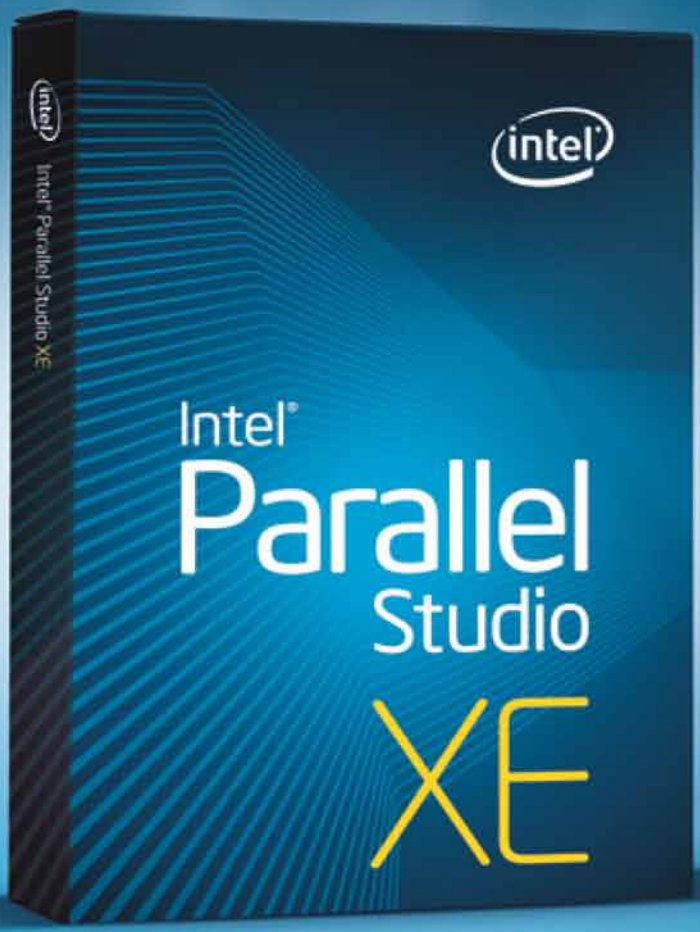
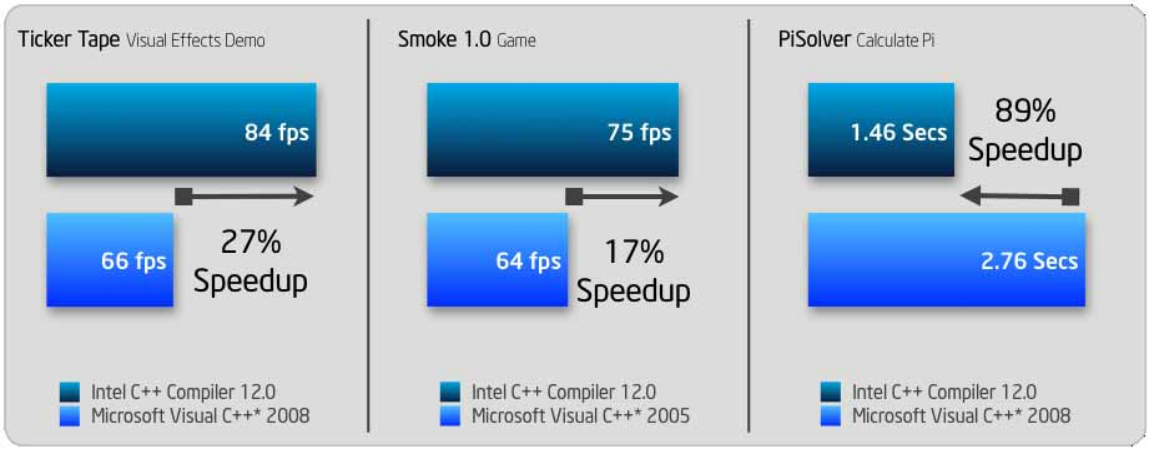
スレッド化されていない アプリケーションでも 大幅なパフォーマンス向上を 容易に実現

