The Parallel Universe

TRANSFORM YOUR CODE

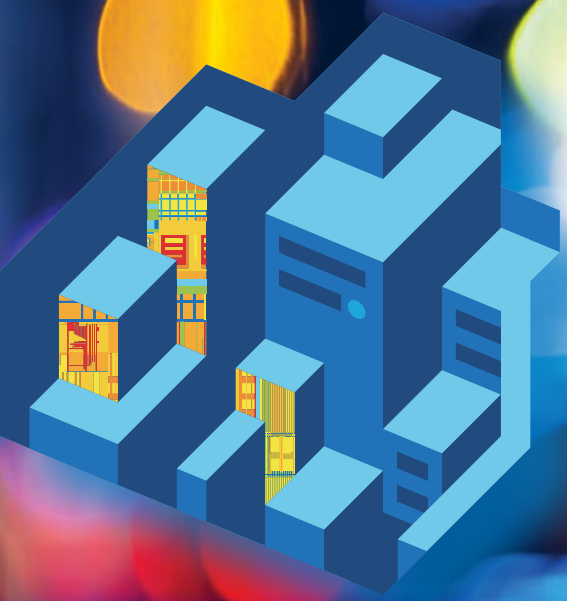
インテル® Parallel Studio XE 2015 リリース

パフォーマンスの最適化

スケーラブルな設計

intel Software

Issue 19 2014



intel Software

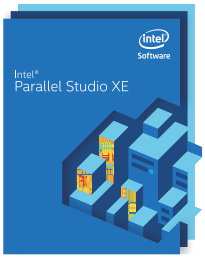
TRANSFORM YOUR CODE

インテル® Parallel Studio XE 2015

優れたアプリケーション・パフォーマンスと信頼性を実現

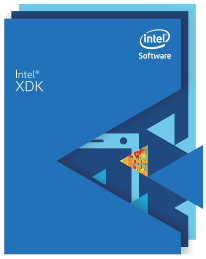
- **Composer Edition:** 高速な並列コードを作成できるように最適化された業界最先端のコンパイラーとライブラリー
- **Professional Edition:** Composer Edition の機能に加えて、最先端の解析ツールである インテル® VTune™ Amplifier XE、高度なメモリー/スレッド・エラー・チェックとチューニング機能
- **Cluster Edition:** Professional Edition の機能に加えて、MPI クラスター通信ライブラリー、MPI エラーチェックとチューニング・ユーティリティー

専門/組込み分野からメディア、HTML5、クロスプラットフォームまで幅広い分野に対応した開発ツール



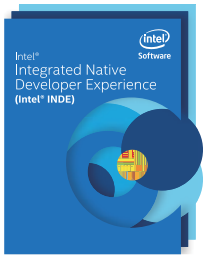
技術計算、
エンタープライズ、
HPC ソフトウェア

組込みシステム /
デバイス
開発ソフトウェア



Web アプリ /
HTML5
開発ソフトウェア

ゲーム / メディア
開発ソフトウェア



ネイティブアプリ
開発ソフトウェア

評価版:

intel.com/software/products

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、
http://software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

© 2015 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、Intel Xeon Phi、Cilk、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。

目次

編集者からのメッセージ 4

ソフトウェアが鍵を握る

James Reinders

注目記事

最適化レポート: インテル® コンパイラーによりパフォーマンスを向上 5

インテル® Parallel Studio XE 2015 のコンパイラーによる最適化レポートは、コードのチューニングとパフォーマンスの向上に役立ちます。
利用可能なレポートデータの種類とアプリケーションへの適用方法を紹介します。

マルチコアからメニーコアまでスケーラブルなパフォーマンスの設計 16

インテル® Advisor XE 2015 は、設計をモデル化し、パフォーマンス・スケーリングと同期の問題を予測するためのフレームワークを提供します。インテル® Advisor XE 2015 のモデル化をインテル® Xeon Phi™ コプロセッサーに拡張する方法を示します。

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) に追加された命令 23

インテル® AVX-512 で強化された新しい命令について簡単に説明します。これには、バイトおよびワード命令と呼ばれる、バイトとワード (8 ビットと 16 ビット) 型の整数データの操作を行う命令が含まれます。
ほとんどのインテル® AVX-512 命令は、ベクトル長の拡張機能により 128 ビットまたは 256 ビットを操作することができます。

Digimarc 社が新たに実現した電子透かしの埋め込み 26

Digimarc 社が最適化とベクトル化により SDK をアップグレードした際に、パフォーマンスを大幅に向上し、開発期間の短縮を実現した、電子透かしの革新の成功例を紹介します。

『High Performance Parallelism Pearls』 31

インテル コーポレーションの並列プログラミング・エバンジェリストである James Reinders とインテル コーポレーションのエンジニアである Jim Jeffers により執筆された最新の書籍では、69 人の専門家によるインテルのマルチコアおよびメニーコア・プロセッサーを最大限に利用するための創意工夫が 28 章からなる "クックブック" としてまとめられています。

© 2015 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、Intel Xeon Phi、Cilk、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。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の使用許諾を受けて使用しています。

編集者からのメッセージ

James Reinders インテル コーポレーションの並列プログラミング・エバンジェリスト兼ディレクター
新しい書籍『Multithreading for Visual Effects』(2014) の共著者で、このほかに、『Intel® Xeon Phi™ Coprocessor High Performance Programming』(2013、日本語 :『インテル® Xeon Phi™ コプロセッサー ハイパフォーマンス・プログラミング』)、『Structured Parallel Programming』(2012、日本語 :『構造化並列プログラミング』)、『Intel® Threading Building Blocks: Outfitting C++ for Multicore Processor Parallelism』(日本語 :『インテル スレッディング・ビルディング・ブロック ― マルチコア時代の C++ 並列プログラミング』、中国語、韓国語の翻訳版があります)、『VTune™ Performance Analyzer Essentials』(Intel Press, 2005) などの文献を発表しています。

ソフトウェアが鍵を握る

現代のハードウェアの機能と性能が最大限に利用されていないことは周知の事実です。ゲームデザイナーからビッグデータ・アナリストまで、最先端のアプリケーション開発者は、ソフトウェアによりハードウェア機能を活用しています。

開発者のニーズに応えるため、インテルでは最新のインテル® Parallel Studio XE 2015 の 3 つのエディションに新機能を追加しました。この号では、高速なコードを迅速に開発するための、いくつかの機能について説明します。

注目記事である「最適化レポート : インテル® コンパイラーによりパフォーマンスを向上」は、チューニングとパフォーマンス向上に役立つ、これまでにない詳細情報を提供します。最新リリースで提供されるレポートデータの種類と内容について説明し、これらの情報を開発プロセスに取り入れる方法を紹介します。

「マルチコアからメニーコアまでスケーラブルなパフォーマンスの設計」では、インテル® Xeon Phi™ コプロセッサーでのスケーラビリティーに取り組みます。特定のワークロードに対し、インテル® Xeon Phi™ コプロセッサーのパフォーマンスがインテル® Xeon® プロセッサーを上回るかどうか自動予測する解析ツールも紹介します。

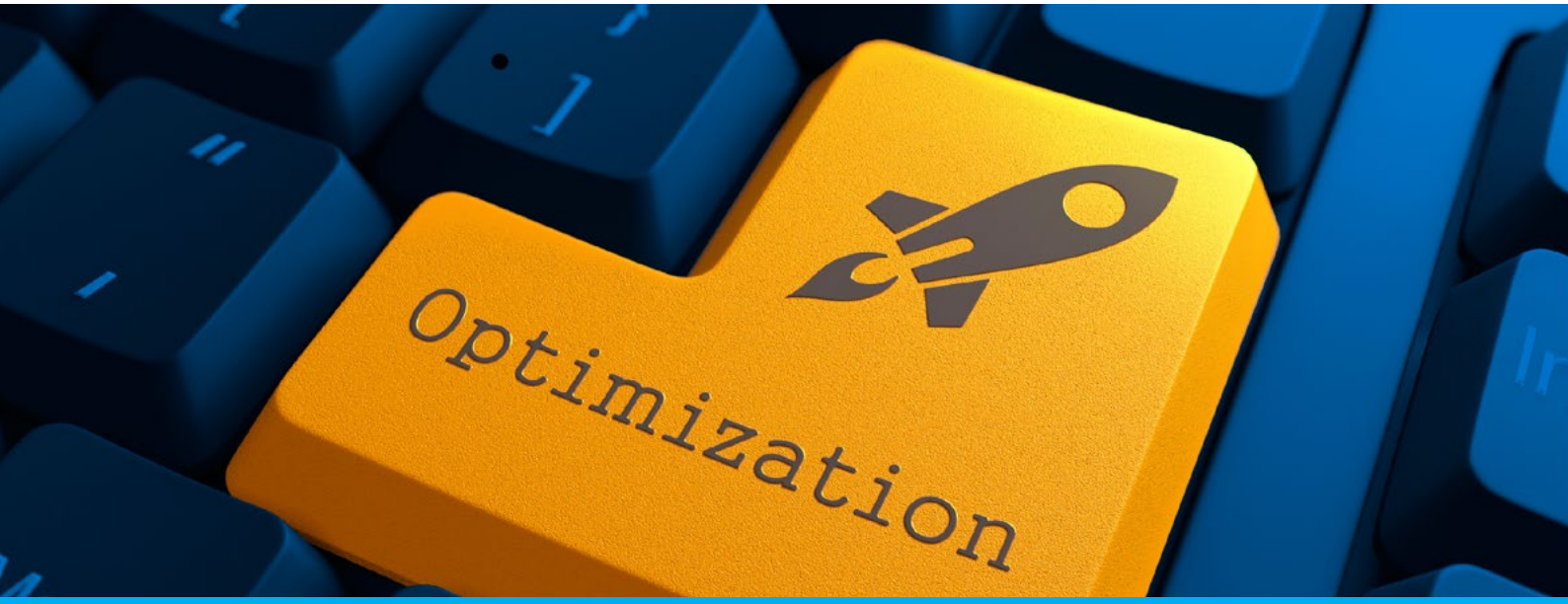
「インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) の追加命令」では、インテル® Xeon Phi™ コプロセッサーと将来のインテル® Xeon® プロセッサーでサポートされる新しい命令の概要を説明します。新機能や新たに追加された命令セットは、コンパイラーによるベクトル化の有用性と同様に、検討してみる価値があるでしょう。

