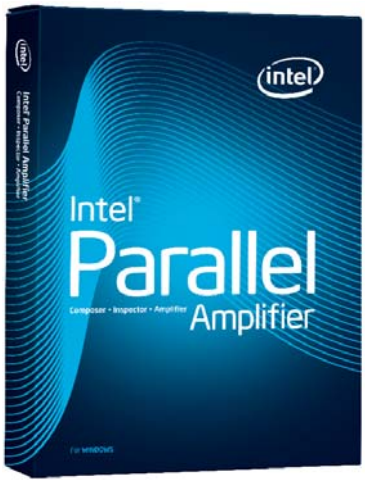
インテル® Parallel Composer

製品紹介

インテル® Parallel Composer



マルチコアシステム向け C/C++ シリアル/並列アプリケーションのビルド

インテル® Parallel Composer は、Windows* ベースのクライアント・アプリケーションへ並列化の導入を行う開発者のためのインテル® C++ コンパイラー、ライブラリー、デバッグ機能を備えた包括的なセットです。Microsoft* Visual Studio* に統合し、Visual C++ と互換性を保ち、開発者の作業をサポートします。IDE の投資を保護しつつ、並列デバッグ機能を含む、今までに例のない幅広い並列化開発機能を提供します。インテル® Parallel Composer は単体製品として、あるいはスレッド化エラーとメモリーエラーを分析するインテル® Parallel Inspector、および並列アプリケーションのパフォーマンスを分析するインテル® Parallel Amplifier も同梱されたインテル® Parallel Studio として購入することができます。

インテル® Parallel Composer のコンポーネント

- 32 ビット・プロセッサー用インテル® C++ コンパイラー、32 ビット・システムにおいて 64 ビット・アプリケーションを作成するクロスコンパイラー、ネイティブ 64 ビット・コンパイラー
- インテル® Parallel Debugger Extension。Microsoft Visual Studio デバッガーに統合されます。
- 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)。受賞歴もある C++ テンプレート・ライブラリーでスレッドをタスクに抽象化し、信頼性と移植性に富んだスケーラブルな並列アプリケーションを作成できます。また、Visual C++ とも併用できます。
- 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)。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高度に最適化されたソフトウェア関数を備えたマルチコア対応の広範囲なライブラリーです。インテル® IPP には、手動で最適化されるプリミティブ・レベルの関数と、コーデックなどのハイレベルなスレッド化ソリューションの両方が含まれています。Visual C++ 環境および .NET 環境でも使用できます。
- サンプルコードと入門ガイド。ツールをすぐに使用できるよう支援します。

インテル® C++ コンパイラー

Microsoft Visual Studio 統合と互換性

インテル® Parallel Studio の全機能が Microsoft Visual Studio 2005/2008 にシームレスに統合されます。インテル® Parallel Composer は、インテル® Parallel Studio の主要な 3 つの機能セットのうちの 1 つです。インテル® C++ コンパイラー、インテル® Parallel Debugger Extension、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® インテグレーテッド・パフォーマンス・プリミティブが含まれています。

このコンパイラーは、ネイティブ 32 ビット開発環境、クロスコンパイル環境 (32 ビット・ホストで 64 ビット・アプリケーションを作成)、ネイティブ 64 ビット開発環境を提供します。32 ビット環境のみ、または 64 ビット環境のみのインストール、あるいは両方のインストールを選択できます。

インテル® C++ コンパイラーおよびインテル® Parallel Debugger Extension は、C/C++ 開発者に多くの利点があります。インテル® Parallel Composer のほかのコンポーネントや、インテル® Parallel Studio をフルで使用する必要はありません。これは、インテル® スレッディング・ビルディング・ブロックとインテル® インテグレーテッド・パフォーマンス・プリミティブを Visual C++ コンパイラーで使用できることを意味します。また、インテル® Parallel Studio のメモリーリークや並列性チェック機能を、Visual C++ でビルドされたアプリケーションで使用することも可能です。つまり、互換性を持つインテル® C++ コンパイラーを含む、インテル® Parallel Studio のさまざまな活用方法を検討する多くの理由がここにあります。

