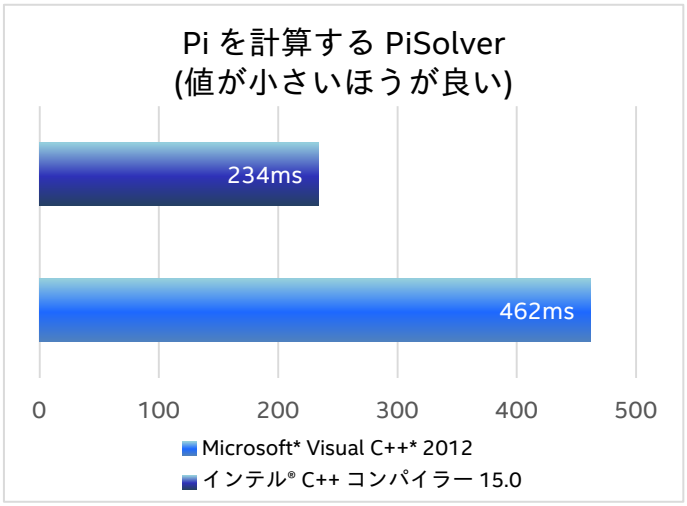
スレッド化されていないアプリケーションでも大幅なパフォーマンス向上を容易に実現

はじめに

本ガイドは、インテル® Parallel Studio XE を使用してアプリケーション中の「hotspot」(多くの時間を費やしているコード領域) を見つけ、それらの領域を再コンパイルすることでアプリケーション全体のパフォーマンスを向上する方法について説明します。

1 つのファイルを再コンパイルするだけで違いが出るのでしょうか?

はい。多くの場合、インテル® Parallel Studio XE の最適化コンパイラーを使用して、1 つのファイルを再コンパイルするだけでパフォーマンスが大幅に向上します。必ずしもアプリケーション全体を再コンパイルする必要はありません。これは、シリアル・アプリケーションと並列アプリケーションの両方に当てはまります。



システム構成: インテル® Core™ i5-3550 プロセッサー 3.30GHz、4 コア、4GB RAM、Microsoft* Windows Server* 2008 R2 Enterprise SP 1 (64 ビット)。

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。* 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。ベンチマークの出典: インテル コーポレーション
最適化に関する注意事項: ドキュメントの最後を参照

パフォーマンス向上のためのステップ

1. hotspot の特定: アプリケーションが時間を費やしている場所を測定する

効率良く最適化を行うには、多くの時間を費やしているアプリケーションのコード部分を最適化します。すでに高速な部分を最適化しても、パフォーマンスはほとんど向上しません。「hotspot」とは、アプリケーションが多くの時間を費やしている場所を指します。hotspot は、インテル® VTune™ Amplifier XE のようなプロファイリング・ツールを使用すると、容易に特定できます。必要のない最適化に時間をかけないでください。hotspot を特定することが大切です。

hotspot が特定できたら、次は何をすれば良いでしょうか。場合によっては、プログラムの実行を高速化する方法がすぐに分かることもあります。例えば、ある操作を繰り返し実行している場合、実行回数を 1 回にできることがあります。しかし、ほとんどの場合、答えはそれほど明確ではありません。このため、「アドバイスを表示したり、自動的に処理できませんか?」という質問をよく受けます。幸いなことに、多くの場合はそれが可能です。

2. hotspot の最適化: hotspot のみ (または 1 つのファイルのみ) 再コンパイルする

多くの場合、インテル® C++ コンパイラーで hotspot が含まれているファイルを再コンパイルするだけでパフォーマンスが向上します。小規模なアプリケーションでは、すべてを再コンパイルしてもそれほど時間はかかりませんが、多くのモジュールやプロジェクトが含まれる大規模なアプリケーションでは、すべてを再コンパイルすることは実用的ではありません。幸い、アプリケーション全体の再コンパイルが必要になることはめったにありません。ほとんどの場合、いつくかのファイル、もしくは 1 つのプロジェクトの再コンパイルが必要になるだけです。インテル® コンパイラーは Microsoft* コンパイラーとバイナリーおよびデバッグ互換なので、これらのコンパイラーでビルドしたオブジェクトをシームレスに利用できます。

実践

ステップ 1: インストールと設定

インテル® Parallel Studio XE のインストールと設定

1. インテル® Parallel Studio XE の評価版をダウンロードしてインストールします。

ステップ 2: サンプル・アプリケーションのインストールと実行

1. 「pisolver-2bsample.zip」サンプルファイルをローカルマシンにダウンロードします。
2. このサンプルは、Microsoft* Visual Studio* を使用して作成された MFC ダイアログベースのプログラムです。ソリューション・ファイルは、Microsoft* Visual Studio* 2010、2012、2013 で利用できます。このプログラムは、内部的に C 関数を呼び出して π の値を求め、GUI に結果を表示します。
3. 書き込み可能なディレクトリーまたはシステムの共有ディレクトリー (<My Documents>\Intel Parallel Studio XE\samples フォルダーなど) に「PiSolver+Sample.zip」ファイルを展開します。

サンプルのビルド:

- Microsoft* Visual Studio* 環境でデフォルトの Microsoft* Visual C++* コンパイラーを使用して PiSolver サンプル・アプリケーションを 「Release」モードでビルドします。
- Microsoft* Visual Studio* で、**[File (ファイル)] > [Open (開く)] > [Project/Solution... (プロジェクト/ソリューション...)]** を選択し、PiSolver.sln ファイルのあるフォルダー (zip を展開したフォルダー) に移動します (図 1)。