"ハードウェアとソフトウェア" による最適化の効果を示すケーススタディー「Digimarc 社が新たに実現した電子透かしの埋め込み」では、インテル® ソフトウェア・ツールにより、Digimarc 社が SDK をアップグレードした際に、パフォーマンスを大幅に向上し、開発期間の短縮を実現した、電子透かしの革新の成功例を紹介します。

新しいツールを利用して、新たなチャンスをものにしてください。ここで紹介した記事が、皆さんのコーディングの課題を解決する鍵となることを願っています。

James Reinders
2014 年 9 月

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。



最適化レポート: インテル® コンパイラーによりパフォーマンスを向上

Martyn Corden インテル コーポレーション開発製品部門テクニカル・コンサルティング・エンジニア

アプリケーションの再コンパイルによって最適化を行う場合でも、最適化レポートを活用することで、さらなるパフォーマンスの向上を達成できます。最新のインテル® コンパイラーでは、レポート情報の取得がより簡単になりました。

最近の最適化コンパイラーは、パフォーマンスが向上するようにコードを構成できますが、どれだけ向上するかはソースコードの記述方法とコンパイラーに与える情報に依存します。インテル® コンパイラーの最適化レポートは、適用された最適化とされなかった最適化およびその理由を示します。この情報を基にコードを再構成したり、コンパイラー・オプションを追加して、コンパイラーによる最適化が適用されるようにすることで、アプリケーションのパフォーマンスをさらに向上できます。

以前のバージョンのインテル® コンパイラーでは、最適化に関する情報を異なるレポートで出力していました。しかし、メッセージが論理的に順序付けられておらず、特にインライン展開やコンパイラーが複数バージョンのループを生成する場合、メッセージが不明瞭であったり、紛らわしいことがありました。また、一部のメッセージはアクションがとりづらかったり、すぐに役立てることができないものがありました。個別に出力されたレポートは参照しづらく、ほかのツールからも利用しにくく、ビルド時間を短縮する際に利用される並列ビルドには適していませんでした。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

インテル® Parallel Studio XE 2015 に含まれる最新のインテル® コンパイラー 15.0 では、**最適化レポートが一新され、個々のレポートが 1 つにまとめられ、前述の制限が解消されたことでユーザー・フレンドリーなレポートになっています。**ここでは、新しい最適化レポートの機能と、コンパイラーによって適用された最適化を理解し、アプリケーションをさらにチューニングするための方法を紹介します。

レポートの有効化と制御

表 1 に、インテル® コンパイラー (Windows*、Linux*、OS X*) で最適化レポートとレポートを制御するコマンドライン・オプションを示します。多くの場合、Linux* と OS X* 用のコンパイラー・オプションは -q で始まり、Windows* 用のコンパイラー・オプション は /Q で始まります。オプション は、C/C++ コンパイラーと Fortran コンパイラーで共通です。

Linux* および OS X*	Windows*	機能
`-qopt-report[=N]`	`/Qopt-report[:N]`	レポートを有効にします。N=1-5 で詳細レベルを指定します (デフォルトは N=2)。
`-qopt-report-file=stdout | stderr | filename`	`/Qopt-report-file:stdout | stderr | filename`	レポートの出力先を指定します (デフォルトは拡張子が .optrpt のファイル)。
	`/Qopt-report-format:vs`	Microsoft* Visual Studio* 表示用にレポートをフォーマットします。
`-qopt-report-routine= fn1[,fn2,…]`	`/Qopt-report-routine: fn1[,fn2,…]`	名前の一部にキーワード fn1 [,fn2,…] を含む関数のみレポートを出力します。
`-qopt-report-filter= "filename,ln1-ln2"`	`/Qopt-report-filter= "filename,ln1-ln2"`	ファイル filename の行 ln1 - ln2 の範囲のみレポートを出力します。
`-qopt-report-phase=phase1[,phase2,…]`	`/Qopt-report-phase:phase1[,phase2,…]`	特定の最適化フェーズの情報のみ出力します。

1a

最適化フェーズ	説明
vec	SIMD 命令による自動ベクトル化
par	コンパイラーによる自動並列化
loop	メモリー、キャッシュ使用率、ループの最適化
openmp	OpenMP* 宣言子によるスレッド化
ipo	インライン展開を含むプロシージャー間の最適化
pgo	プロファイルに基づく最適化 (ランタイム・フィードバックを使用)
cg	コード生成時の最適化
offload	インテル® Xeon Phi™ コプロセッサーへのオフロードとデータ転送
all	すべての最適化フェーズ (デフォルト)

1b

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

レポートの出力

レポートはデフォルトでは無効になっています。**-qopt-report** (Linux* および OS X*) または **/Qopt-report** (Windows*) オプションで有効にできます。並列ビルドに対応するため、デフォルトでは、各ソースファイルごとに個別のレポート (.optrpt ファイル) がソースファイルと同じディレクトリーに生成されます。**-qopt-report-file** (Linux* および OS X*) または **/Qopt-report-file** (Windows*) オプションを使用して、異なる名前や stderr、stdout にレポートを出力することもできます。

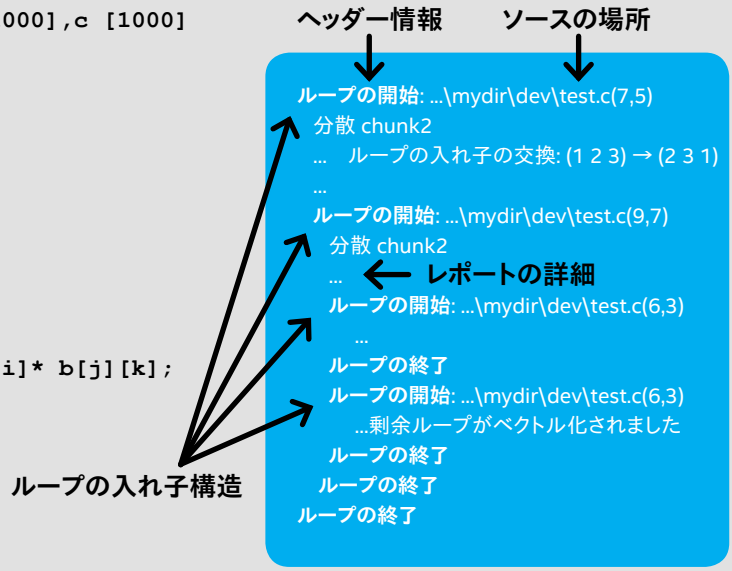
-g (Linux* および OS X*) または **/Zi** (Windows*) オプションが指定されているデバッグビルドでは、ループの最適化に関する情報がアセンブリー・コードとオブジェクト・ファイルに埋め込まれることがあります。これにより、アセンブリー・コードのループ構造が理解しやすくなり、コンパイラーから提供される最適化情報をほかのソフトウェア・ツールで利用できるようになります。

最適化レポートは非常に大きくなることがあります。**-qopt-report-routine** (Linux* および OS X*) または **/Qopt-report-routine** (Windows*) オプションにより特定の関数や、**-qopt-report-filter** (Linux* および OS X*) または **/Qopt-report-filter** (Windows*) によりソースファイルの行番号の範囲を制限することができます。

ループ関連のレポートのレイアウト

入れ子構造のループの最適化に関するメッセージは、**図 2** のように階層表示されます。各ループに対し、"ループの開始" メッセージ、ソースコードの行と列番号、"ループの終了" メッセージが生成されます。入れ子構造が一目で分かるようにインデントが挿入されています。コンパイラーが複数バージョンのループを生成する場合、入れ子構造がソースコードとは異なることがあり、ループは 2 つ以上のサブループに "分散" (分割) されることもあります。**図 2** の部分的なレポートから、行 6 の外部ループが最適化されたコードでは 2 つの内部ループに変換されていることが分かります。

```
1 double a[1000][1000],b[1000][1000],c [1000]
[1000];
2
3 void foo() {
4 int i,j,k;
5
6 for( i=0; i<1000; i++) {
7  for( j=0; j< 1000; j++) {
8    c[j][i] = 0.0;
9    for( k=0; k<1000; k++) {
10     c[j][i] = c[j][i] + a[k][i]* b[j][k];
11    }
12   }
13  }
14 }
```



2

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

階層表示により、コンパイラーによる最適化をコードの特定のループに直接関連付けることができます。

ベクトル化されたループ中の SIMD ロード命令は、ロードするデータがメモリーアドレス (SIMD レジスターの幅の倍数) でアライメントされている場合、最も効率良く実行できます。そのため、コンパイラーは最初のいくつかの反復を " ピール "[1] し、ベクトル化されたループがアライメントされたデータを処理できるようにすることがあります。ベクトル化されたカーネルの後にあるいくつかの反復は、"剰余" ループとして別途最適化されます。**図 3** は、最適化レポートでピールループと剰余ループがどのように出力されるかを示したものです。

```
ループの開始: ggF.cc(124,5) インライン展開後: ggF.cc(56,7)
   リマーク #15018: ループはベクトル化されませんでした: 内部ループではありません。
   ループの開始: ggF.cc(138,5) インライン展開後: ggF.cc(60,15)
      ピール
         リマーク #25460: ループはベクトル化されませんでした
   ループの終了
   ループの開始: ggF.cc(138,5) インライン展開後: ggF.cc(60,15)
         リマーク #15145: ベクトル化のサポート: アンロールファクターが 4 に設定されています。
         リマーク #15002: 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: ggF.cc(138,5) インライン展開後: ggF.cc(60,15)
      剰余
         リマーク #15003: 剰余ループがベクトル化されました
   ループの終了
ループの終了
```

ピールと剰余による
ベクトル化

3

[1]: ピールとは、果物の皮をむくことにたとえて、アライメントされていないデータにアクセスするループ反復をループから分離することを意味します。

ループおよびベクトル化レポートの活用

新しい最適化レポートは、コンパイラーによって適用された最適化だけでなく、コンパイラーの最適化を妨げるコーディングを理解するのに役立ちます。**図 4** の単純な C コード例とそのビルド結果について考えてみましょう (レポートとその解釈方法は C++ と Fortran でほぼ同じです)。関数 **`foo()`** は、配列 **`theta`** をロードし、算術関数を含む計算を行い、結果配列 **`sth`** に格納しています。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

```
#include <math.h>
void foo (float * theta, float * sth)  {
  int i;
  for (i = 0; i < 128; i++)
       sth[i] = sin(theta[i]+3.1415927);
}

$ icc -c -qopt-report=2 -qopt-report-phase=loop,vec -qopt-report-file=stderr foo.c

最適化レポート開始: foo(float *, float *)

    レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

ループの開始: foo.c(4,3)
<マルチバージョン v1>
    リマーク #25228: データの依存関係のループをマルチバージョンにしました。
    リマーク #15399: ベクトル化のサポート: アンロールファクターが 2 に設定されています。
    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: foo.c(4,3)
<マルチバージョン v2>
    リマーク #15304: ループはベクトル化されませんでした: マルチバージョンのベクトル化できないループ・インス
    タンスです。
ループの終了
```