インテル® Parallel Studio XE Windows* 版



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。



アプリケーションの詳細 ›

ビデオを見る ›

アプリケーションの詳細 ›

ビデオを見る ›

詳細は後述のセクションを参照

† Microsoft* Visual Studio* 2008 (Ticker Tape および PiSolver); Microsoft* Visual Studio* 2005 (Smoke)

** インテル® C++ Composer XE Update 1

システム環境: Ticker Tape および Smoke: インテル® Core™ i7 プロセッサー (4 コア) 3.20GHz、3GB RAM、NVIDIA GeForce 9800 GX2、Windows Vista* Ultimate SP2。
PiSolver: インテル® Core™i7 プロセッサー (4 コア)、1.6GHz、4GB RAM、Microsoft* Windows* 7

パフォーマンス向上のための 2 つのステップ

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の測定

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。まずは 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 回にできることもあります。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

多くの場合、インテル® C++ Composer XE の最適化コンパイラーで 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。幸いなことに、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: インテル® Parallel Studio XE のインストールと設定

推定所要時間: 15-30 分

1 インテル® Parallel Studio XE の評価版をダウンロードします。

2. **parallel_studio_xe_2011_setup.exe** をクリックしてインテル® Parallel Studio XE をインストールします (システムにより異なりますが、約 15-30 分かかります)。

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 Sample.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。

このサンプルは、Microsoft* Visual Studio* 2005 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。Microsoft* Visual Studio 2008* および Microsoft* Visual Studio 2010* 向けにソリューション・ファイルも含まれています。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。

2. **PiSolver.zip** ファイルをシステムの書き込み可能なフォルダー (例えば、**マイ ドキュメント\Visual Studio 20xx\Intel\samples フォルダー**) に展開します。

サンプルのビルド:

1. 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。

   a. 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 (PiSolver.zip を展開したフォルダー) に移動します。図 1

   b. 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。**[Build (ビルド)] > [Configuration Manager (構成マネージャ)]** を選択して、**[Active solution configuration (アクティブ ソリューション構成)]** ドロップダウン・ボックスから **Release** 設定を選択し、**[Configuration Manager (構成マネージャ)]** を閉じます。図 2