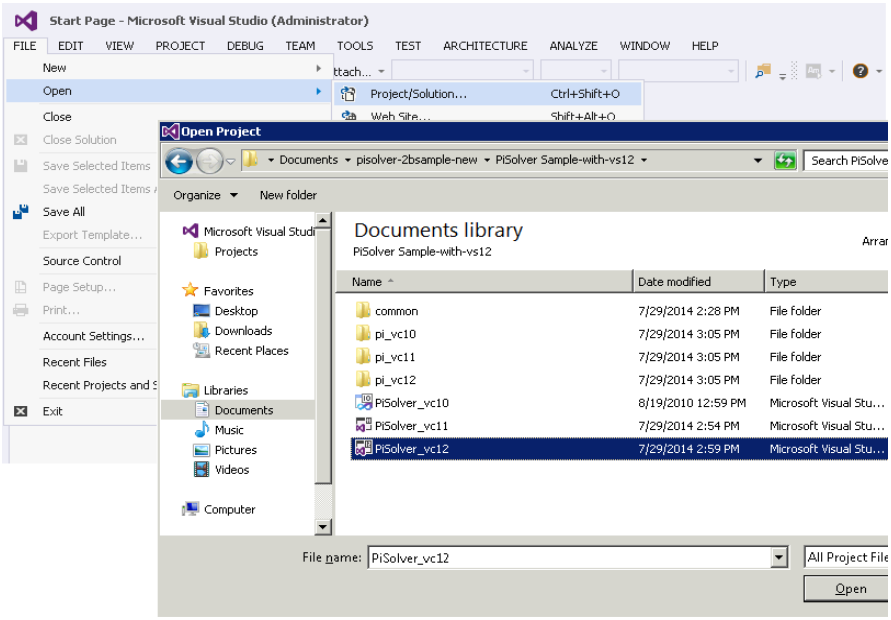


図 1

- Microsoft* Visual C++* で Release (最適化) 構成設定を使用して、ソリューションをビルドします。**[Solution Configuration (ソリューション構成)]** ドロップダウン・リストから Release 設定を選択します (図 2)。

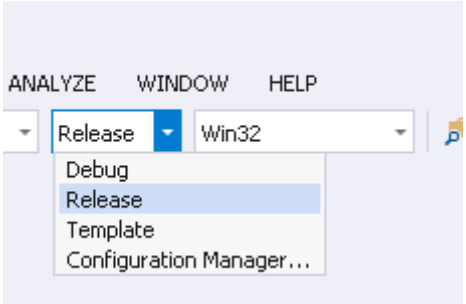


図 2

- **[Build (ビルド)] > [Build Solution (ソリューションのビルド)]** でプログラムをビルドします (図 3)。

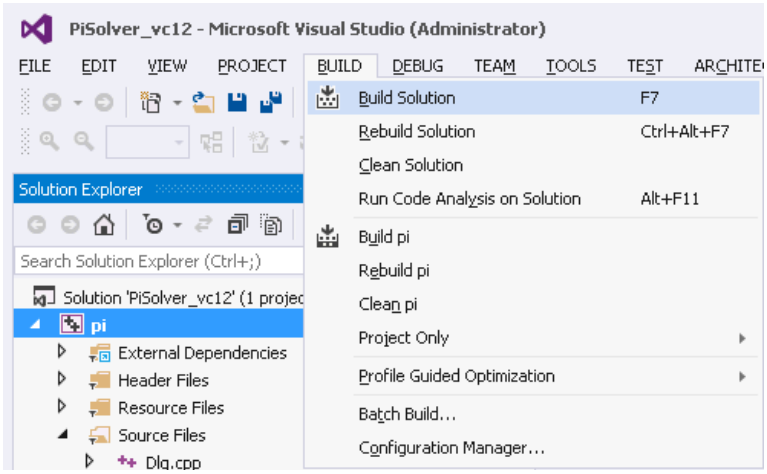


図 3

- Microsoft* Visual Studio* で **[Debug (デバッグ)] > [Start Without Debugging (デバッグなしで開始)]** を選択して、アプリケーションを実行します (図 4)。

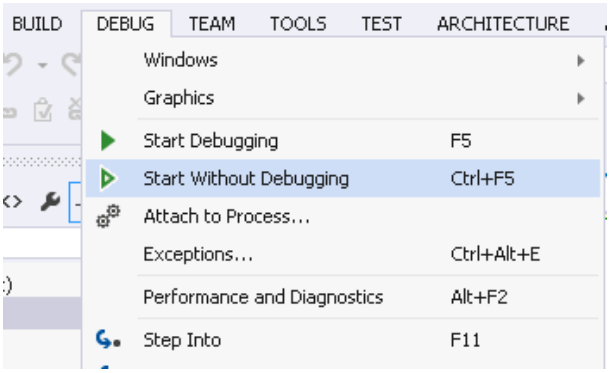


図 4

- **[Calculate (計算)]** ボタンをクリックして π の値を計算し、処理に要した時間 (ミリ秒) を確認します (図 5)。

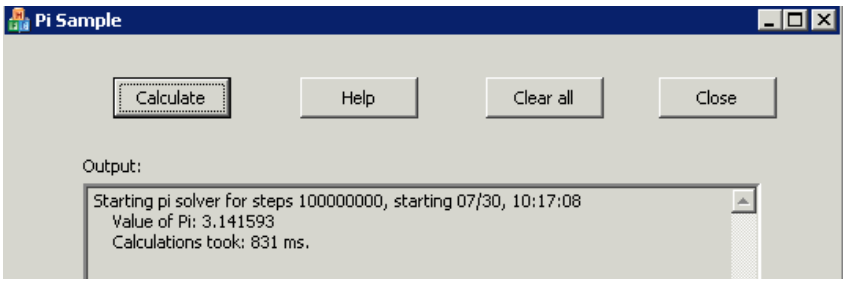


図 5

ステップ 3: インテル® VTune™ Amplifier XE の実行: hotspot を特定する

1. Release (最適化) 構成であっても、デバッグシンボルが生成されることを確認します。デバッグシンボルを生成することで、インテル® VTune™ Amplifier XE はアプリケーションに関する多くの情報を提供できるようになります。
   - **[Solution Explorer (ソリューション エクスプローラ)]** ウィンドウで **pi** をシングルクリックして、**pi** プロジェクトをハイライトします。
   - **[Project (プロジェクト)] > [Properties (プロパティ)]** を選択して **[Pi Property Pages (pi プロパティ ページ)]** ダイアログボックスを開きます。
   - **[Configuration Properties (構成プロパティ)]** が展開されていない場合は展開します。
   - **[C/C++]** を展開して、**[General (全般)]** をクリックします。
   - **[Debug Information Format (デバッグ情報の形式)]** で **[Program Database (/Zi) (プログラム データベース (/Zi))]** を選択して **[Apply (適用)]** をクリックします (図 6)。