4

このレポートは、ソースコードの 1 つのループに対し、データ依存性を考慮してコンパイラーが 2 つのバージョン (マルチバージョン) を生成したことを示しています。コンパイル時に、コンパイラーはポインター引数 **theta** と **sth** がエイリアスされるかどうか分かりません。つまり、2 つのポインターがオーバーラップして安全にベクトル化できない可能性があります。そのため、ベクトル化されたバージョンとされていないバージョンを生成し、データのオーバーラップをチェックするランタイムテストを追加して、安全な場合はベクトル化されたループが実行されるようにし、安全でない場合はベクトル化されていないループが実行されるようにします。

2 つのポインター引数がエイリアスされないことが分かっている場合、**-fargument-noalias** (Linux* および OS X*) または **/Qalias-args-** (Windows*) オプションか、**restrict** キーワードとともに -restrict (Linux* および OS X*) または /Qrestrict (Windows*) オプションを指定して、コンパイラーに知らせることができます。別の方法として、**#pragma ivdep** または **#pragma omp simd** により、ループを安全にベクトル化できることをコンパイラーに直接知らせることもできます (**#pragma omp simd** を使用する場合は、-qopenmp (Linux* および OS X*) または /Qopenmp (Windows*) オプションか、-qopenmp-simd (Linux* および OS X*) または /Qopenmp-simd (Windows*) オプションを指定する必要があります)。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

どのアプローチを使用する場合も、ベクトル化されたバージョンのみが生成され、コンパイラーはデータのオーバーラップをチェックするランタイムテストを生成する必要がありません。**図 5** は、**図 4** のコード例を、コマンドライン・オプションによりレポートの詳細レベルを上げてビルドした結果です。

```
$ icc -c -fargument-noalias  -qopt-report=4 -qopt-report-phase=loop,vec
-qopt-report-file=stderr foo.c

最適化レポート開始: foo(float *, float *)

    レポート: ループの入れ子、ベクトルの最適化 [loop, vec]

ループの開始: foo.c(4,3)
   リマーク #15389: ベクトル化のサポート: 参照 theta にアラインされていないアクセスが含まれています。
   [ foo.c(5,14) ]
   リマーク #15389: ベクトル化のサポート: 参照 sth にアラインされていないアクセスが含まれています。
   [ foo.c(5,5) ]
   リマーク #15381: ベクトル化のサポート: ループ本体内でアラインされていないアクセスが使用されました。
   [ foo.c(5,5) ]
   リマーク #15399: ベクトル化のサポート: アンロールファクターが 2 に設定されています。
   リマーク #15417: ベクトル化のサポート: 浮動小数点数をアップコンバートします (単精度から倍精度 1)。
   [ foo.c(5,14) ]
   リマーク #15418: ベクトル化のサポート: 浮動小数点数をダウンコンバートします (倍精度から単精度 1)。
   [ foo.c(5,5) ]
   リマーク #15300: ループがベクトル化されました
   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
   リマーク #15451: マスクなし非アライン・ユニット・ストライド・ストア: 1
   リマーク #15475: --- ベクトルループのコストサマリー開始 ---
   リマーク #15476: スカラーループのコスト: 114
   リマーク #15477: ベクトルループのコスト: 40.750
   リマーク #15478: スピードアップの期待値: 2.790
   リマーク #15479: 軽いベクトル操作: 9
   リマーク #15480: 中オーバーヘッドのベクトル操作: 1
   リマーク #15481: 高オーバーヘッドのベクトル操作: 1
   リマーク #15482: ベクトル化された算術ライブラリーの呼び出し: 1
   リマーク #15487: 型変換: 2
   リマーク #15488: --- ベクトルループのコストサマリー終了 ---
   リマーク #25015: ループの最大トリップカウントの予測=64
ループの終了
```

5

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

レポートから、ループに対し 1 つのバージョンのみ生成されていることが分かります。コストサマリーは、ベクトル化によるスピードアップの期待値が約 2.79 であることを示しています。これは悪くありませんが、さらにスピードアップできます。リマーク 15417 と 15418 は、行 5 の列 14 と列 5 の単精度と倍精度の変換に関するもので、サマリーから 2 度の変換が行われたことが分かります。ソースコードを確認すると、配列 **theta** は単精度ですが、リテラル定数 3.1415927 はデフォルトで倍精度になるため、これらを足した結果は倍精度になります。つまり、sin 関数の倍精度バージョンが呼び出されますが、**sth** へ格納するため、結果は単精度に変換されます。

これはパフォーマンスに 2 つの影響を及ぼします。1 つ目は、精度が高いほど sin 関数の計算時間が長くなります。2 つ目は、倍精度は SIMD レジスターで単精度の 2 倍のベクトル幅を必要とするため、ベクトル命令で一度に操作可能な要素数が半減します。ソースコードで、リテラル定数と sin 関数を明示的に単精度へ変更してみましょう。

```
sth[i] = sinf(theta[i]+3.1415927f);
```

すると、精度の変換に関するメッセージが出力されなくなり、スピードアップの期待値はほぼ倍の 5.4 になります。これは、実行時間のほとんどがベクトル算術ライブラリーの呼び出し (リマーク 15482) に費やされ、オーバーヘッドが小さいベクトル操作 (リマーク 15479) にはわずかな時間しかかからないためです。

次に、ベクトル化されたループの最大トリップカウントの予測が、256 から 64 (リマーク 25015) に減っていることが分かります。つまり、各ベクトル操作は 4 つの float 値 (16 バイト) を処理します。これは、デフォルトで、ベクトル幅が 16 バイトのインテル® ストリーミング SIMD 拡張命令 (インテル® SSE) 向けにコンパイルされるためです。ベクトル幅が 32 バイトのインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応インテル® プロセッサー向けにコンパイルする場合は、**-xavx** (Linux* および OS X*) または **/QxAVX** (Windows*) コンパイラー・オプションを指定します。その場合、レポートの出力内容は次のようになります。

```
リマーク #15477: ベクトルループのコスト : 11.620
リマーク #15478: スピードアップの期待値 : 9.440
...
リマーク #25015: ループの最大トリップカウントの予測 =32
```

インテル® Xeon Phi™ コプロセッサーをターゲットにした場合、最大トリップカウントは 16 になり、ベクトル幅は 64 バイト (16 の float 値) となります。

次に、アライメントに関するメッセージについて見てみましょう。通常、インテル® AVX 向けに 32 バイト境界 (インテル® SSE 向けの場合は 16 バイト境界、インテル® Xeon Phi™ コプロセッサー向けの場合は 64 バイト境界) でアライメントされているメモリーのほうが、アライメントされていないメモリーよりも効率良くアクセスできます。リマーク 15381 は、ループ内でアライメントされていないメモリーアクセスが検出された際に出力される一般的な警告です。リマーク 15389、15450、15451 から、**theta** のロード操作と **sth** へのストア操作で、コンパイラーはデータがアライメントされていないことを仮定しているのが分かります。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

theta と **sth** はポインター引数として渡されるため、それらがアライメントされているかどうかコンパイラーには判断できません。**__declspec(align(32))** (Windows*) または **__attribute__((align(32)))** (Linux* および OS X*) を指定して宣言するか、**_mm_malloc()** または Posix* の **memalign()** により割り当てることで、データをアライメントすることができます。関数 **foo()** に渡されるポインター引数がアライメントされていることが明確な場合は、キーワード **__assume_aligned()** を使用してコンパイラーに知らせることができます。

```
__assume_aligned(theta,32);
__assume_aligned(sth,32);
```

これらのキーワードは、関数のポインター引数がアライメントされたデータを指すことが確実であると分かっている場合にのみ使用すべきです。実行時にアライメントのチェックは行われません。**__assume_aligned** キーワードを指定して再コンパイルすると、次のようなアライメントされたメモリーアクセスに関するメッセージのみが出力されます。

```
リマーク #15388: ベクトル化のサポート: 参照 theta にアラインされたアクセスが含まれています。
```

このベクトル化により、スピードアップの期待値が約 20% 向上します。

```
リマーク #15477: ベクトルループのコスト: 9.870
リマーク #15478: スピードアップの期待値: 11.130
```

sth がアライメントされたため、ストアされるデータが近い将来 (つまり、キャッシュから追い出される前に) 再びアクセスされない場合、コンパイラーはメモリーへ直接ストアするストリーミング・ストア (一時的でないストア) を生成することができます。これにより、キャッシュをフェッチしてから書き込む "read for ownership" を回避できるようになり、データの読み書きを頻繁に行い、メモリー帯域幅によりパフォーマンスが制限されるアプリケーションでは利点が得られる可能性があります。また、キャッシュをほかのことに利用することができます。コンパイラーは、データが非常に大きい場合のみ (通常は数メガバイトを超える場合)、自動的にストリーミング・ストアを生成します。トリップカウントを 2000000 に増やすか、ループの前に **#pragma vector nontemporal** を追加すると、コンパイラーによりストリーミング・ストア命令が生成され、最適化レポートに次のメッセージが追加されます。

```
リマーク #15467: マスクなしアライン・ストリーミング・ストア: 1
リマーク #15412: ベクトル化のサポート: sth のストリーミング・ストアが生成されました。
```

このように小さな関数であっても、最適化レポートは役立つ情報をもたらします。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

インライン展開の IPO レポート例

IPO レポートは、複数の関数に関連する最適化情報を提供します。ここでは、インライン展開に注目してみましょう。

```
 3    static void bar (float a[N], float b[N]) {
        …     //    大きな本体
11    }
12
13    static void foo(float a[N], float b[N])  {
        …     //    小さな本体
21    bar(a, b);
22    }
23
24    extern int main() {
26    float a[N];
27    float b[N];
        …
35    foo(a, b);
36    foo(a, b);
37    printf("result %d %d\n",b[0], b[N-1]]);
38    }
```

```
icc -qopt-report=3 -qopt-report-phase=ipo sm.c
 インライン展開オプション値:
  -inline-factor: 100
 ...
 インライン展開レポート: (main) [1]  sm.c(24,19)
   -> インライン展開: [35] foo()
     -> [21] bar()
  - > インライン展開: [36] foo()
     -> [21] bar()
   - >外部関数: [37] printf

 インライン展開レポート: (bar) [2] sm.c(3,42)

 実行されないスタティック関数: (foo) sm.c(13,42)
```