   c. **[Build (ビルド)] > [Build Solution (ソリューションのビルド)]** を選択して、ソリューションをビルドします。図 3

図 1

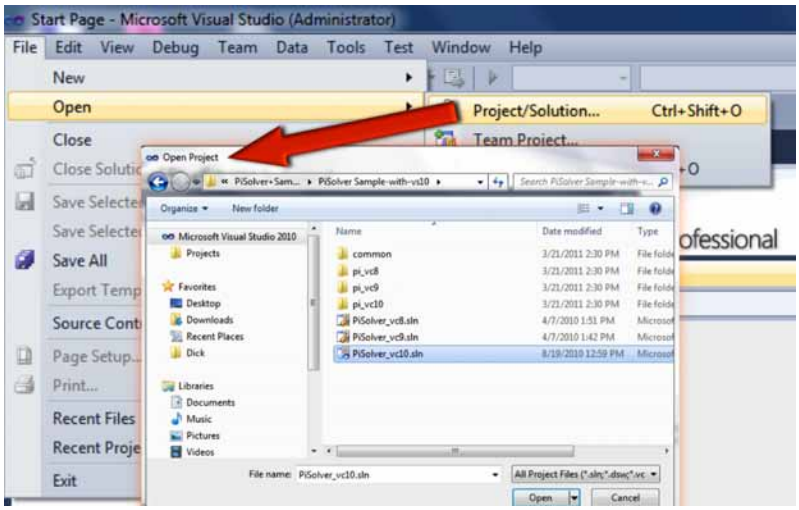


図 2

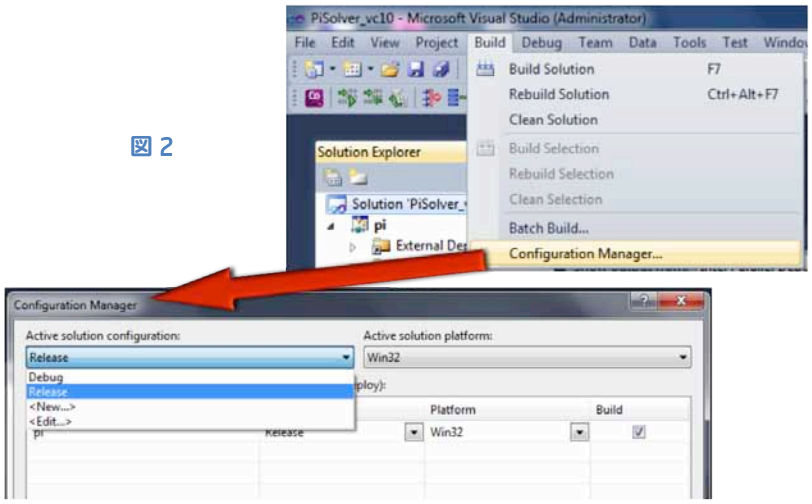
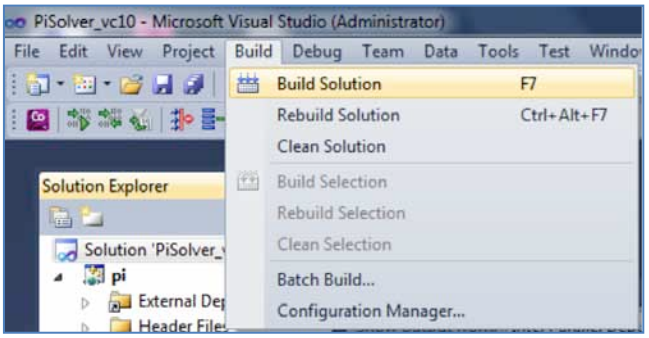