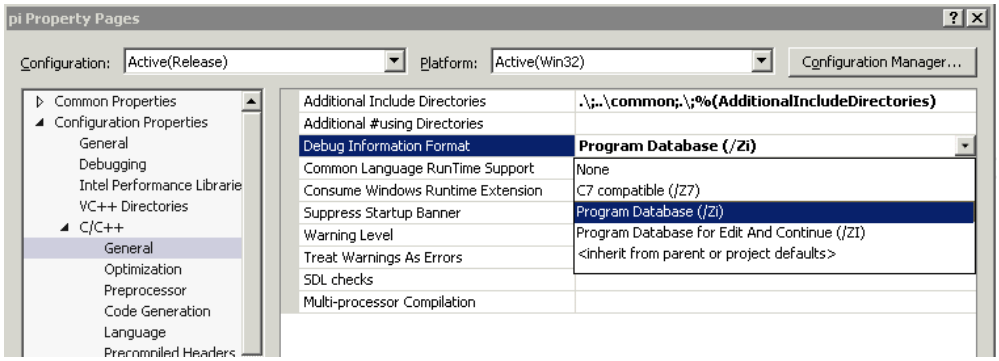


図 6

- **[Linker (リンカ)]** プロパティーを展開して、**[Debugging (デバッグ)]** をクリックし、**[Generate Debug Info (デバッグ情報の生成)] > [Yes (/DEBUG) (はい (/DEBUG))]** を選択します。**[Apply (適用)]**、そして **[OK]** をクリックします (図 7)。

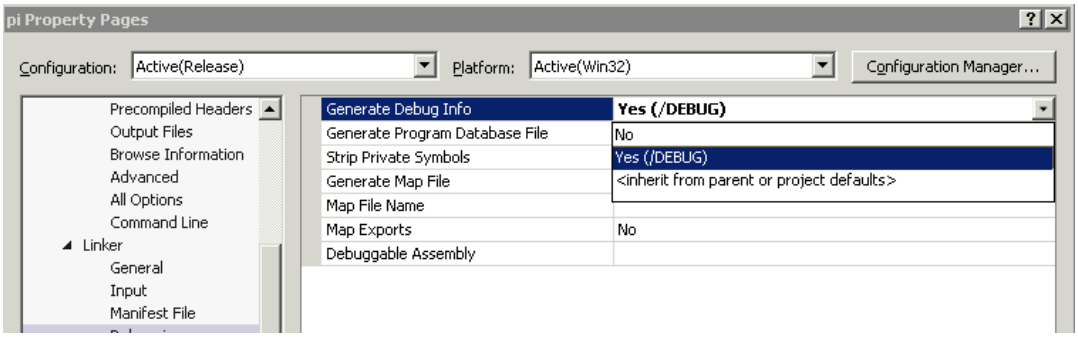


図 7

2. インテル® VTune™ Amplifier XE ツールバーにある **[New Analysis (新しい解析)]** ボタンをクリックし (図 8)、**[Hotspots]** を選択します (図 9)。

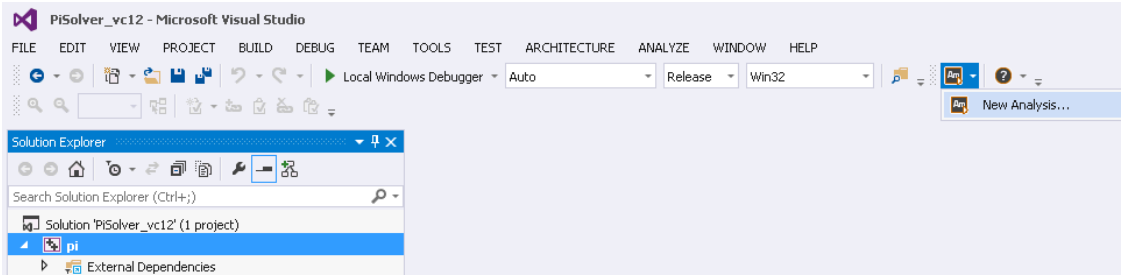


図 8

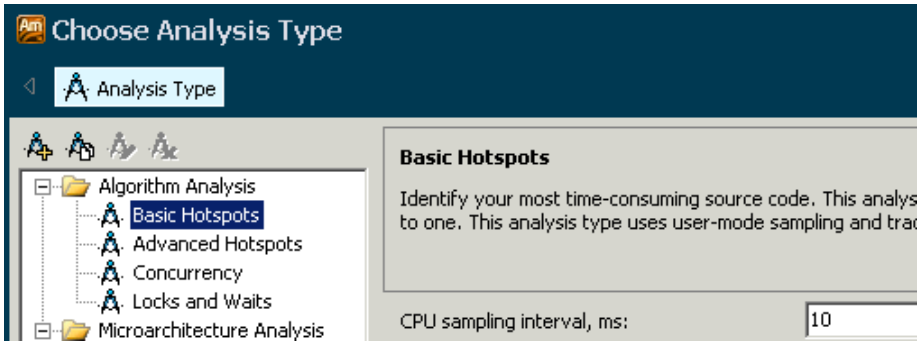


図 9

3. **[Start (開始)]** ボタンをクリックします。PiSolver アプリケーションが起動します。
4. PiSolver アプリケーションで、**[Calculate (計算)]** ボタンをクリックして計算を行い、ダイアログボックスに結果と時間が表示されたら、**[Close (閉じる)]** ボタンをクリックします。この時点で、インテル® VTune™ Amplifier XE によるデータ収集は完了し、図 10 のような hotspot レポートが表示されます。(hotspot の分析結果を含むテキストボックスが表示されます。内容を確認してからこのテキストボックスを閉じます。)

