

```
for(u = 0; u < feature.xmax; u++) {
        A_TYPE uv = u + v*feature.xmax;
        A_TYPE xy = x+u + (y+v)*image.xmax;
        sabdi = sabdi + abs(int(feature.intens[uv] - image.intens[xy]));
// 空の特徴画素を除き、飽和したイメージ画素を除去
        if(feature.intens[uv] > IMIN) {
                if(image.intens[xy] < IMAX)
                        sabdc = sabdc + abs(int(feature.color[uv] - image.color[xy]));
                else
                        sabdc = sabdc + CMAX;
        }
}
```

- 2 つのリダクション、2 つの条件と “else” 節
- 個別に識別できますが、すべて同時にはできません (sabdc を依存性として識別)
  - プログラマーの支援が必要 …

© 2017 iSUS *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。

明示的なベクトル化

OpenMP* SIMD ディレクティブを使用して、コンパイラーにベクトル化の方法を指示:

#pragma omp simd reduction(+:sabdi,sabdc)

- コンパイラーに **sabdi**、**sabdc** をリダクション変数として扱い、ベクトル化することを指示

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cpp driver.cpp match4.cpp /Qopt-report2
                                                                    /Qansi-alias /Fesample9
```

sample9

```
…
    LOOP BEGIN at match4.cpp(39,5)
        リマーク #15301: OpenMP SIMD LOOP がベクトル化されました。
    LOOP END
…
SUCCESS image 0 match at (x,y) 78922 ref 78922 sabd=31657.000000 6084.000000
…
time per image 0.20
```

- ループはもはや分割されず、ループ全体がベクトル化されます
- SIMD は非常に強力ですが、正当性の保証はプログラマーが行う必要があります
  - REDUCTION、PRIVATE などの節で示される変数以外に依存関係はありません

© 2017 iSUS *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。

さらに細かな最適化 (1)

最外の x と y ループをインターチェンジすることで、データの局所性を最善できます:

sample10

$ icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cpp driver.cpp match5.cpp /Qopt-report3 /Qansi-alias /Fesample10

SUCCESS image 0 match at (x,y) 376316 ref 376316 sabd=31593.000000 6598.000000
time per image 0.19

```
// イメージ内の有効な位置に対するループ
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)
        for (y = 0; y < (image.ymax-feature.ymax+1); y++) {
                for (x = 0; x < (image.xmax-feature.xmax+1); x++) {
                        sabdi = 0;
                        sabdc = 0;
```

```
// イメージ内の有効な位置に対するループ
#pragma omp parallel for private(sabdi,sabdc, x, y, u, v)
        for (x = 0; x < (image.xmax-feature.xmax+1); x++) {
                for (y = 0; y < (image.ymax-feature.ymax+
                        sabdi = 0;
                        sabdc = 0;
```

ループの開始 C:¥2017_c¥match4.cpp(60,2)
　リマーク #25096: ループ交換は完了しませんでした: ループの入れ子が完全ではありません (ソースのいずれかまたは他のコンパイラー変換による)
　リマーク #25451: アドバイス: ループ交換は (可能であれば) ループの入れ子に適用できます。推奨する順列 : (1 2) -> (2 1)

© 2017 iSUS *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。

さらに細かな最適化 (2)

Sample10 のベクトル化レポートに次の行が含まれます:

remark #15487: 型変換: 2

- これは、すべてのリダクション変数が浮動小数点型であり、else 節では整数型であることを知るヒントとなります
- sabdi と sabdc を整数として宣言 (S_TYPE を変更しますが、オーバーフローしないことを確認)!
- unsiged int を int にすることで最適化に貢献 (P_TYPE, A_TYPE - image_data5.h を参照)
  - 符号なし整数のオーバーフローを考慮しなければいけません。符号付き整数は必要ありません

```
#define P_TYPE unsigned int  // used for pixels
#define A_TYPE unsigned int  // used for addressing (array indices)
#define S_TYPE float         // used for summations
```