図 3



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

e. [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

ステップ 3: インテル® VTune™ Amplifier XE 2011: の実行 hotspot の特定

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。

   a. [Solution Explorer (ソリューション エクスプローラ)] ウィンドウで Pi を右クリックして、Pi プロジェクトをハイライトします。

   b. **[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。

   c. [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。

   d. [C/C++] を展開して、[General (全般)] をクリックします。

   e. [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

   f. [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 4

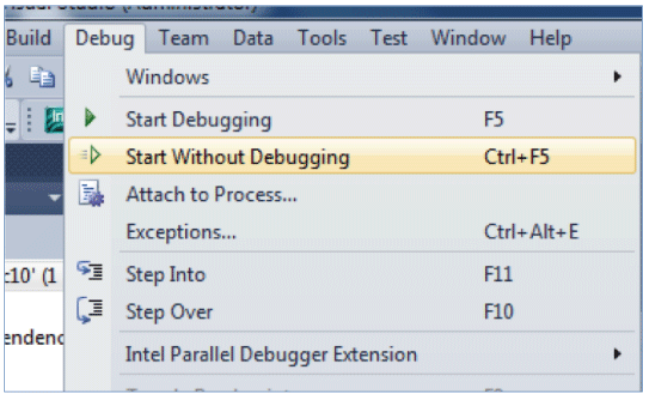


図 5

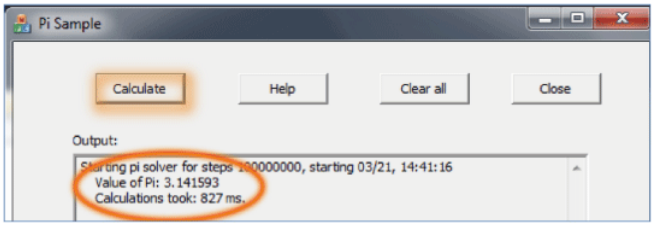


図 6

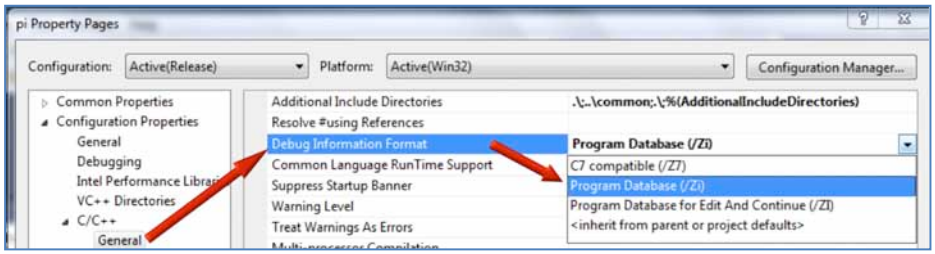
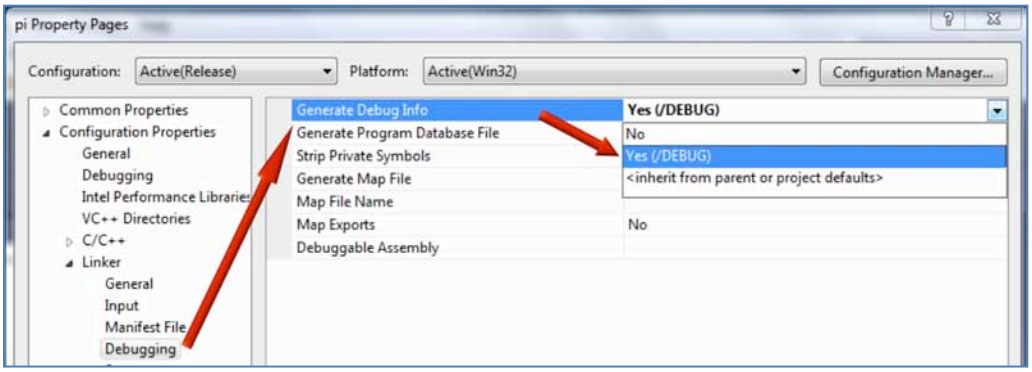


図 7



2. インテル® VTune™ Amplifier XE ツールバーの **[New Analysis (新しい解析)]** から **[Hotspots]** を選択します。図 8
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 9 のような hotspot レポートが表示されます。(hotspot の解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閉じます。)
5. [Bottom-up (ボトムアップ)] 関数リストの 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、最も時間を費やしている (piGetSolutions) の CalcPi のコールスタックをダブルクリックして、hotspot 関数 CalcPi を呼んでいるソースコードを表示します。
6. 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。