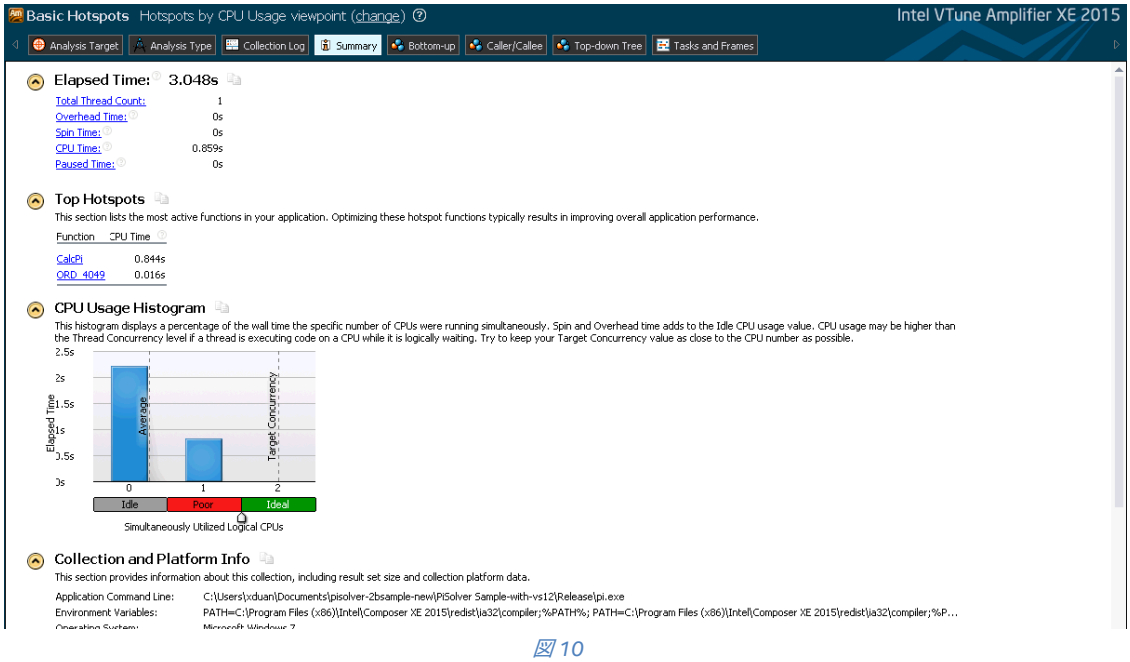


図 10

5. **[Bottom-up (ボトムアップ)]** をクリックしてすべての hotspot を確認します (図 11)。

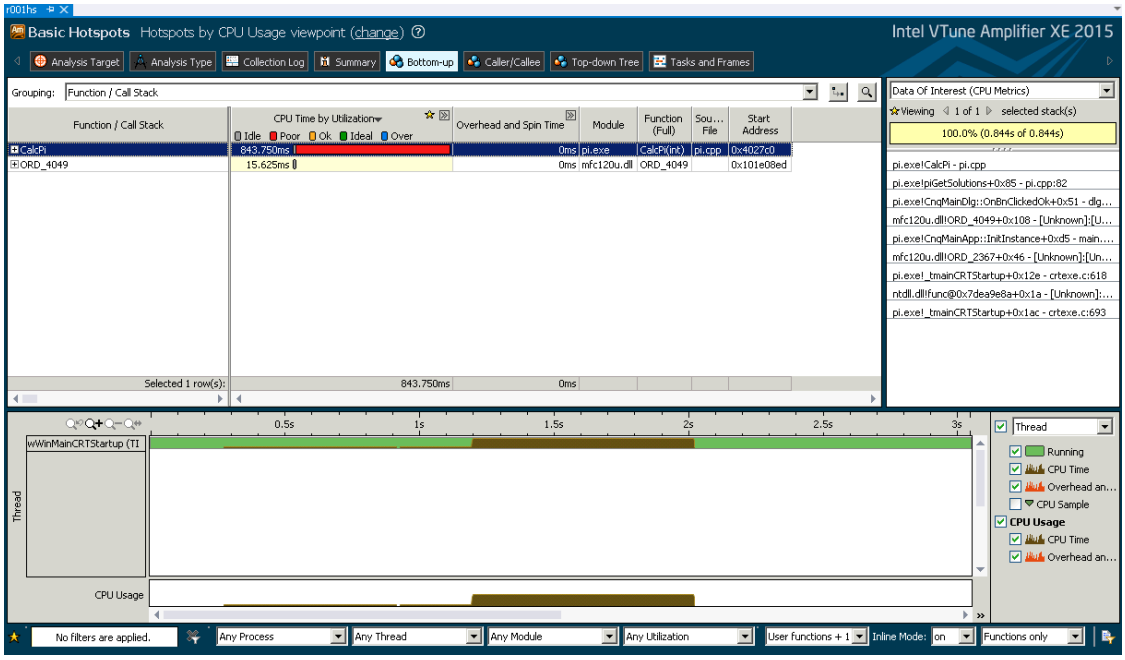


図 11

1. hotspot 分析からの結果。結果が収集されるたびに番号 (000) が上がります。
2. **[Function/Call Stack (関数/コールスタック)]** は、hotspot データのデフォルトのグループレベルです。矢印ボタンをクリックして、グループレベルを変更できます。
3. 関数名の前にあるプラス記号をクリックすると、選択された関数のコールスタックを表示できます。展開されたレベルごとに、選択された関数の呼び出し元、次にその呼び出し元の呼び出し元、のように順に表示されます。
4. CPU 時間は、論理プロセッサーで関数を実行するのにかかる時間です。複数のスレッドの場合は CPU 時間が合計されます。これは、hotspot 分析結果の **[Data of Interest (特定のデータ)]** 列です。
5. 選択された関数のスタック情報全体がグリッドに表示されます。黄色のバーは、hotspot 関数の CPU 時間に対する選択されたスタックの割合を示しています。
6. タイムライン・ビューにより、実行時間を通してスレッド全体の CPU アクティビティーを確認することができます。

6. ボトムアップの関数リストで CalcPi の先頭にあるプラス記号をクリックして、関数のコールスタックを展開します。そして、関数 CalcPi をダブルクリックして、ソースコードを表示します。ソースファイル名を確認します (図 12)。

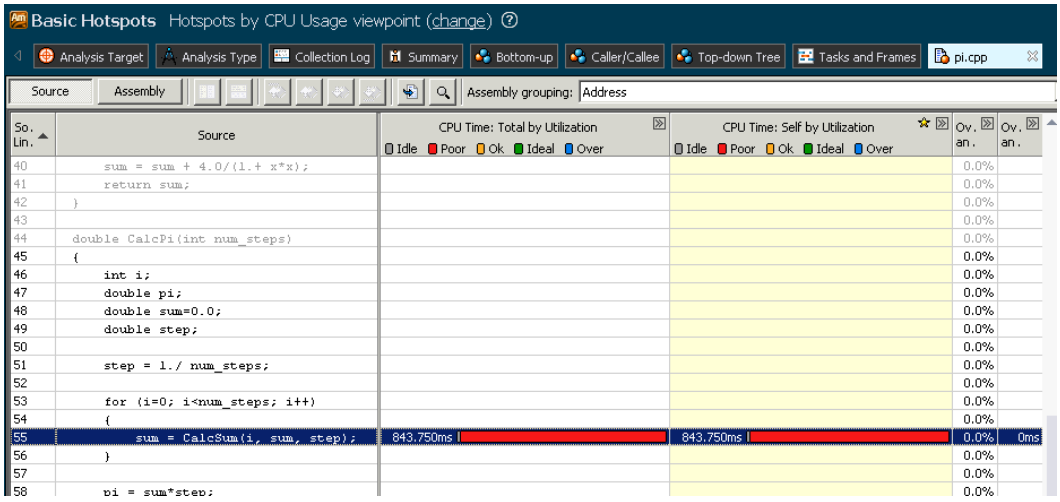


図 12

7. 一部のアプリケーションでは、トップダウン・ツリー・ビューを使用したほうがコールツリーを確認しやすいでしょう。大規模なアプリケーションでは、hotspot を含む関数を特定するために大規模な関数ツリーを展開することになります。PiSolver サンプルでは、hotspot は pi.cpp に含まれています。