6

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

図 6 は、小さなスタティック関数 **foo()** を 2 回呼び出し、その後 **printf** を呼び出して結果を出力するメインプログラムです。**foo()** は、大きなスタティック関数 **bar()** を呼び出します。各関数ごとにインライン展開レポートが生成されます。行 24、列 19 から始まる **main()** では、行 35 と 36 の **foo()** がインライン展開され、**foo()** では、行 21 の **bar()** がインライン展開されます。**main()** は行 37 で **printf()** を呼び出します。コンパイラーは **printf** のソースコードにアクセスできないため、これは外部関数と見なされます。行 3、列 42 から始まる **bar()** には関数呼び出しが含まれません。行 13、列 42 から始まるスタティック関数 **foo()** の本体は、この関数へのすべての呼び出しがインライン展開されるため、実行されないものと見なされます。この関数は外部から呼び出されることがないため、コンパイラーはこの関数のスタンドアロン・バージョンを生成する必要がありません。

レポートレベル 3 では、間接関数呼び出しも出力され、"INDIRECT" とマークされます。より高いレポートレベルでは、コンパイラーがアクセスできるすべての呼び出された関数のサイズと、呼び出す関数をインライン展開した場合のサイズの増加が表示されます。

最適化レポートのインライン展開フェーズの最初に、適用されたインライン展開パラメーター値とそれを変更するためのコンパイラー・オプションのリストが表示されます。レポートの情報を基に、これらのオプションを使用してインライン展開の量を制御できます。例えば、**-inline-factor** (Windows* の場合は **/Qinline-factor**) を 100 から 200 に変更すると、インライン展開の対象を制御するすべてのサイズ制限が 2 倍になります。**inline**、**noinline** や **forceinline** のようなプラグマを使用したり、関数の属性で **__attribute__** や **__declspec** キーワードを指定して、個々の関数のインライン展開を制御することができます。詳細は、『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。

その他のレポートフェーズ

-qopt-report-phase=par: コンパイラーによる自動並列化 (スレッド化) に関するメッセージを出力します。ベクトル化およびループレポートと同様の構造で、これらのレポートと統合されます。

-qopt-report-phase=openmp: OpenMP* プラグマと宣言子のスレッド構造に関するメッセージを出力します。

-qopt-report-phase=pgo: プロファイル関数を含むプロファイルに基づく最適化 (PGO) に関するメッセージを出力します。

-qopt-report-phase=cg: 組込み関数の変換 (下位レベルの構造への変換) など、コード生成時の最適化に関するメッセージを出力します。

-qopt-report-phase=loop: キャッシュ・ブロッキング、プリフェッチ、ループ交換、ループ融合など、ループとメモリーの最適化に関するメッセージを出力します。

-qopt-report-phase=offload: インテル® Xeon Phi™ コプロセッサーとの間で転送されるデータのサマリーを出力します。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

まとめ

インテル® C/C++ および Fortran コンパイラー 15.0 の新しい最適化レポートは、豊富な情報を分かりやすい形式で提供します。レポートには、適用された最適化とされなかった最適化の情報が含まれます。これらのレポートから、アプリケーションのパフォーマンスをさらに向上するためのチューニングに役立つヒントを得ることができます。

詳細は、インテル® Parallel Studio XE 2015 Composer Edition の**「インテル® C++ コンパイラー・ユーザー・リファレンス・ガイド」**または**「インテル® Fortran 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。

* 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。

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。

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

インテル® コンパイラーを試そう

次のソフトウェアで利用可能:
インテル® Parallel Studio XE 2015 Composer/Professional/Cluster Edition >

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

マルチコアからメニーコアまで スケーラブルなパフォーマンスの設計

Ekaterina Antakova インテル コーポレーション開発製品部門ソフトウェア・エンジニア

アプリケーションをスレッド化したのにパフォーマンスがあまり向上しなかった経験はありませんか？ あるいは、コア数を増やしてもパフォーマンスが向上しない "スケーラビリティーの限界" に達したことはありませんか？ 並列アルゴリズムの実装には多大な労力が必要です。実装を始める前に、いくつかの候補を比較して最適なものを選ぶことができたら良いと思いませんか？ インテル® Advisor XE 2015 はそのためのソリューションです。設計をモデル化し、パフォーマンス・スケーリングと同期の問題を予測するフレームワークを提供します。ここでは、インテル® Advisor XE 2015 のモデル化をインテル® Xeon Phi™ コプロセッサーに拡張する方法を紹介します。

インテル® Parallel Studio XE 2015 の Cluster Edition と Professional Edition に含まれるインテル® Advisor XE 2015 は、シリアルコードから並列化の可能性を見つけ、インテル® Xeon Phi™ コプロセッサーにより利点が得られる個所をピンポイントで特定し、スケーリングの妨げとなる主な要因を識別します。インテル® Advisor XE は、http://www.isus.jp/article/intel-software-dev-products/intel-advisor-xe/ から利用できます。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

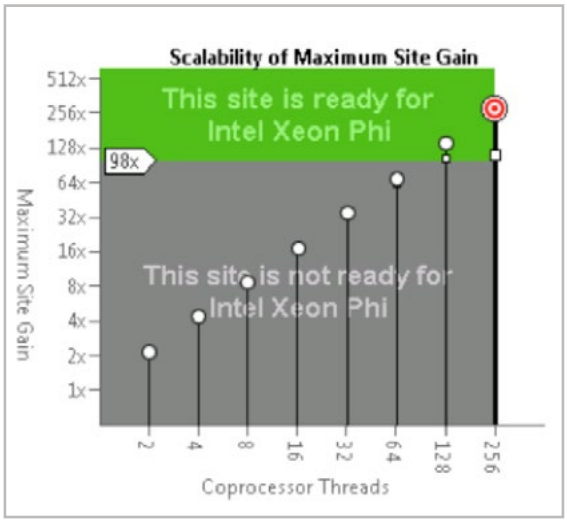
新しい CPU とコプロセッサーへ効率良く移行できるようにし、 プログラマーの負担を軽減するため、 インテルは並列プログラミングによりパフォーマンスを向上できる最適な場所の特定を支援します。 例えば、 インテル® Xeon Phi™ コプロセッサーを利用して、 高度な並列ワークロードを効率良く処理できる場所を特定します。 それには、 アプリケーションにおいて "高度に並列" が何を意味するのかを理解する必要があります。インテル® Advisor XE は、さまざまな新機能によりその手助けをします。

ワークロードに対するインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーでのピーク・パフォーマンスを自動的に比較する機能は、 どのコード領域をコプロセッサーで実行すべきか適切な判断を下すのに役立ちます。 大きなデータセットの並列スケーラビリティーを予測する機能は、 最近のマルチコア / メニーコア・ハードウェアで並列効率を向上する方法を理解するのに役立ちます。

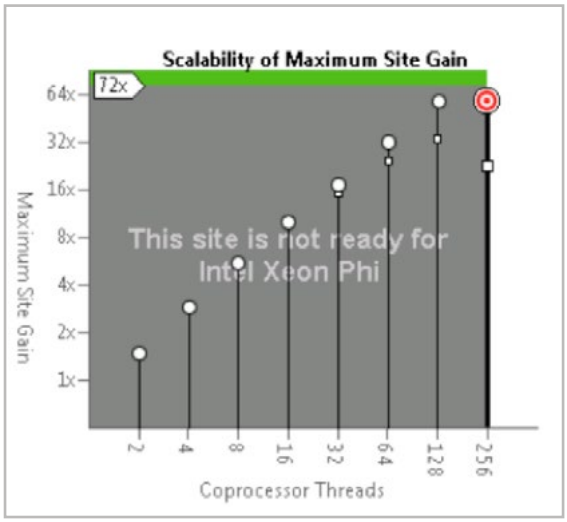
モデル化機能は、アプリケーションに関する次の 3 つの質問に答えます。

1. **ワークロードに対しインテル® Xeon Phi™ コプロセッサーが適しているかどうか？**
   インテル® Advisor XE の適合性解析は、 ワークロードに対し、 インテル® Xeon Phi™ コプロセッサーのパフォーマンスがインテル® Xeon® プロセッサーを超えるかどうか自動的に予測します。 また、 開発サイクルの早い段階で、 現在のアプリケーション構造がコプロセッサーでスケーリングするかどうか判断することができます。

   図 1 と **2** は、 イメージ処理アルゴリズムの異なるステージを実行する 2 つのテストループの並列バージョンで得られるパフォーマンス・ゲイン予測です。 緑色の領域は、 インテル® Xeon Phi™ コプロセッサーでこのワークロードを実行することで得られるゲインを示します。 2 つの図の緑色の領域を比較すると、**図 1** のほうが良くスケーリングし、 コプロセッサーで実行するのに適しているように見受けられます。 この予測によると、 コプロセッサーに移植した場合、 128 以上のスレッドでスピードアップが得られるはずです。**図 2** のループは、 128 以上のコプロセッサー・スレッドでゲインが伸びなくなるため、 "高度に並列" とは言えません。 つまり、 より高度に並列なアルゴリズムに変更しない限り、 コプロセッサーでの実行には適していません。

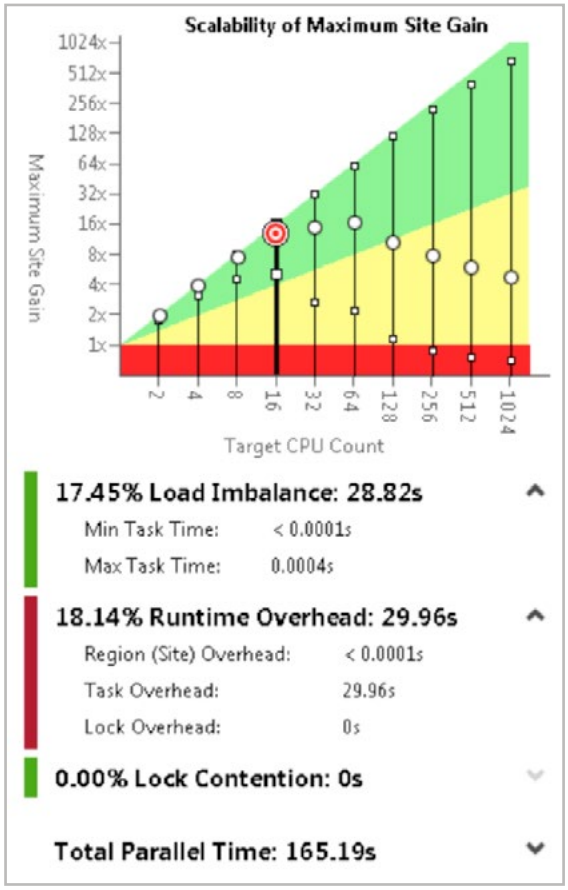


1



2

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。



3

2. **アプリケーションで、並列パフォーマンスとスケーラビリティーを妨げている主な要因は?**
シリアルコードを効率良く並列化する場合、期待するパフォーマンスとスケーラビリティーの妨げとなる主な要因を理解することが不可欠です。インテル® Advisor XE の適合性解析は、並列ジョブのインバランス、ロック競合、並列ランタイム・オーバーヘッドによる並列パフォーマンスの問題に関する情報を提供します。