```
ループの開始 C:¥2017_c¥match5.cpp(43,5)
   リマーク #15487: 型変換: 2
ループの終了
```

```
$ icl /Qstd=c++14 /O2 /Qxcore-avx2 /Qopenmp set_seed.cpp init.cpp driver.cpp match5a.cpp /Qopt-report3
                                                                                /Qansi-alias /Fesample11
SUCCESS image 0 match at (x,y) 867713 ref 867713 sabd=31004.000000 6241.000000
...
time per image 0.15
```

sample11

- 最適化の効果により、型変換の警告はなくなりました。

© 2017 iSUS *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。

まとめ

- デフォルトの最適化と比較して、系統的な最適化は劇的なスピードアップをもたらします (このシステムとカーネルでは、40 倍以上)
- 最適化なしの -O0 比較すると、130 倍以上
- 試みた価値があります!

最適化	ソース	イメージごとの時間 (秒)
-O2 のベースライン	match	6.79 (sample1)
-xcore-avx2	match	6.87 (sample2)
ユニットストライド	match1	4.90 (sample3)
部分的なベクトル化	match1a	4.09 (sample4)
OpenMP* + critical	match2a	0.68 (sample6)
distribute	match3	0.67 (sample7)
SIMD ベクトル化	match4	0.20 / 0.15(sample9)
外部ループを入れ換え	match5	0.19 / 0.14(sample10)
型変換なし	match5a	0.15 / 0.13(sample11)

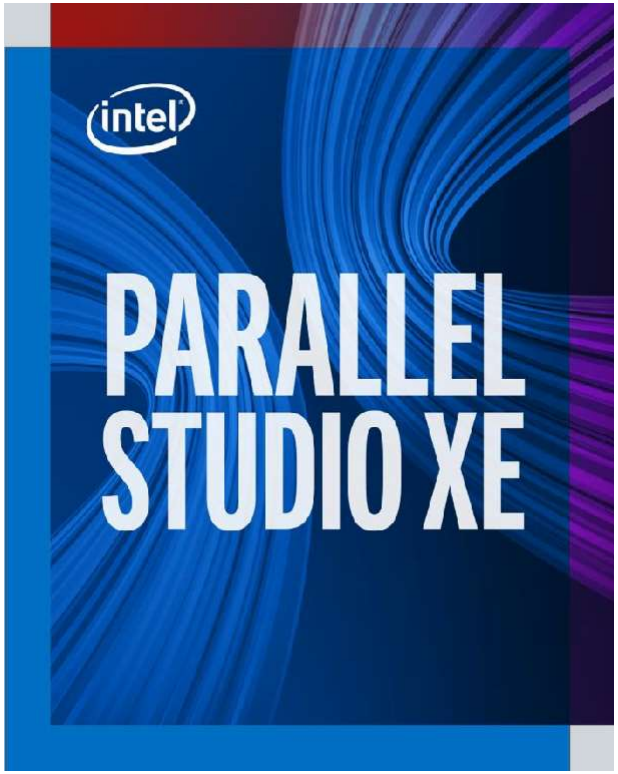
性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。

© 2017 iSUS *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。

パフォーマンスに優れたコード

インテル® Parallel Studio XE 2018
30 日間無料評価版をダウンロード:

makebettercode.com/parallelstudioxe-eval (英語) ›



© 2017 iSUS *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。

関連情報

OpenMP* に関する情報

OpenMP* による SIMD を利用するベクトル･プログラミング

http://jp.xlsoft.com/documents/intel/catalog/Intel_ParallelUniverse_Issue22_JPN.pdf

OpenMP* の現在と将来

https://jp.xlsoft.com/documents/intel/magazine/Intel_ParallelUniverse_Issue27_JPN.pdf

インテル® Xeon Phi™ プロセッサー向けにコードを現代化

https://jp.xlsoft.com/documents/intel/magazine/Intel_ParallelUniverse_Issue26_JPN.pdf

コードモダン化のリンク

- モダンコード開発者コミュニティー
  http://www.isus.jp/modern-code/
- インテル® Code Modernization Enablement Program
  https://software.intel.com/code-modernization-enablement (英語)
- インテル® Parallel Computing Centers
  https://software.intel.com/ipcc (英語)
- 技術ウェビナーシリーズ
  https://software.intel.com/events/development-tools-webinars (英語)
- インテル® Parallel Universe マガジン (日本語訳)
  http://www.isus.jp/products/psxe/issue/

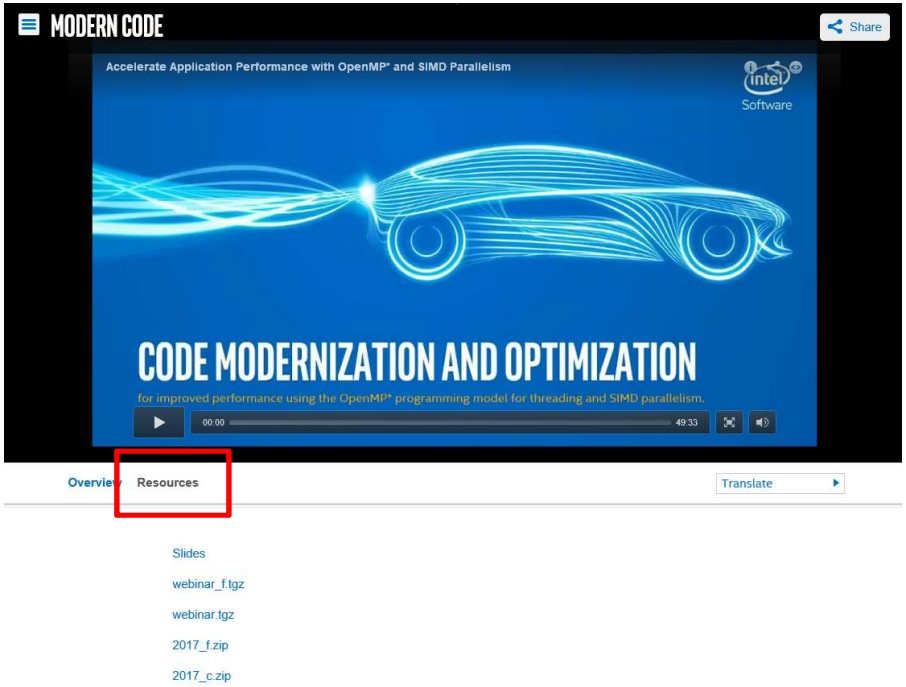
© 2017 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

サンプルコードの入手方法

インテル® デベロッパー・ゾーンの以下の URL をアクセスしてください:

https://software.intel.com/en-us/videos/accelerate-application-performance-with-openmp-and-simd-parallelism