ステップ 4:インテル® C++ コンパイラーでコンパイルする

1. Microsoft* Visual Studio* の **[Solution Explorer (ソリューション エクスプローラ)]** で、hotspot が含まれているファイルのプロジェクトを特定します。PiSolver サンプルでは、pi.cpp は **pi** プロジェクトに含まれています。
2. **[Solution Explorer (ソリューション エクスプローラ)]** で **pi** をクリックして、**pi** プロジェクトをハイライトします。
3. **[プロジェクト] > [Intel Compiler XE 2015 (インテル(R) コンパイラー XE 2015)] > [Use Intel C++ (インテル(R) C++ を使用)]** を選択します。
4. インテル® C++ コンパイラーの **[Confirmation (確認)]** ボックスが表示されます。**[OK]** をクリックします。
5. Microsoft* C++ コンパイラーを使用するようにプロジェクトの設定を変更します。

Microsoft* Visual Studio* 2010 ユーザー:

- **[Project (プロジェクト)] > [Properties (プロパティ)] > [Configuration Properties (構成プロパティ)] > [C/C++] > [General [Intel C++] (全般 [インテル(R) C++])]** で **[Use Visual C++ Compiler (Visual C++ コンパイラーの使用)]** を **[Yes (はい)]** に変更します (図 13)。

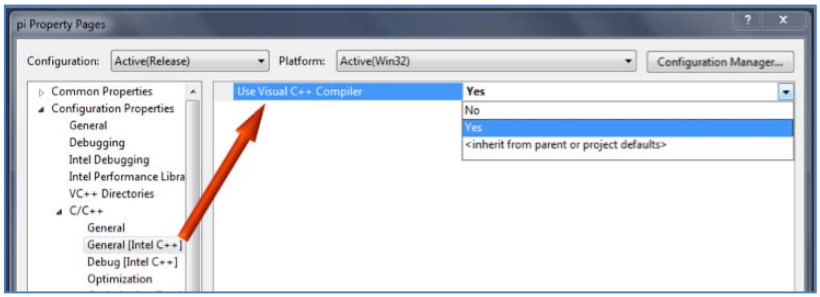


図 13

- **[Apply (適用)]**、そして **[OK]** をクリックします。これで、プロジェクトで Microsoft* C++ コンパイラーを使用できるようになりました。

Microsoft* Visual Studio* 2012 および 2013 の場合:

- **[Project (プロジェクト)] > [Properties (プロパティ)] > [Configuration Properties (構成プロパティ)] > [General (全般)] > [Platform Toolset (プラットフォーム・ツールセット)]** を Visual Studio 2012 (v110) または Visual Studio 2013 (v120) に変更します (図 14)。

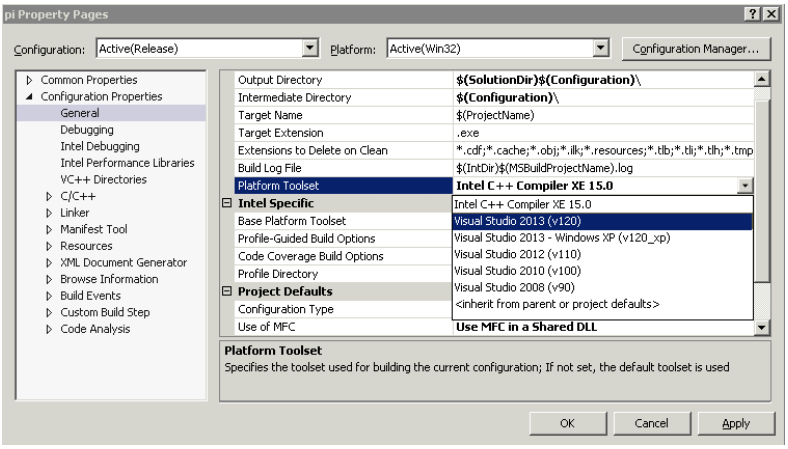


図 14

6. **[Apply (適用)]**、そして **[OK]** をクリックします。現在、プロジェクトは Microsoft* C++ コンパイラーを使用するように設定されています。**pi.cpp** ファイルでインテル® C++ コンパイラーを使用するように設定します。

Microsoft* Visual Studio* 2010 ユーザー:

[Solution Explorer (ソリューション エクスプローラ)] で **pi** プロジェクト、そしてソースファイルを展開します。pi.cpp ファイルを右クリックして、**[Properties (プロパティ)] > [Configuration Properties (構成プロパティ)] > [C/C++] > [General [Intel C++] (全般 [インテル(R) C++])]** で **[Use Visual C++ Compiler (Visual C++ コンパイラーの使用)]** を **[No (いいえ)]** に変更します (図 15)。

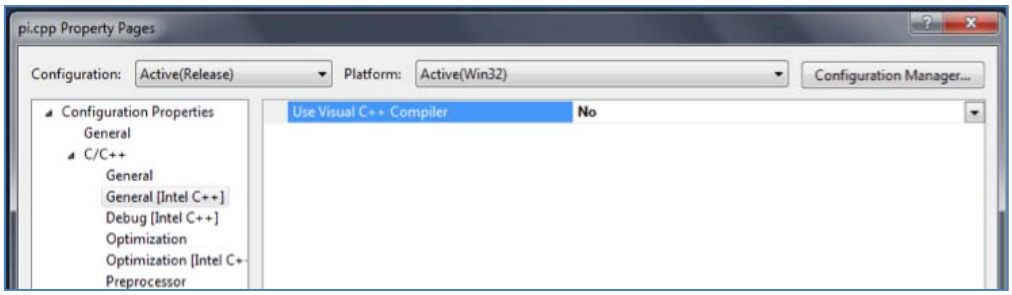


図 15

[Apply (適用)]、そして **[OK]** をクリックします。

注: Visual Studio* 2010 では、Ctrl + 左クリックで複数のファイルを選択して、インテル® C++ コンパイラーでビルドすることもできます。