図 8

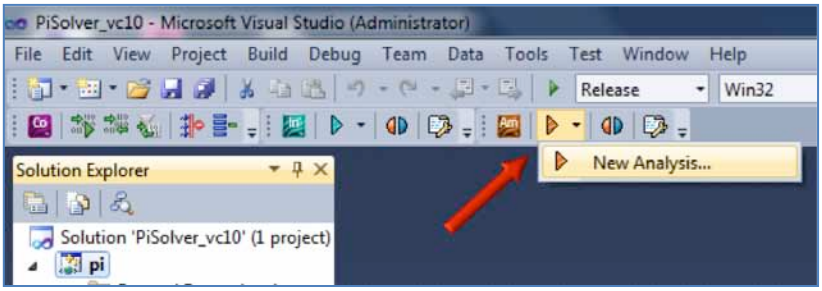
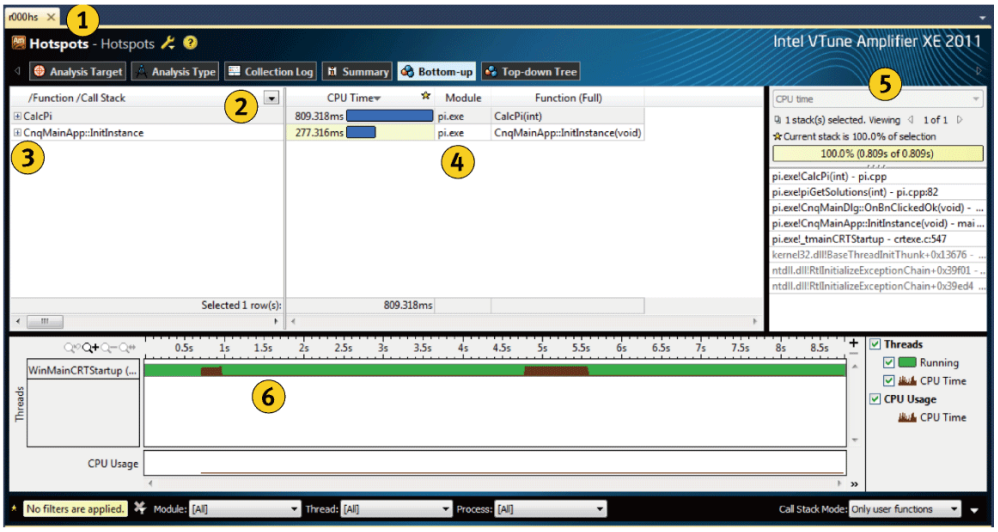


図 9



1. hotspot 解析からの結果。結果が収集されるたびに番号 (000) が上がります。
2. [Function - Caller Function Tree (関数 - 呼び出し元関数ツリー)] 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。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 解析結果の [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. [Timeline (タイムライン)] ビューはスレッド間の CPU アクティビティーの変化を表示します。

ステップ 4: インテル® C++ Composer XE に含まれているインテル® C++ コンパイラーを使用してコンパイル

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. **[Project (プロジェクト)] > [Intel C++ Composer XE 2011 (インテル(R) C++ Composer XE 2011)] > [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。

Microsoft* Visual Studio* 2005 および Microsoft* Visual Studio* 2008 に関するノート:コンパイルに時間のかかる大規模なアプリケーションでは、**[Do not clean project(s) (プロジェクトをクリーンしない)]** チェックボックスをオンにすることもできます。PiSolver の例では、オンにする必要はありません。

これで、Microsoft* コンパイラーの代わりにインテル® C++ コンパイラーを使用する新しいプロジェクト設定が作成されました。次に、選択したファイルにのみインテル® C++ コンパイラーを使用し、その他のファイルには Microsoft* コンパイラーを使用するように設定を変更します。

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* 2005 および Microsoft* Visual Studio* 2008 ユーザー:

a. **[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)]** で、Microsoft* Visual C++* コンパイラー (cl.exe) を使用するようにコンパイラーと環境の設定を変更します。図 10

b. [Confirmation (確認)] ボックスが表示されたら、[Continue (続行)] をクリックします。その後、[Apply (適用)]、そして [OK] をクリックします。これで、インテル® Parallel Composer XE 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* 2010 ユーザー:

a. **[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 11

b. [Apply (適用)]、そして [OK] をクリックします。これで、インテル® C++ Composer XE 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。