(英語)

画面中ほどにある「Resources」をクリックすると、*tgz (Linux* 用) と *.zip (Windows* 用) のサンプルコードをダウンロードできます



© 2017 iSUS *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。

サンプルコードの構成

image_data.h	イメージデータのデータ構造定義
image_data5.h	sample11 用イメージデータ構造定義
driver.cpp	main ルーチン
init.cpp	初期化関数
set_seed.cpp	データ初期化用の乱数データ生成
match.cpp	sample1、sample2 用のソース
match1.cpp	sample3 用のソース
match1a.cpp	sample4 用のソース
match2.cpp	sample5 用のソース
match2a.cpp	sample6 用のソース
match3.cpp	sample7、sample8 用のソース
match4.cpp	sample9 用のソース
match5.cpp	sample10、sample11用のソース

page5 へ戻る

© 2017 iSUS *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。

役立つ記事

自動ベクトル化が失敗した場合の対処:
https://www.isus.jp/products/c-compilers/what-to-do-when-auto-vectorization-fails/
ベクトル化によるパフォーマンスの向上:
https://www.isus.jp/products/c-compilers/improve-performance-with-vectorization/
コンパイラー最適化入門: 第 1 回 SIMD 命令とプロセッサーの関係:
https://www.isus.jp/products/c-compilers/compiler_part1/
インテルのベクトル化ツール:
https://www.isus.jp/intel-vectorization-tools/
ケーススタディー: インテル® アドバンスト・ベクトル・エクステンションを使用したブラック-ショールズの計算:
https://www.isus.jp/financial/case-study-computing-black-scholes-with-avx/
インテルの最新の並列化ツールによる明示的なベクトル化:
https://www.isus.jp/products/psxe/vectorization-gets-explicit/
明示的なベクトル・プログラミング – 最も一般的な手法:
https://www.isus.jp/products/psxe/explicit-vector-programming-bkm/
ループのベクトル化によるプログラムの最適化:
https://www.isus.jp/products/c-compilers/program-optimization-through-loop-vectorization/

© 2017 iSUS *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。

お待たせしました

インテル® 64 アーキテクチャーおよび IA-32 アーキテクチャー最適化リファレンス・マニュアル (-038 Oct 2017 参考訳版)

インテル® VTune™ Amplifier 2017 オンラインヘルプ (iSUS 翻訳版)

完成しました: https://www.isus.jp/ からどうぞ!!

© 2017 iSUS *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。

著作権と商標についておよび最適化に関する注意事項

本資料に掲載されている情報は現状のまま提供され、いかなる保証もいたしません。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を許諾するための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。

© 2017 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、Xeon、Xeon Phi、VTune 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。

* 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。

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® 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。

改訂 #20110804

© 2017 iSUS *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。