1. pi プロジェクトをクリックしてハイライトし、**[Build (ビルド)] > [Build Pi (pi のビルド)]** を選択してビルドします。**[Output (出力)]** ペインには、pi.cpp はインテル® コンパイラーでビルドされ、その他のファイルは Microsoft* コンパイラーでビルドされることを示すメッセージが表示されます。
2. **[Debug (デバッグ)] > [Start Without Debugging (デバッグなしで開始)]** を選択して PiSolver アプリケーションを実行し、アプリケーションのウィンドウにある **[Calculate (計算)]** ボタンをクリックします。Microsoft* Visual C++* コンパイラーで pi.cpp をコンパイルした場合よりも大幅に高速化されていることが確認できます。

Microsoft* Visual Studio* 2012 および 2013 の場合:

[Solution Explorer (ソリューション エクスプローラ)] で pi プロジェクト、そしてソースファイルを展開します。pi.cpp ファイルを右クリックして、**[Intel Compiler XE 2015 (インテル(R) コンパイラー XE 2015)] > [Use Intel C++ (インテル(R) C++ を使用)]** を選択します (図 16)。

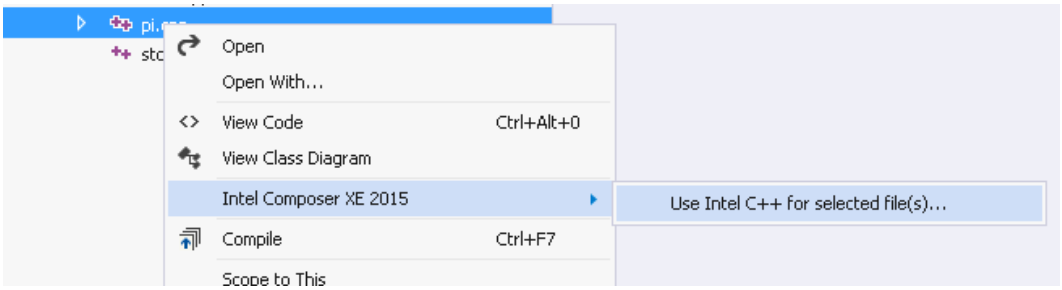


図 16

[OK] をクリックします。

注: Visual Studio* 2012 および 2013 では、Ctrl + 左クリックで複数のファイルを選択して、インテル® C++ コンパイラーでビルドすることもできます。

1. pi プロジェクトをクリックしてハイライトし、**[Build (ビルド)] > [Build Pi (pi のビルド)]** を選択してビルドします。**[Output (出力)]** ペインには、pi.cpp はインテル® コンパイラーでビルドされ、その他のファイルは Microsoft* コンパイラーでビルドされることを示すメッセージが表示されます。
2. **[Debug (デバッグ)] > [Start Without Debugging (デバッグなしで開始)]** を選択して PiSolver アプリケーションを実行し、アプリケーションのウィンドウにある **[Calculate (計算)]** ボタンをクリックします。Microsoft* Visual C++* コンパイラーで pi.cpp をコンパイルした場合よりも大幅に高速化されていることが確認できます。

結果

テストシステムでは、インテル® C++ コンパイラー XE で再コンパイルしただけで PiSolver の実行速度が 96% も向上しました。

アプリケーション	PiSolver
再コンパイル前 [1]	0.462 秒
再コンパイル後 [2]	0.234 秒
スピードアップ	1.97 倍

システム構成: インテル® Core™ i5-3550 プロセッサー 3.30GHz、4 コア、4GB RAM、Microsoft* Windows Server* 2008 R2 Enterprise SP1 (64 ビット)。

この例では、インテル® C++ コンパイラー XE で再コンパイルしただけでパフォーマンスが向上しました。スレッド化されていないアプリケーションを含め、ほとんどの場合は再コンパイルするだけで大幅なパフォーマンスの向上が得られます。

ライブラリー関数に多くの時間が費やされていることがインテル® VTune™ Amplifier XE で分かることもあります。その場合、ライブラリー関数をより高速なものに置換することで、アプリケーションを簡単にスピードアップすることができます。

最適化コンパイラーに加えて、インテル® Parallel Studio XE for C++ には、マルチコアとベクトル化の利点を活用する効率良いコードを簡単に記述できるように支援するライブラリーとプログラミング・モデルが含まれています。

- インテル® IPP は、マルチメディア、データ処理、通信アプリケーション向けに、よく使用される基本的なアルゴリズムを網羅した、数々の最適化関数を提供します。
- インテル® MKL は、極めて高いパフォーマンスが求められるアプリケーション向けの数学ルーチンで、主要な数学関数として BLAS、LAPACK、ScaLAPACK1、スパースソルバー、高速フーリエ変換、ベクトル演算などが含まれています。
- インテル® TBB は広く使用されている C++ テンプレート・ライブラリーであり、安定性を備え、移植性とスケーラビリティーに優れた並列アプリケーションの作成を簡素化します。
- C/C++ 向けの言語拡張であるインテル® Cilk™ Plus は、並列プログラミング用の単純で非常に強力なモデルを提供します。その一方で、ランタイム・ライブラリーとテンプレート・ライブラリーは、並列アプリケーションの実行向けにきめ細やかにチューニングされた環境を提供します。

ステップ 5: インテル® VTune™ Amplifier XE の使用と結果の比較

1. インテル® VTune™ Amplifier XE ツールバーにある **[New Analysis (新しい解析)]** ボタンをクリックし、**[Hotspots]** 解析を選択します。**[Start (開始)]** をクリックすると、アプリケーションが再度実行されます。**[Calculate (計算)]** ボタンを再度クリックして、計算が完了したら **[Close (閉じる)]** ボタンをクリックします。
2. **[Summay (サマリー)]** タブでアプリケーションの合計時間を確認します。CPU 時間が減少しているのが分かります。また、コールツリーでは、ほかの関数は比較的短時間であるのに対して、piGetSolutions だけが突出しています。大規模なアプリケーションでは、1 つの hotspot を解決すると、最適化すべき別の hotspot が見つかることがあります (図 17)。