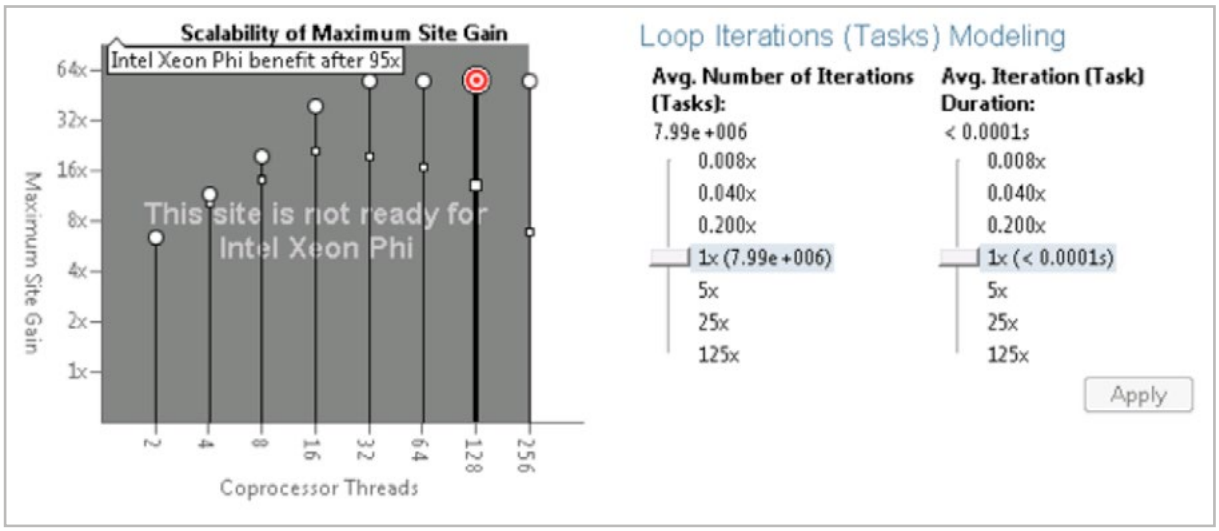
図 3 は、並列化によりパフォーマンスが向上する可能性のあるループを示しています。3 つのマーカー (上から□、○、□) は、ロックによるオーバーヘッド、タスクのチャンク、並列フレームワークなどを変更することで得られるパフォーマンスの範囲を表しています。真ん中の○は、選択したランタイム・モデル・オプションに応じて移動します。3 つのマーカーの範囲も選択したタスク・モデル・オプションに応じて変わります。**図 3** では、予測される最大パフォーマンスは CPU の増加に伴い大きくなります (上の□で示されています)。しかし、このループの現在のパフォーマンス・ゲイン (○で示されています) は、64 CPU 以上ではスケーリングしないとインテル® Advisor XE は予測しています。この原因の 1 つは、並列ランタイムのオーバーヘッドが大きいことです。合計並列実行時間が約 165 秒であるのに対し、並列実行のオーバーヘッドは合計約 30 秒で、並列タスク間に大きなインバランスがあり、約 29 秒の待機時間が発生しています。これは、並列タスクの粒度とループ分割に問題があることを示しています。ランタイム・オーバーヘッドが大きいことから、ワークロードに対して並列タスクが小さすぎ、タスクの生成が頻繁に行われすぎていることが分かります。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

3. **ワークロード・サイズの増加が並列スケーラビリティーに及ぼす影響は？**
反復空間のモデル化は、ワークロード・サイズの増加の影響を予測します。ループの反復を増やした場合、反復の実行時間を長くした場合、あるいはその両方を同時に行った場合をモデル化します。そして、小さなサンプルを実行して、データセット・サイズと計算量が増えた場合のパフォーマンスの変化を解析し、マルチコア CPU とコプロセッサーを最大限に利用するため、適切なデータセット・サイズを特定します。

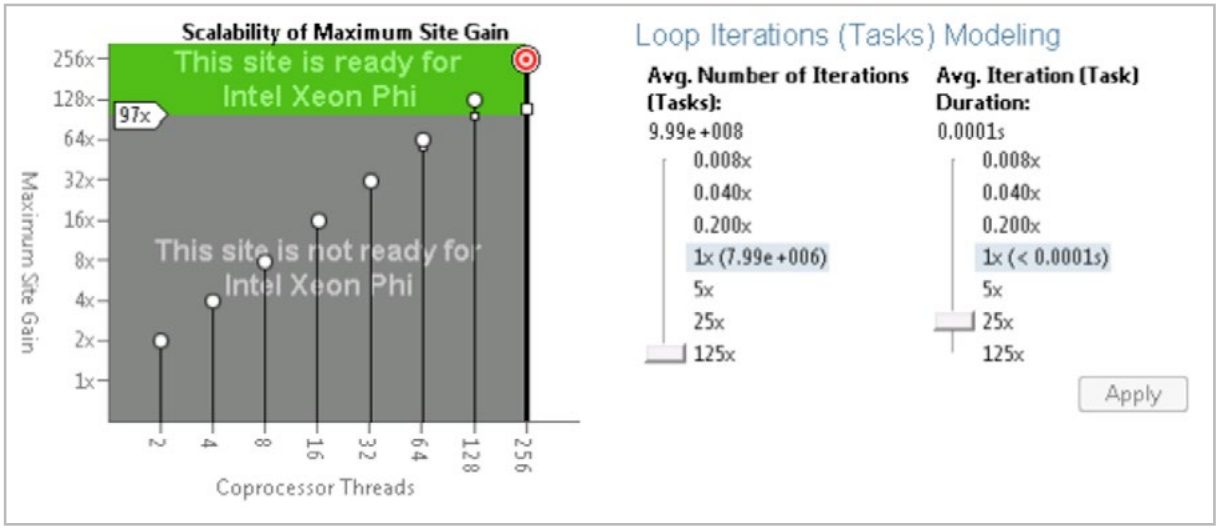
図 4 は、小さなテスト・ワークロードでの並列ループモデルのスケーラビリティー予測と、大きなデータサイズの処理をモデル化するためのユーザー・インターフェイスです。現在の予測では、このワークロードは 32 以上の CPU でスケーリングしません。

入力データを大きくするとスケーラビリティーは向上するでしょうか? 解析したループの特性と処理するデータサイズを考慮して、反復回数は約 125 倍に増加し、各反復の実行時間は約 25 倍になると仮定します。Loop Iterations (Tasks) Modeling で、Avg. Number of Iterations (Tasks) と Avg. Iteration (Task) Duration のバーを操作して設定します。



4

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。



5

図 5 は、新しいデータサイズとパラメーターを適用したこのループモデルのパフォーマンス予測です。 このループは、入力データが大きくなると (反復回数が増え、 各反復の実行時間が長くなると)、 スケーラビリティーが向上し、 インテル® Xeon Phi™ コプロセッサーでの実行に適した候補となります。

インテル® Advisor XE の新しいモデル化機能は、並列フレームワークの CPU 負荷の高い作業、タスクの粒度 (チャンクサイズとスケジュール)、ロードバランス、ロック競合、オーバーヘッドを測定し、その結果に応じてモデルを生成します。予測パフォーマンスとゲインを解析し、インテル® Xeon Phi™ コプロセッサーでの実行に適しているかどうか判断するため、モデルには CPU 使用率に関するパラメーター、コプロセッサー固有のランタイム・オーバーヘッド、データサイズのデータ転送オーバーヘッド (オフロード実行モデルの場合) が含まれます。

新しいワークロード・スケーリング機能について、初期ベータ版の評価を行ったグラフィックス処理を行う開発者は、「ベータ版インテル® Advisor XE 2015 は、大きなイメージを複数のレンダリングに分割するポリシーを選択する際にデータサイズの予測をするのに非常に役立ちました。」と述べています。

インテル® Advisor XE を利用することで、さまざまな CPU 数とインテル® Xeon Phi™ コプロセッサーで並列コードのスケーラビリティーをモデル化し、アプリケーション中の最もパフォーマンスが向上する可能性のある領域を特定して、そのコード領域に開発労力を集中することができます。より大きなワークロードで試してみることは、ターゲット・プラットフォームで多くのコアを最大限に利用するのに最適なワークロード・サイズを理解するのに役立ちます。異なる条件下でアプリケーションのワークロードがどのようにスケールするのか検証し、コプロセッサーで実行することでパフォーマンスが向上するかどうか調べることができます。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

インテル® Advisor XE によるインテル® Xeon Phi™ コプロセッサーでのパフォーマンスのスケーラビリティー予測

インテル® Advisor XE のモデル化機能は、アプリケーションのアルゴリズムのスケーラビリティーを理解し、プログラムに実装を行った場合の現実的なスケーラビリティーを把握し、コード内のスケーリングの妨げとなる要因を特定するのに非常に役立ちます。残念ながら、同じ処理を実行しつつスケーラビリティーが向上する、全く異なるアプローチやアルゴリズムを提供できるツールはありません。しかし、インテル® Advisor XE を利用することで、さらなるスケーラビリティーを追求すべきかどうか判断し、さまざまなアプローチを簡単に素早く評価することができます。これにより、スケーリングに最適な手法を見つけ、インテル® Xeon Phi™ コプロセッサーのような高度な並列システムでスケーリングするアプリケーションを特定できます。