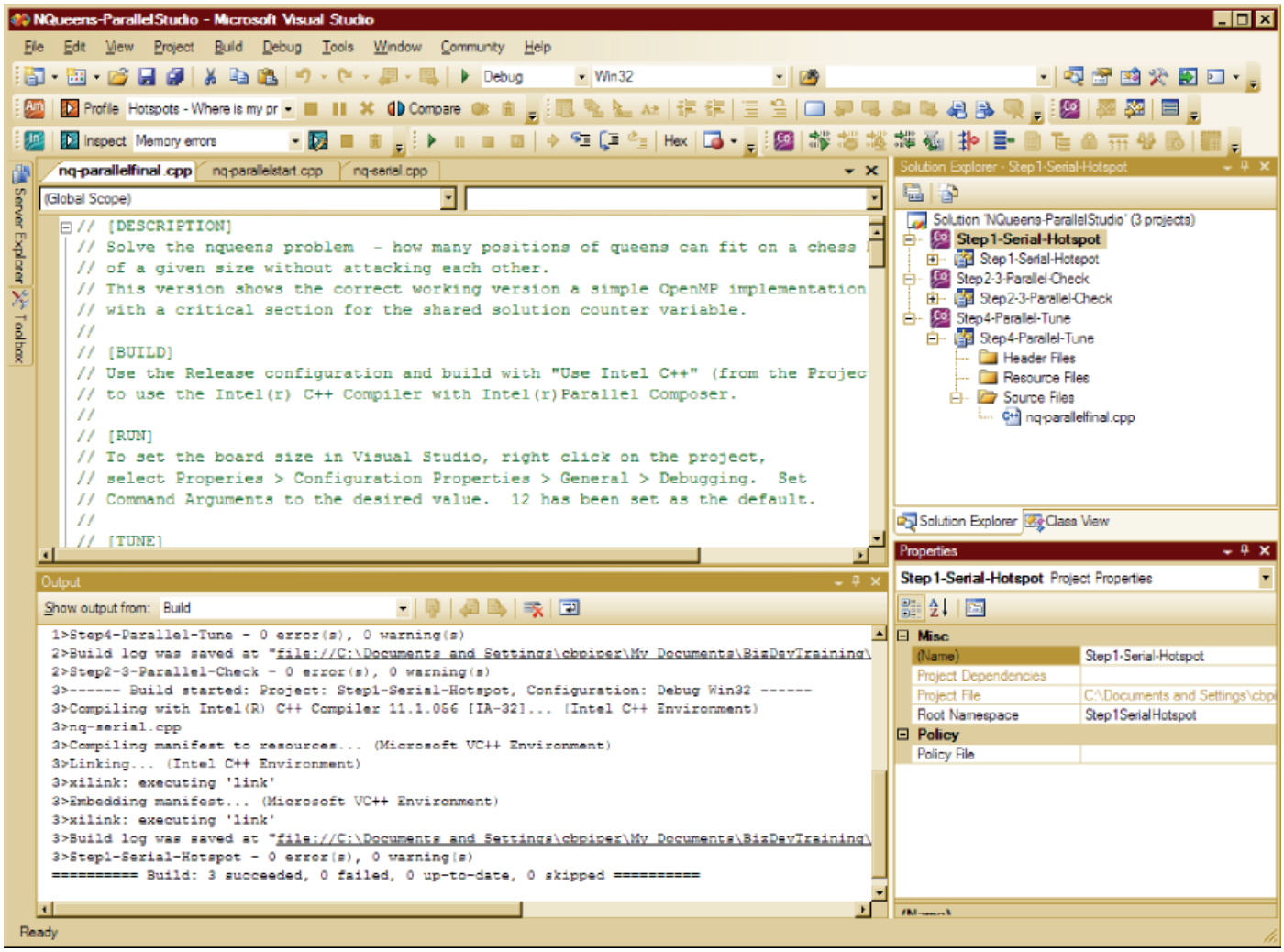


図 1: Visual Studio に統合されたインテル® Parallel Composer。画面のソリューションではインテル® C++ を使用しています。[プロジェクト] メニュー、あるいはソリューションやプロジェクト名をクリックして、 Visual C++ に簡単に切り替えることができます。

入門ガイドと拡大する並列化のコミュニティーとの交流

インテル® Parallel Composer は、機能のクイックツアーを含む入門ガイドや使用方法をステップごとに紹介するコードサンプルを提供しています。また、短編ビデオへのリンクもあり、インテル® Parallel Composer の並列機能を迷うことなく使用し始めることができます。インテル® Parallel Composer のユーザーからは、入門ガイドは非常に有効であるとの評判です。また、サンプルコードは並列化概念や手法の導入に役立ち、ツールの効率的な活用を導くという声も寄せられています。

入門ガイドは、いくつかの方法で提供されています。Web ページからや、Visual Studio の [ヘルプ] メニュー (詳細なドキュメントも提供)、Windows の [スタート] ボタンから表示されるインテル® Parallel Studio またはインテル® Parallel Composer のメニューからアクセスできます。

さらに、インストールの完了時に表示されるプロンプトからもアクセス可能です。並列化は初めての方も、プロの方も、入門ガイドにひととおり目を通すことをお勧めします。

ツールを使用し始めたら、開発者コミュニティーが役立つでしょう。インテルのマルチコア・プロセッサー・ベースのシステムを活用するために、拡大するコミュニティーに是非ご参加ください。インテルでは、開発者が活発にアイディアを出し合い、意見を投稿し、ポイントを獲得して、インテル® Black Belt Software Developer の称号を手にできるフォーラムを提供しています。また、開発者の関心を引くような並列化に関するさまざまなトピックを擁する大規模なナレッジベースも用意されています。コミュニティーへのご参加をお待ちしています。並列プログラミングおよびマルチコア・コミュニティーについては、http://software. intel.com/en-us/articles/intel-parallel-studio/ (英語) を参照してください。このページから、ブログ、ナレッジベース、ダウンロードなどを含むすべてのリソースにアクセスできます。お気に入りリストへの追加もお忘れずに!