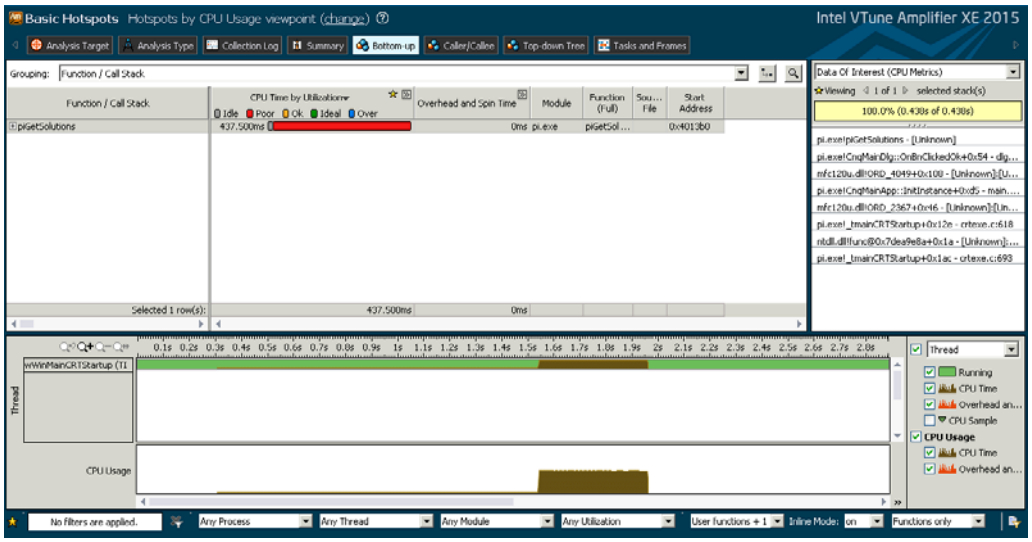


図 17

3. 結果を比較する別の方法として、インテル® VTune™ Amplifier XE の **[Compare Results (結果の比較)]** 機能があります。この機能を使用すると、以前の実行結果と並べて比較し、異なる箇所を確認することができます。これを行うには、結果ファイルを 2 つ選択して、**[Compare (比較)]** ボタンをクリックします。図 18 のような比較画面が表示されます。関数ごとの変更箇所を示す詳細レポートが表示され、CalcPi の時間が大幅に短縮されたことが分かります。

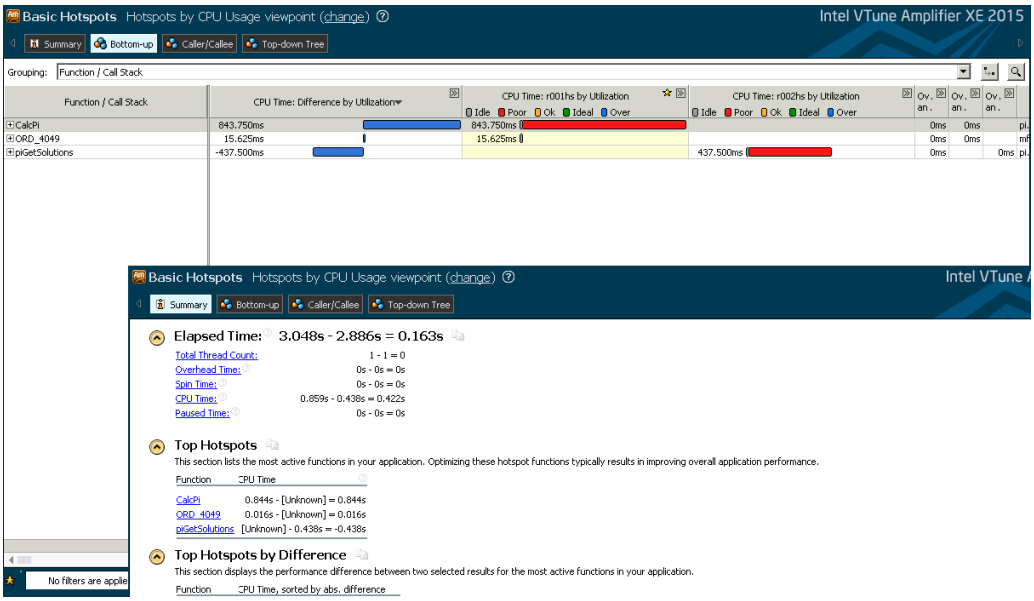


図 18

大規模で複雑なアプリケーションの場合のヒント

PiSolver サンプルは小さなアプリケーションですが、hotspot を素早く検出し、再コンパイルする方法を紹介しています。複数のプロジェクトからなる大規模なアプリケーションでは、hotspot プロファイルで長い時間を費やしている関数が多数表示されることがあります。そのような場合、1 つ (または 2 つ) のファイルだけではなく、hotspot が検出されたプロジェクト全体をリビルドするほうが簡単で、より良いパフォーマンスを得られることがあります。次に hotspot をリビルドする場合のいくつかのヒントを示します。

- Microsoft* Visual C++* コンパイラーの場合は、**[Whole-Program Optimization (プログラム全体の最適化)]** を **[Link-time Code Generation (/GL) (リンク時のコード生成を使用 (/GL))]** に設定してみてください。インテル® コンパイラーの場合は、**[Interprocedural Optimization (プロシージャー間の最適化)]** を **[Multi-file (/Qipo) (複数ファイル (/Qipo))]** に設定してみてください。この最適化を行うことで、アプリケーションによっては、ファイル間のインライン展開とその他のファイル/関数間の最適化によりパフォーマンスが大幅に向上します。どちらのコンパイラーでも、「Release」構成を作成するとこのオプションがデフォルトで有効になります。ただし、最適化を行ったコンパイラーがリンクも行う必要があります。そのため、PiSolver サンプルで行ったように、インテル® コンパイラーで 1 つのファイルだけを再コンパイルすると、インテル® コンパイラーによるプログラム全体の最適化の利点を完全に活用することができません。これは、まさに本ドキュメントの冒頭で紹介した Smoke アプリケーションの結果 (詳細は、リンクされたビデオを参照) に当てはまります。Smoke は多数のプロジェクトからなる非常に大きなアプリケーションですが、hotspot プロジェクトが比較的少ないため、短時間のリビルドでパフォーマンスが大幅に向上しました。
- Smoke のように、ソリューションに多数のプロジェクトがある大規模なアプリケーションでは、hotspot ファイルを含むプロジェクト全体をリビルドするほうが簡単です。プロジェクト全体の設定をインテル® C++ コンパイラー XE のものに切り替え、プロジェクトをリビルドするだけです。この場合、前述の「インテル® C++ コンパイラーを使用してコンパイル」のステップ 5 と 6 はスキップします。
- 多くのアプリケーションでは、プリコンパイル済みヘッダー (今後の使用のために保存されたプリコンパイル済み .h ファイルなど) にも注意が必要です。Microsoft* コンパイラーによってビルドされたプリコンパイル済みヘッダーは、インテル® コンパイラーでは使用することができません。リビルドするか、使用しないようにしてください。インテル® コンパイラーをプロジェクト全体に使用する場合は、インテル® コンパイラーによってプリコンパイル済みヘッダーがビルドされるため問題ありません。ただし、1 つのファイルだけをリビルドする場合は、インテル® コンパイラーでビルドするファイルに対して次の操作を行う必要があります。
    - インテル® コンパイラーでコンパイルするファイル (例えば pi.cpp) を右クリックして、**[Properties (プロパティ)]** を選択します。**[Property Page (プロパティ ページ)]** ボックスで、**[Configuration Properties (構成プロパティ)] > [C/C++] > [Precompiled Headers (プリコンパイル済みヘッダー)] > [Create/Use Precompiled Header (プリコンパイル済みヘッダーの作成/使用)] > [Not Using Precompiled Headers (プリコンパイル済みヘッダーを使用しない)]** を選択します (図 19)。