図 10

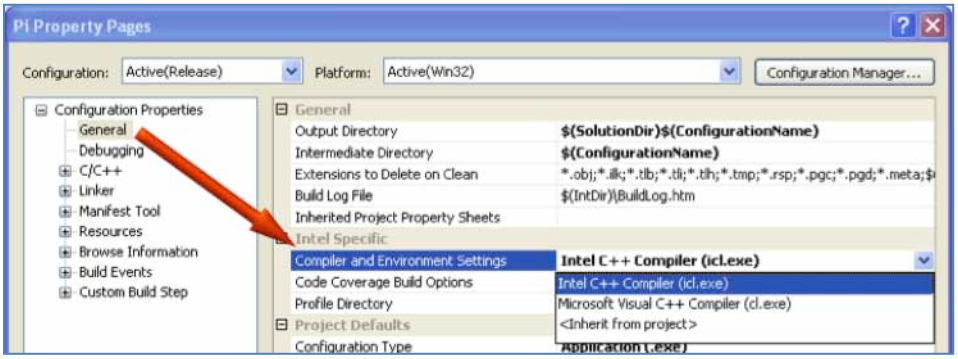
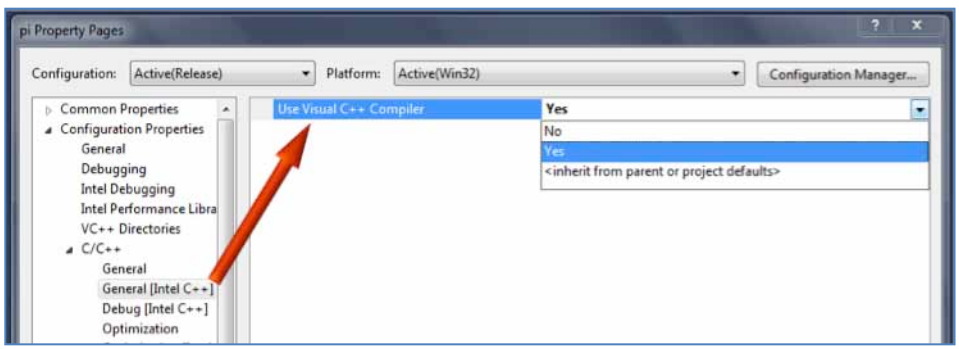


図 11



6. **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。

   a. Microsoft* Visual Studio* 2005 および Microsoft* Visual Studio* 2008 ユーザー: 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)] > [General (全般)]** で、インテル® C++ コンパイラー (icl.exe) を使用するようにコンパイラーと環境の設定を変更します。図 12

[Apply (適用)]、そして **[OK]** をクリックします。

   b. Microsoft* Visual Studio* 2010 ユーザー: **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 13

[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。

7. **Pi** プロジェクトをクリックしてハイライトし、**[Build (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。

8. **[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。

図 12

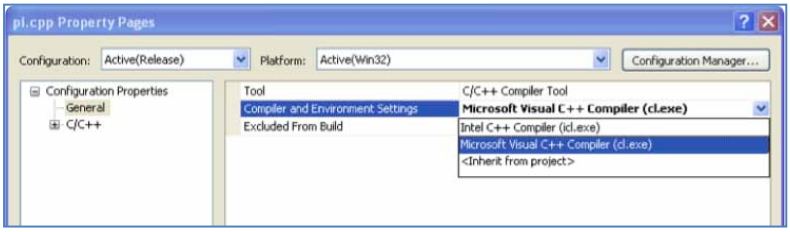
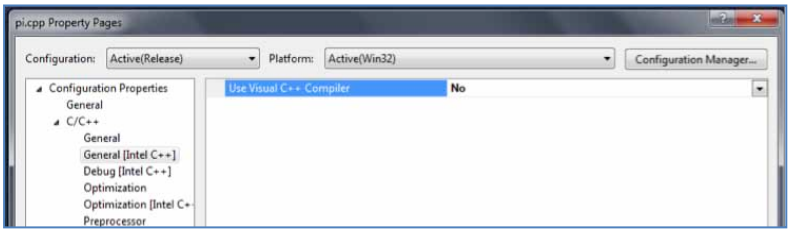


図 13



結果

テストシステムでは、インテル® C++ Composer 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
導入前 [1]	0.827 秒
導入後 [2]	0.421 秒
速度向上	96%

システム環境:インテル® Core™ i7 プロセッサー (4 コア)、1.6GHz、4GB RAM、Microsoft* Windows 7
[1] Microsoft* Visual Studio* 2010
[2] インテル® C++ Composer XE 2011 Update 1