インテル® Advisor XE を試そう

次のソフトウェアで利用可能:
インテル® Parallel Studio XE 2015 Professional/Cluster Edition >

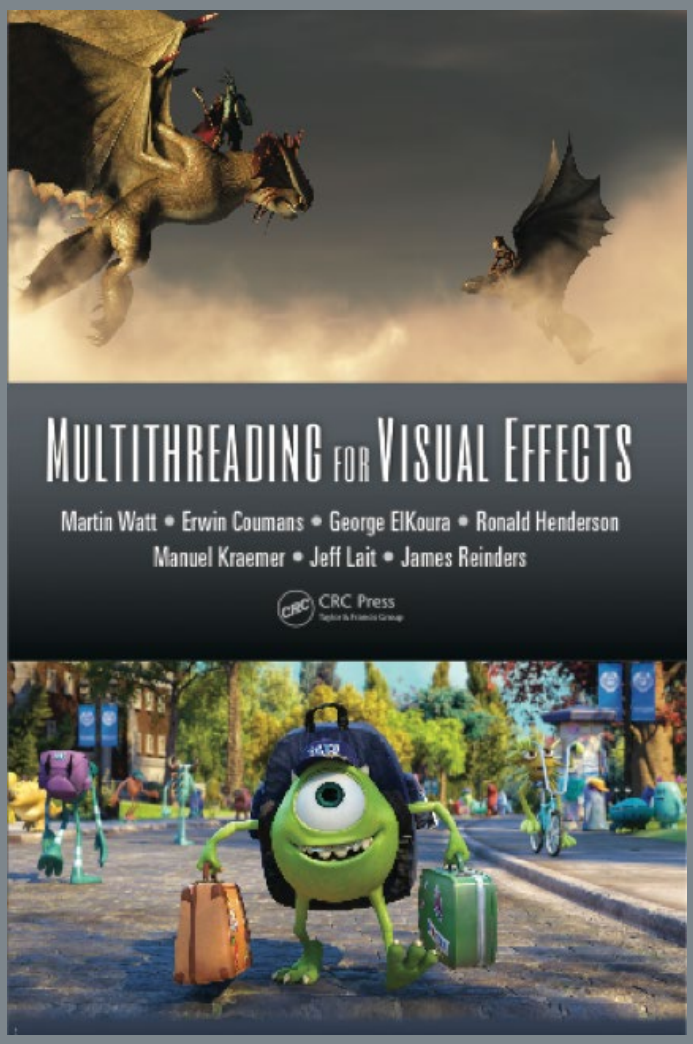
インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

BLOG HIGHLIGHTS

新しい書籍:『Multithreading for Visual Effects』

JAMES REINDERS »

DreamWorks Animation、Pixar、Side Effects、AMD、Intel から集まった著者達による、SIGGRAPH 2013 コース「Multithreading in Visual Effects」をベースにした書籍です。コースで使用された資料を大幅に拡張 / 更新し、Pixar の Manuel Kraemer 氏により OpenSubdiv に関する章が追加されています。Ron Henderson 氏は、FLUX ガス・シミュレーション・システム (第 5 章) の開発が評価され、2014 年 2 月に Technical Achievement Award を受賞しました。



各章と著者のリスト

- Multithreading Introduction and Overview
  James Reinders, Intel Corporation
- Houdini: Multithreading Existing Software
  Jeff Lait, Side Effects Software, Inc.
- The Presto Execution System: Designing for Multithreading
  George ElKoura, Pixar Animation Studios
- LibEE: Parallel Evaluation of Character Rigs
  Martin Watt, DreamWorks Animation
- Fluids: Simulation on the CPU
  Ron Henderson, DreamWorks Animation
- Bullet Physics: Simulation with OpenCL™
  Erwin Coumans, Advanced Micro Devices, Inc.
- OpenSubdiv: Interoperating GPU Compute and Drawing
  Manuel Kraemer, Pixar

この書籍は、日本語翻訳版の出版が予定されています。

この記事の続きはこちら (英語) でご覧になれます。 ›

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

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) に追加された命令

James Reinders インテル コーポレーションの並列プログラミング・エバンジェリスト兼ディレクター

『Intel® Architecture Instruction Set Extensions Programming Reference (インテル® アーキテクチャー命令セット拡張プログラミング・リファレンス)』に、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) に追加された命令の定義が加わりました。

2013 年にインテル® AVX-512 に関する最初のブログ (英語) で述べたとおり、インテル® AVX-512 は今後出荷予定のインテル® Xeon Phi™ プロセッサー / コプロセッサー (開発コード名 Knights Landing) で初めて実装されます。

インテルでは、Knights Landing 以降に出荷予定のいくつかのインテル® Xeon® プロセッサーでもインテル® AVX-512 をサポートするべく取り組んでいます。対象プロセッサーでは、すでに発表されているインテル® AVX-512 命令とともに新たに追加された命令を利用できます。

新しい命令は、2 つのグループに分けることができ、インテル® AVX-512 の一部の命令を強化します。1 つ目は、バイトおよびワード命令と呼ばれる、整数演算を強化するバイトとワード (8 ビットと 16 ビット) 操作で、AVX512BW CPUID フラグが含まれます。これらの命令は、マスクレジスターの 64 ビットすべてを利用します。2 つ目は、ダブルワードおよびクワッドワード命令と呼ばれる、整数および浮動小数点演算を強化するダブルワードとクワッドワード (32 ビットと 64 ビット) 操作で、AVX512DQ CPUID フラグが含まれます。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

前述の 2 つとは独立した追加機能として、ベクトル長の拡張があります。これにより、ほとんどのインテル® AVX-512 命令は、512 ビットだけでなく 128 ビットまたは 256 ビットの操作が可能となります。現在では、ほとんどの基本命令、競合検出命令に加えて、新しいバイト、ワード、ダブルワード、クワッドワード命令にも適用できます。インテル® AVX-512 のベクトル長の拡張には、AVX512VL CPUID フラグが含まれます。ベクトル長の拡張により、ほとんどのインテル® AVX-512 操作で XMM (128 ビット、インテル® SSE 用) レジスターと YMM (256 ビット、インテル® AVX 用) レジスターも利用することができます。また、マスクレジスターの使用、レジスター 16..31 へのアクセスを含む EVEX エンコーディング機能を ZMM レジスターだけでなく XMM および YMM レジスターに適用できます。

製品化前のテスト用エミュレーター

新しいインテル® AVX-512 命令の早期開発を支援する、インテル® Software Development Emulator を http://www.intel.com/software/sde (英語) から入手できます。

インテル® AVX-512 命令ファミリー

インテル® AVX-512 基本命令は、すべてのインテル® AVX-512 実装に含まれます。インテル® AVX-512 競合検出命令はオプションの拡張とされていましたが、コンパイラーによるベクトル化で良い結果をもたらすことが実証されたため、インテル® AVX-512 対応のインテル® Xeon® プロセッサーに含まれる予定です。つまり、基本命令と競合命令はどちらも、将来のインテル® Xeon Phi™ コプロセッサー / プロセッサーと将来のインテル® Xeon® プロセッサーのすべてのインテル® AVX-512 実装に含まれます。

Knights Landing (開発コード名) は、インテル® AVX-512 の指数、逆数、プリフェッチ命令をサポートし、インテル® AVX-512 対応の最初のインテル® Xeon® プロセッサーは、インテル® AVX-512 のバイト、ワード、ダブルワード、クワッドワード命令とベクトル長の拡張をサポートします。将来のインテル® Xeon Phi™ コプロセッサー / プロセッサー (開発コード名 Knights Landing 以降) では、追加のインテル® AVX-512 命令が提供される可能性がありますが、少なくとも Knights Landing と同等のサポート (基本命令、競合検出命令、指数および逆数命令、プリフェッチ命令) が保持されるでしょう。同様に、インテル® Xeon® プロセッサー・ファミリーでもインテル® AVX-512 命令のサポートには、少なくとも基本命令、競合検出命令、バイトおよびワード命令、ダブルワードおよびクワッドワード命令、ベクトル長の拡張が含まれるでしょう。それぞれのプロセッサー・ファミリーでこれらを前提として、コンパイラー設計と対応作業を単純化できます。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

インテル® AVX-512 サポート

追加されるインテル® AVX-512 命令の詳細情報が明らかになったことで、製品が出荷されるまでにツール、アプリケーション、オペレーティング・システムでこれらの命令に対応することが可能になりました。インテルは、オープンソース・プロジェクト、アプリケーション・プロバイダー、ツールベンダーと協力して、インテル® AVX-512 への対応に取り組んでいます。インテルのコンパイラー、ライブラリー、解析ツールは、すでにインテル® AVX-512 に対応しています。追加の命令についても、2014 年 11 月のアップデートで対応しています。

インテル® AVX-512 のドキュメント

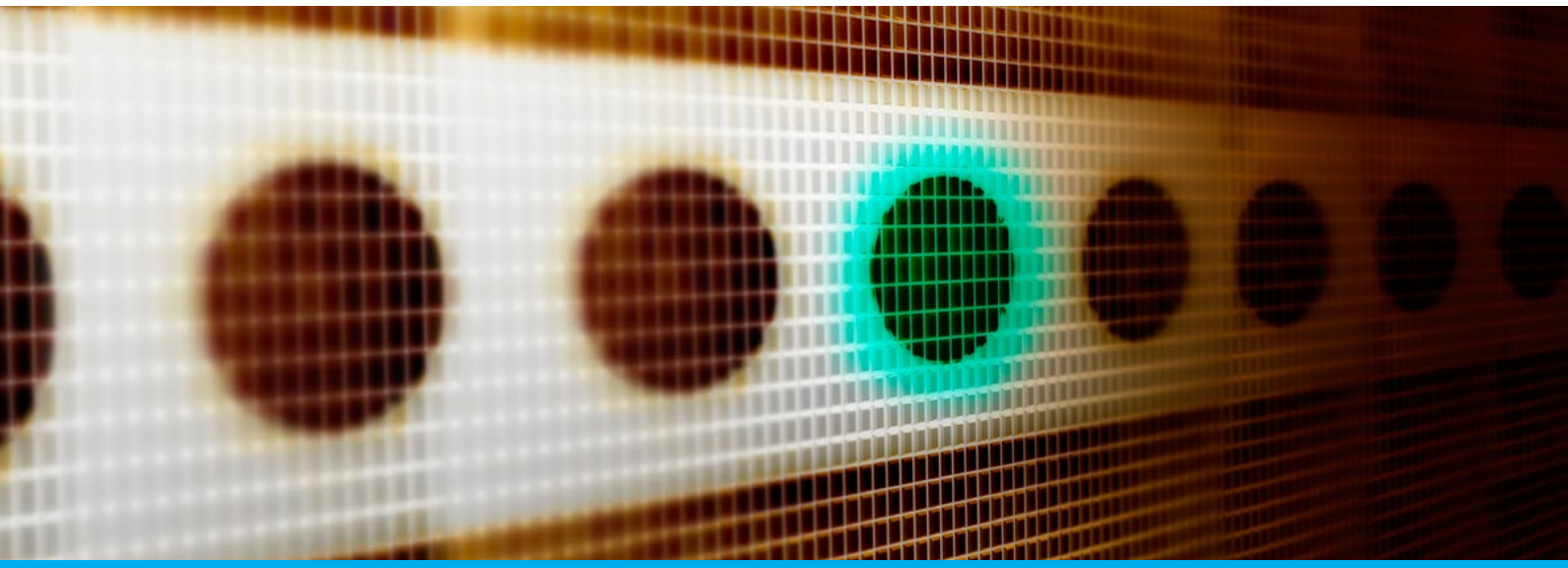
インテル® AVX-512 命令の詳細は、『Intel® Architecture Instruction Set Extensions Programming Reference (インテル® アーキテクチャー命令セット拡張プログラミング・リファレンス)』の第 2 章～第 7 章に記載されています。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® コンパイラーを試そう