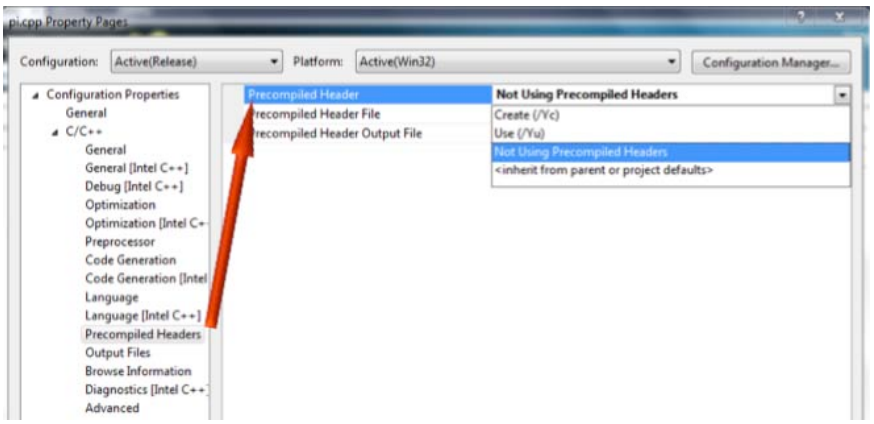


図 19

まとめ

インテル® C++ コンパイラーを使用して 1 つのファイルを再コンパイルするだけでアプリケーションを高速化できます。ポイントは、hotspot を含むソースファイルを再コンパイルすることです。インテル® VTune™ Amplifier XE を使用すると hotspot を特定できるため、最適化作業の労力を軽減できます。

重要な用語と概念

重要な用語

CPU 時間: 論理プロセッサーでスレッドの実行に費やした時間。複数のスレッドの場合は、すべてのスレッドの CPU 時間の合計です。アプリケーション CPU 時間は、アプリケーションで実行されたすべてのスレッドの CPU 時間の合計です。

ターゲット: インテル® VTune™ Amplifier XE を使用して分析する実行ファイル。

重要な概念

hotspot 分析: アプリケーション・フローの理解と、実行に長い時間がかかっているコード領域 (hotspot) の特定に役立ちます。hotspot は、アプリケーション全体のパフォーマンスに大きな影響を及ぼすため、重点的にチューニングを行う箇所です。

インテル® VTune™ Amplifier XE は、関数で費やされた時間順にアプリケーションの関数のリストを作成します。関数のコールスタックも表示するため、時間を費やしている関数がどのように呼び出されているかを確認できます。オーバーヘッドの少ない (約 5%) 統計的サンプリング・アルゴリズムを使用して、アプリケーションの実行速度を大幅に低下させることなく必要な情報を取得します。

関連情報

ラーニングラボ – テクニカルビデオ、ホワイトペーパー、Webinar など

インテル® Parallel Studio XE 製品ページ – HOW TO ビデオ、入門ガイド、ドキュメント、製品の詳細情報、サポートなど

評価ガイド – さまざまな機能の使用法を紹介する評価ガイド

30 日間の評価版のダウンロード

購入方法: 言語別のスイート

インテル® Parallel Studio XE には、開発のニーズに応じて 3 つのエディションがあります。Composer Edition と Professional Edition では、C++ または Fortran のいずれかの言語で利用できます。

- **Composer Edition:** 高速な並列コードを構築するためのコンパイラー、パフォーマンス・ライブラリー、並列モデルが含まれています。
- **Professional Edition:** Composer Edition の機能に加えて、高速な並列コードの設計、ビルド、デバッグ、チューニング用にパフォーマンス・プロファイラー、スレッド設計/プロトタイピング・ツール、メモリー/スレッドデバッガーが含まれています。
- **Cluster Edition:** Professional Edition の機能に加えて、MPI を含む高速な並列コードの設計、ビルド、デバッグ、チューニング用に MPI クラスター通信ライブラリー (MPI エラーチェックおよびチューニング・ユーティリティー付き) が含まれています。

	インテル® Parallel Studio XE Composer Edition[1]	インテル® Parallel Studio XE Professional Edition[1]	インテル® Parallel Studio XE Cluster Edition
インテル® C++ コンパイラー	√	√	√
インテル® Fortran コンパイラー	√	√	√
インテル® TBB (C++ のみ)	√	√	√
インテル® IPP (C++ のみ)	√	√	√
インテル® MKL	√	√	√
インテル® Cilk™ Plus (C++ のみ)	√	√	√
インテルによる OpenMP* 実装	√	√	√
ローグウェーブ IMSL* ライブラリー[2] (Fortran のみ)	バンドルおよびアドオン	アドオン	アドオン
インテル® Advisor XE		√	√
インテル® Inspector XE		√	√
インテル® VTune™ Amplifier XE[3]		√	√
インテル® MPI ライブラリー[3]			√
インテル® Trace Analyzer & Collector			√
オペレーティング・システム (開発環境)	Windows* (Visual Studio*) Linux* (GNU*) OS X*[4] (XCode*)	Windows* (Visual Studio*) Linux* (GNU*)	Windows* (Visual Studio*) Linux* (GNU*)

注:
1. C++ または Fortran のいずれか、あるいは両言語で利用できます。
2. Windows* Fortran スイートのアドオンまたは Composer Edition のバンドルとして利用できます。
3. スイートのバンドルまたはスタンドアロンとして利用できます。
4. OS X* の言語スイートとして利用できます。

インテル® Parallel Studio XE の詳細:

- 以下の Web サイトをご覧ください。 http://intel.ly/parallel-studio-xe
- あるいは、左の QR コードをスキャンしてください。

30 日間の評価版:

- http://intel.ly/sw-tools-eval の Web サイトで、「Product Suites」をクリックしてください。

著作権と商標について

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に関する詳細は、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

JPN/1501/PDF/XL/SSG/KS boost-performance-studioxe-evalguide/Rev-082014