この例では、インテル® C++ Composer XE の最適化コンパイラー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® C++ Composer XE 2011 にはインテル® 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) も含まれています。インテル® IPP は、デジタルメディアおよびデータ処理アプリケーション向けに高度に最適化された、ソフトウェア関数の広範囲なマルチコア対応ライブラリーです。インテル® IPP は、よく使用される基本的なアルゴリズムを含む、最適化された関数を多数提供します。

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 ツールバーの **[New Analysis (新しい解析)]** ボタンを再度クリックして、**[Quick Hotspot Analysis (クイック Hotspot 解析)]** を選択します。
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 14
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。インテル® VTune™ Amplifier XE ツールバーにある [Compare (比較)] ボタンをクリックすると、図 15 のような比較画面が表示されます。詳細レポートは関数ごとの変更箇所を示します。以下の詳細レポートでは、CalcPi の時間が大幅に短縮されたことが分かります。

図 14

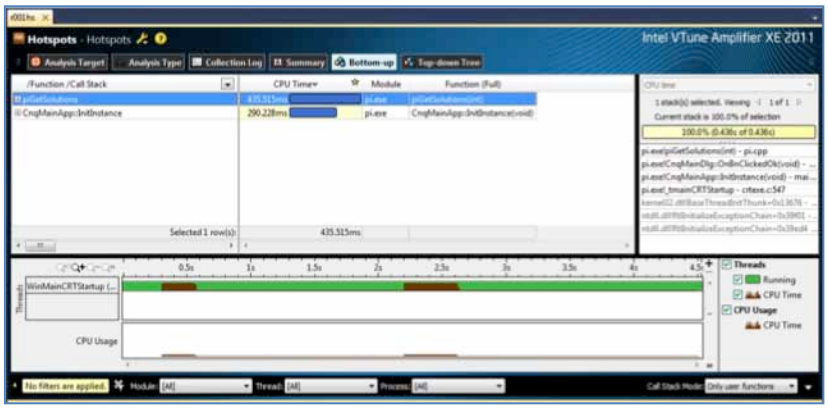
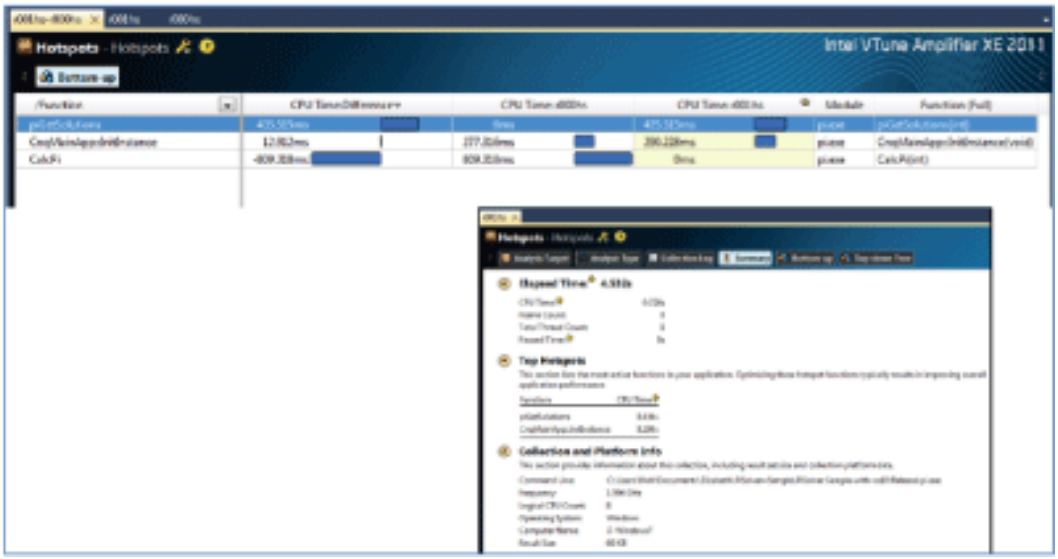