次のソフトウェアで利用可能:
インテル® Parallel Studio XE 2015 Composer/Professional/Cluster Edition >

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。



Digimarc 社が新たに実現した電子透かしの埋め込み

インテル® ソフトウェア・ツールにより世界中の開発者が利用する SDK をアップグレード

はじめに

Digimarc 社のコア・テクノロジーは、あらゆるメディアに電子情報を埋め込むことを可能にします。同社の電子透かしテクノロジーは、オーディオ、イメージ、ビデオ、印刷物に、コンピューターやデジタルデバイスが簡単に検出できる方法で永続的な電子情報を埋め込みます。この電子透かしの有無により、企業や検査担当者は、コンテンツが正規のソースから提供されたものかどうか、あるいは変更 / 偽造されているかどうかを素早く判断することができます。さらに、電子透かしは、暗号化されたコンテンツのセキュリティーを強化し、資産を保護するとともに、流出元の特定にも役立ちます。

Digimarc 社のソフトウェア開発キット (SDK) を利用することで、世界中のパートナー各社と開発者はそれぞれの製品に電子透かしを埋め込むことができます。Digimarc 社ではこの SDK をアップデートし、パフォーマンスを向上するとともに、最新のハードウェア機能を利用したいと考えていました。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

課題

マルチコア・プロセッサーを最大限に利用して、SDK に不可欠な数学計算を多用するイメージ処理ライブラリーのパフォーマンス向上を図ることが Digimarc 社の目標でした。以前のライブラリー・バージョンは 2000 年初期のプロセッサーをターゲットにしており、SIMD ベクトル命令とマルチコア・プロセッサーの能力を利用することができませんでした。

マルチコアに移行するためには、すべてのコードの書き直しが必要でした。また、アプリケーションは、複数のオペレーティング・システムで動作できる必要がありました。

ソリューション

Digimarc 社でさまざまなソリューションを検討した結果、最小限の労力で、低レベルのプログラミング手法を使用せずにアプリケーション・コードをベクトル化できたのはインテル® ソフトウェア・ツールだけでした。ベクトル化なしでは、プロセッサーのリソースを最大限に利用することは不可能でした。

Digimarc 社のアプリケーションは SIMD 命令に非常に適しており、1 つのコンパイラーで複数のプラットフォームに対応できることも大きなメリットでした。Digimarc 社のソフトウェア・エンジニアである Goran Negovetic 氏は、「優れた並列性を得るため、ロックを使用しないアルゴリズムが必要でした。つまり、スレッド間でデータを共有しないようにする必要がありました。これは大変な作業でしたが、現在および将来のプロセッサーとプロセッサー・コアの増加に対応できる、ほぼ完璧なスケーラビリティーを実現することができました。」と述べています。

成功の鍵は、さまざまなプラットフォームに移植可能なソリューションを開発できたことです。インテル・ツールにより、1 つのプラットフォーム向けにコードを最適化することで、コードを変更することなく、ほかの OS プラットフォームでも同じ結果を得ることができました。最初に、Windows* 上でコードの開発とチューニングを行い、次に同じコードを OS X* 上でコンパイルしてリリースしました。現在は、同じコードベースで組込み Linux* プラットフォーム向けの作業を行っています。

「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) によりコードを簡単に並列化できました。パフォーマンスの測定にはインテル® VTune™ Amplifier XE が、スレッドエラーの検証にはインテル® Inspector XE が非常に役立ちました。」

Digimarc 社 ソフトウェア・エンジニア Goran Negovetic 氏

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

インテル® ソフトウェア・ツールにより、Digimarc 社は非常に高いパフォーマンス目標を達成することができました。新しいコードベースは、より高度なアルゴリズムを採用し、スケーラビリティーを最大化するため完全に並列化することができます。マルチコア・プロセッサーでは優れたパフォーマンスを発揮する一方、シングルコアではレガシーコードを実行しなければならないため、SIMD ベクトル命令によりデータ並列処理を利用しました。

Digimarc 社では、開発プロセス全体にインテル・ツールを導入したところ、「インテル® TBB によりコードを簡単に並列化できました。パフォーマンスの測定にはインテル® VTune™ Amplifier XE が、スレッドエラーの検証にはインテル® Inspector XE が非常に役立ちました。」と Goran 氏は述べています。

トレーニングから実際のコードのベクトル化まで、インテルのサポートは非常に重要でした。「ツールのトレーニング、コードの最適化とベクトル化、新しい言語機能 (C++ 11) のサポート、OS X* コンパイラーのサポートなど、プロジェクト全体をとおしてインテルの協力が得られたことは大きかったです。」と Goran 氏は振り返っています。

結果

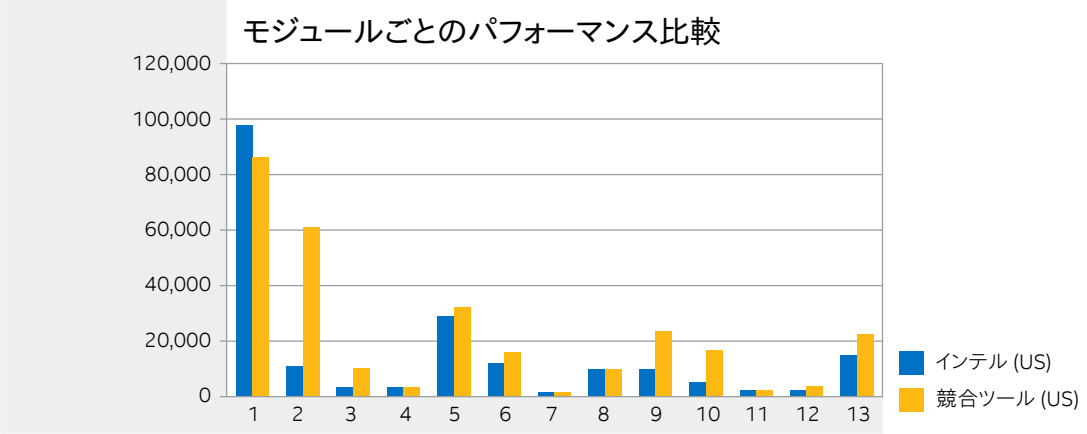
インテル® ソフトウェア・ツールは使いやすく、ほかへの影響が少ないプラグマや自動ベクトル化により、最小限のコード変更で優れたパフォーマンスを実現できました。さらに、ほかのツールでは得られない生産性の向上と開発コストの削減を Digimarc 社にもたらしました。

「優れた並列性を得るため、ロックを使用しないアルゴリズムが必要でした。つまり、スレッド間でデータを共有しないようにする必要がありました。これは大変な作業でしたが、現在および将来のプロセッサーとプロセッサー・コアの増加に対応できる、ほぼ完璧なスケーラビリティーを実現することができました。」

Digimarc 社 ソフトウェア・エンジニア Goran Negovetic 氏

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

モジュール	インテル・ツール (US)	競合ツール (US)
1	98632	87138
2	8634	62013
3	338	8208
4	1350	1173
5	29810	32457
6	9641	13470
7	724	690
8	7010	5951
9	7997	22493
10	3955	13019
11	744	882
12	1547	2356
13	13826	22105
合計	**184208**	**271955**



インテル® ソフトウェア・ツールは、Digimarc 社のパフォーマンス向上に大きく貢献し、競合製品よりも優れた結果をもたらしました。

利点	
パフォーマンスの向上	シングルコアで約 1.6 倍
生産性	1 つのコンパイラーで複数のプラットフォーム (Windows*、OS X*) 向けの最適化に対応 自動ベクトル化により開発時間を短縮し、時間と手間のかかる低レベルのプログラミング作業を軽減
フォワード・スケーリングと クロスプラットフォーム開発	1 つのコードベースで複数のプロセッサー・ターゲットに対応
顧客満足度	顧客の求めるパフォーマンスを期間内に実現し、顧客満足度を向上
開発コスト	開発期間を短縮し、開発コストを削減

システム構成	
オペレーティング・システム	Microsoft* Windows Server* 2008 R2 Standard SP1 (6.1.7601)、HotFixes 171 (最新: KB982018)
プロセッサー	インテル® Core™ i7-3770 プロセッサー 3.40GHz
周波数	3401MHz
メモリー	8134MB

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

まとめ

多種多様な業界と幅広い顧客向けの革新的な技術への取り組みでは、最新のハードウェア機能を利用し、クロスプラットフォームの使用モデルに対応した開発が、戦略的にも、技術的にも重要です。インテル® ソフトウェア・ツールにより、Digimarc 社は SDK をアップグレードし、パフォーマンスの大幅な向上、開発期間の短縮を実現して、電子透かしの革新に成功しました。

Digimarc 社について

アメリカ・オレゴン州ビーバートンにある Digimarc 社は、世界中の企業と政府機関がオーディオ、ビデオ、イメージを含むあらゆる形式のコンテンツを識別できるように、ソリューション開発、知的財産ライセンス、さまざまな業界のビジネスパートナーへの開発サービスを提供しています。Digimarc 社の顧客は、梱包、小売、オーディオ、雑誌、書籍、新聞、ダイレクトメール、金融など広範な業界にわたり、同社の製品には Digimarc Discover*、Digimarc Guardian*、SDK などがあります。