図 15

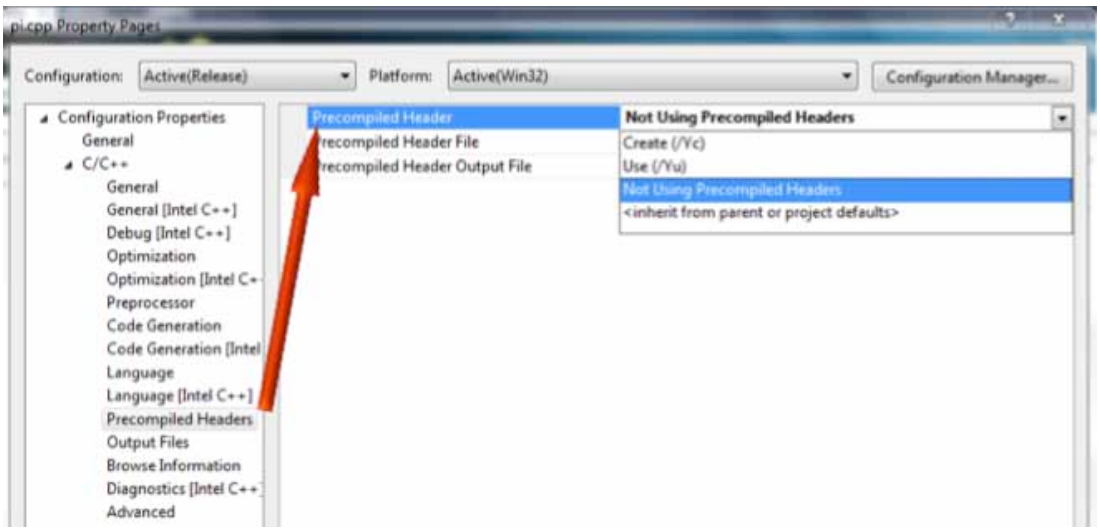


大規模で複雑なアプリケーションの場合のヒント

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++ Composer 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++ Composer XE に含まれているインテル® 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 16

図 16



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 を使用して解析する実行ファイル。

重要な概念

hotspot 解析: 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。

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。

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 の再生など

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

最適化に関する注意事項

インテル® コンパイラー、関連ライブラリーおよび関連開発ツールには、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーで利用可能な命令セット (SIMD 命令セットなど) 向けの最適化オプションが含まれているか、あるいはオプションを利用している可能性がありますが、両者では結果が異なります。また、インテル® コンパイラー用の特定のコンパイラー・オプション (インテル® マイクロアーキテクチャーに非固有のオプションを含む) 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。これらのコンパイラー・オプションと関連する命令セットおよび特定のマイクロプロセッサーの詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・オプション」を参照してください。インテル® コンパイラー製品のライブラリー・ルーチンの多くは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。インテル® コンパイラー製品のライブラリー・ルーチンの多くは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。インテル® コンパイラー製品のコンパイラーとライブラリーは、選択されたオプション、コード、およびその他の要因に基づいてインテル製マイクロプロセッサーおよび互換マイクロプロセッサー向けに最適化されますが、インテル製マイクロプロセッサーにおいてより優れたパフォーマンスが得られる傾向にあります。

インテル® コンパイラー、関連ライブラリーおよび関連開発ツールは、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。

インテルでは、インテル® コンパイラーおよびライブラリーがインテル製マイクロプロセッサーおよび互換マイクロプロセッサーにおいて、優れたパフォーマンスを引き出すのに役立つ選択肢であると信じておりますが、お客様の要件に最適なコンパイラーを選択いただくよう、他のコンパイラーの評価を行うことを推奨しています。インテルでは、あらゆるコンパイラーやライブラリーで優れたパフォーマンスが引き出され、お客様のビジネスの成功のお役に立ちたいと願っております。お気づきの点がございましたら、お知らせください。

改訂 #20110307