詳細は、www.digimarc.com (英語) を参照してください。

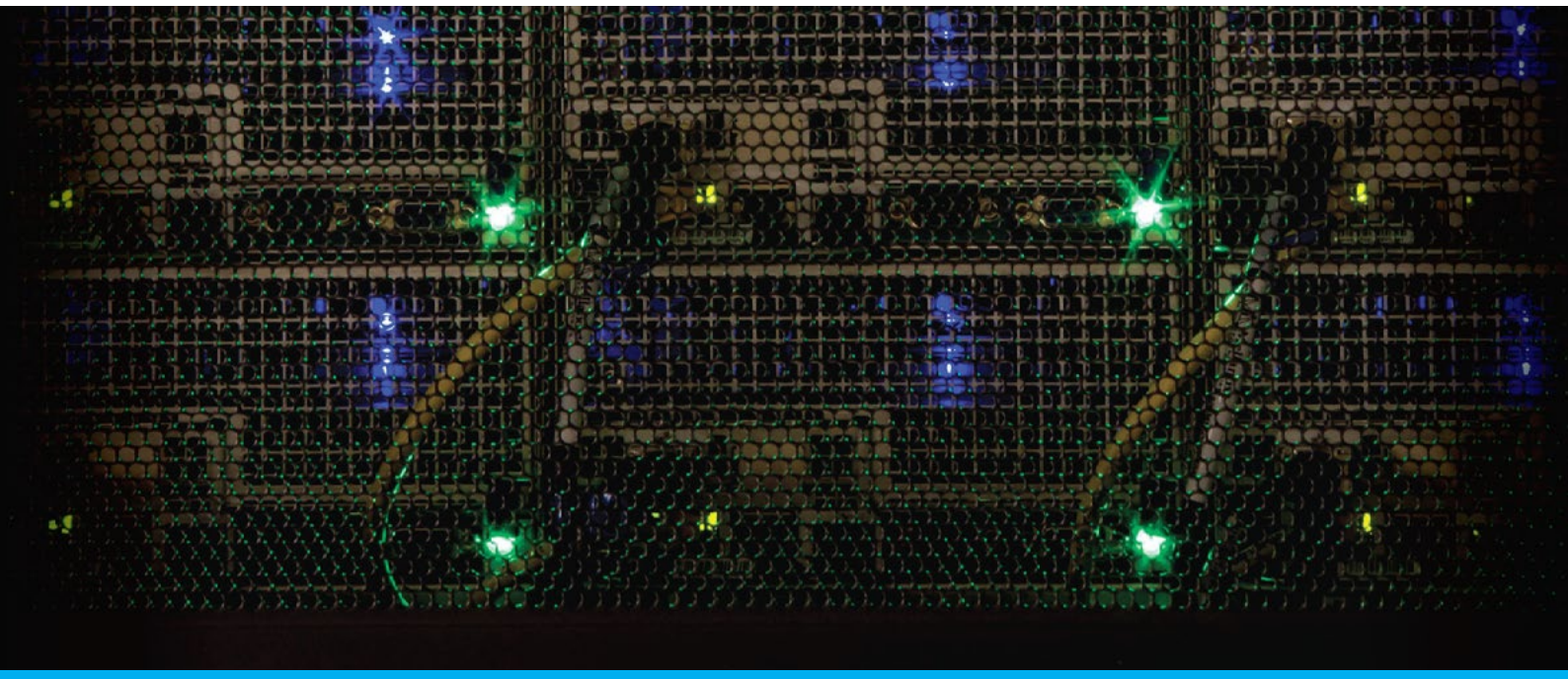
インテル® ソフトウェア開発ツールについて

インテルは、25 年以上にわたって、ハイパフォーマンス・コンピューティング業界において標準規格に準拠したツールを開発者の方々に提供してきました。業界最先端のツールには、インテル® C/C++ および Fortran コンパイラーに加えて、インテル® VTune™ Amplifier XE、インテル® Inspector XE、インテル® Trace Analyzer & Collector などのパフォーマンス・プロファイルと解析ツールがあります。インテル® MPI ライブラリー、インテル® MKL、インテル® Cilk™ Plus、インテル® TBB などのパフォーマンス・ライブラリーとプログラミング・モデルは、現在、そして将来に対応したアプリケーションのビルドに必要なツールを開発者に提供します。

インテル® ソフトウェア開発ツールの詳細は、http://intel.ly/perf-tools (英語) を参照してください。

インテル®
ソフトウェア開発ツールを試そう >

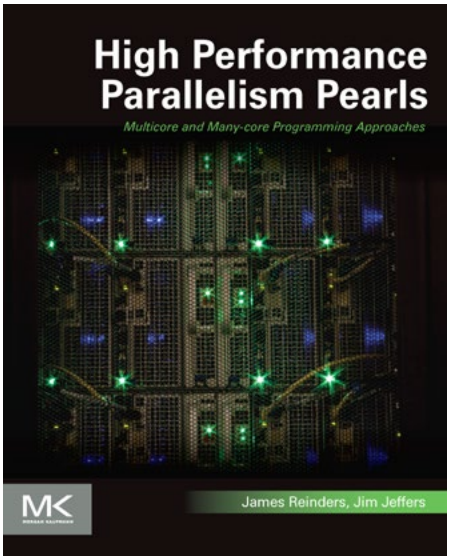
インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。



並列パフォーマンスを利用するための最良かつ実用的な手法

『High Performance Parallelism Pearls』

69 人の専門家の経験が凝縮された 28 章からなる "クックブック"



簡単にインテル® Xeon Phi™ コプロセッサー・ファミリーの優れた並列性を利用してコードを実行できるため、最適化に集中し、ハイパフォーマンスを実現することが可能です。並列処理を細かくチューニングすることで、正しいアプリケーションを正しく効率良いアプリケーションにすることができます。

インテル コーポレーションの並列プログラミング・エバンジェリストである James Reinders とインテル コーポレーションのエンジニアである Jim Jeffers により執筆された最新の書籍は、69 人の専門家の実際の経験を基に、インテルのマルチコアおよびメニーコア・プロセッサーを最大限に利用するための創意工夫を紹介しています。

本書は、2013 年に出版された「Intel® Xeon Phi™ Coprocessor High-Performance Programming 」の続編ともいえる技術書であり、さまざまな分野での応用技術が紹介されています。

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

概要

『High Performance Parallelism Pearls』には、次のトピックが含まれます。

- インテル® Xeon Phi™ コプロセッサー、インテル® Xeon™ プロセッサー、その他のマルチコア・プロセッサー・ベースのシステムの計算能力を最も効率良く利用する方法
- 化学、エンジニアリング、環境科学など、幅広い業界と分野に渡るプログラミング例
- これらのシステムの強力な機能だけでなく、これらのヘテロジニアス・システムにおいて並列処理を最大限に利用する方法を示す成功事例

各章で、役立つ詳細な技術情報が得られます。

- さまざまな分野の 69 人の専門家によって実証された結果
- プロセッサーとコプロセッサー向けの最適化の実践的な手法と説明
- 実際のソースコード: 主要部分を書籍で紹介し、完全なコードはダウンロード可能

購入情報

『High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches』(ISBN 978-0128021187、Morgan Kaufmann 刊) は、**Elsevier Store** および **Amazon** から購入できます。

lotsofcores.com で詳細、レビュー、第 1 章の完全版をご覧いただけます。

「James Reinders と Jim Jeffers の新刊『High Performance Parallelism Pearls』は、インテル® Xeon Phi™ 製品ファミリーの優れた性能について、69 人の HPC 専門家の経験を 28 章にまとめています。」

– Rob Farber 氏

2014 年 9 月 30 日に開催された TechEnablement 社の Teaching the World About Intel Xeon Phi, TechEnablement において

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

BLOG HIGHLIGHTS

GNU* ツールにおけるインテル® Memory Protection Extensions (インテル® MPX) のサポート

IGOR ZAMYATIN »

不正なメモリーアクセスは、C/C++ プログラムでよくある問題で、時間のかかるデバッグ、不安定なプログラム、脆弱性につながります。ソフトウェアの不具合に対する攻撃の多くは、バッファー・オーバーフロー (またはバッファー・オーバーラン) による適切でないメモリーアクセスに関連しています。このようなメモリーの不具合を見つけ、プログラムを攻撃から守るための既存の手法とツールは、ソフトウェアによるソリューションしかなく、保護するコードのパフォーマンスを低下させます。

インテルの新しい ISA 拡張であるインテル® Memory Protection Extensions (インテル® MPX) は、アプリケーションにおいてわずかなパフォーマンス・オーバーヘッドでメモリーを保護します。この新しい拡張を利用するには、OS カーネル、binutils、コンパイラー、システム・ライブラリー・サポートの変更が必要です。

インテル® MPX は、" 境界レジスター " と呼ばれる新しいレジスターにポインター境界を格納し、命令によってこれらのレジスターを操作します (詳細は、プログラミング・リファレンスを参照してください)。そのため、最初に binutils と GCC で新しいハードウェア機能のサポートを実装する必要があります。

この記事では、インテル® MPX をサポートするための GNU* Binutils、GCC、Glibc の変更について説明します。

この記事の続きはこちらでご覧になれます。 >

インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、**software.intel.com/en-us/articles/optimization-notice/#opt-jp** を参照してください。

The Parallel Universe Issue 19 日本語訳は iSUS 事務局内 Parallel Universe 編集部が担当いたしました。本誌に関するご意見やご感想、ご提案がございましたら、ぜひお聞かせください。今後の翻訳・活動のための参考とさせていただきます。

1. Web フォーラムをご利用いただく場合
   http://www.isus.jp/forum/forum_article/ よりアクセスし、
   フォーラムへご投稿ください。
2. 電子メールをご利用いただく場合
   info@isus.jp 宛に電子メールをお送りください。

---

iSUS – IA Software User Society (http://www.isus.jp/)
iSUS は、IA プラットフォーム向けにソフトウェアを開発する開発者に向けて、開発に有用な情報を発信することを目的として設立され、現在も精力的に活動を続けています。

編集後記

一月某日、翻訳誌の一連の管理を任されました。一から雑誌を作るのではなく、翻訳元が存在するのだからそこまで大した作業は必要ないだろうと、当初は軽く考えていました。しかし、今となっては当時の浅はかさを呪うばかりです。翻訳作業のほかに、翻訳元の記述内容の正当性チェック (本誌は該当しませんが、まれに間違っている場合があります!) と、日本語としての表現が間違っていないかチェックする必要があります。また、マーケティング観点でのチェックにブランディング・チェック、さらには英語と日本語では文字数が異なってくるので、改めて冊子としての体裁を整える作業も欠かせません。作業担当者間のスケジュール調整も当然重要です。無事に全行程を終え公開までこぎつけることができ、各種作業を担当していただいた関係者の皆様に感謝を捧げるとともに、読者の皆様に少しでも有用な情報が届くことを願います。

iSUS 事務局 Parallel Universe 編集部員 田中

Software

The Parallel Universe

© 2015 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、Intel Xeon Phi、Cilk、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。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の使用許諾を受けて使用しています。