ラムダ関数のサポート

インテル® コンパイラーは、C++ の次期標準、C++0x の改訂案をサポートするラムダ関数を実装する最初の C++ コンパイラーです。ラムダ構造は、C++ の関数オブジェクトあるいは C の関数ポインターとほぼ同じです。クロージャーとともに使用して、コードとスコープを組み合わせることで強力な概念を表します。クロージャーは、関数定義を原文どおり含むバインディング形式によって確立されるバインディングの値を参照、変更することのできる関数です。つまり、ラムダ関数とクロージャーはセットで、関数オブジェクトと関数ポインターまわりの構文糖とみなすことができ、関数オブジェクト、ラムダを使いたいときにすぐに記述できる便利な方法を提供します。

次の図 2 のソースコードは、ラムダ式で作成された関数オブジェクトの例です。C++ とインテル® TBB の統合をより密にすることで、ラムダ関数とクロージャーを使用して、コードをパラメーターとして渡し、ファンクター operator() の概念を簡単に表すことができます。

```
void solve() {
  parallel _ for(blocked _ range<size _ t>(0,size, 1),
  [](const blocked _ range<int> &r){
    for (int i = r.begin(); i != r.end(); ++i)
    setQueen(new int[size], 0, (int)i);
    };
  });
}
```

図 2: ラムダ関数のソースコード例

単純並列関数

インテル® Parallel Studio のインテル® C++ コンパイラーでは、新しい 4 つのキーワード (__taskcomplete、__task、__par、__critical) が提供され、簡単に並列プログラミングを行えるようになっています。このキーワードにより、開発アプリケーションが並列化の利点を享受できるようにするには、/Qopenmp コンパイラー・オプションを指定してから、再コンパイルします。これにより、実際の並列化のレベルを管理する適切なランタイム・サポート・ライブラリーがリンクされます。この新しいキーワードは、OpenMP* 3.0 ランタイム・ライブラリーを使用して、並列化を行いますが、OpenMP プラグマと宣言子構文で実際に並列化を指定する必要はありません。そのため、コードはより自然に C または C++ で記述されます。

上記のキーワードは、文のプリフィックスです。たとえば、関数の solve()、using _ _par を並列化することができます。引数間のオーバーラップがないと想定して、solve() 関数を _ _par キーワードを追加して変更します。関数を呼び出す方法には変更がなく、計算処理が並列化されます。例を図 3 に示します。

```
void solve() {
        _ _ par for(int i=0; i<size; i++) {
        // 最初の行のすべての位置を試行します
        // 各再帰で個別の配列を作成します
        // ここから開始です
        setQueen(new int[size], 0, i);
        }
}
```

図 3: __par の例。4 つの単純並列関数のうちの 1 つで、インテル® Parallel Studio のインテル® C++ コンパイラーに新しく追加された関数。

OpenMP 3.0

OpenMP は、移植性を備えたマルチスレッド・アプリケーション開発のための業界スタンダードです。細粒度 (ループレベル) および粗粒度 (関数レベル) のスレッド化に効果的です。OpenMP 3.0 では、宣言子アプローチを使用して、データ並列化とタスク並列化の両方をサポートするようになりました。このアプローチは、シリアル・アプリケーションを並列アプリケーションに変換する簡単で強力な方法を提供し、マルチコアシステムおよび対称型マルチプロセッサー・システムの並列実行から、潜在的な大きなパフォーマンス・ゲインを引き出すことを可能にします。

OpenMP で記述され、ビルドされたアプリケーションは、1 つのプロセッサーのみで実行されている場合、結果はソースコードを変更していない場合と同じです。つまり、結果は変更されていない、シリアル実行コードから得られる結果と同じです。

これにより、シリアルの一貫性を維持しながら、段階的にコードを変更していくことが簡単になります。宣言子のみがコードに挿入されるため、段階的なコードの変更を可能にしつつ、1 つのプロセッサーしか搭載していないシステムでも動作するソフトウェアの共通のコードベースを維持することができます。

OpenMP は、複数のプラットフォームとオペレーティング・システムをサポートするシングル・ソースコード・ソリューションです。また、OpenMP ランタイムにより適切なコア数が選択されるため、コア数をアプリケーションに「ハードコード」する必要はありません。

OpenMP 3.0 のタスク・キューイング

不規則なパターンの動的データや再帰など複雑な制御構造体を持つプログラムでは、効率的に並列化するのは困難です。OpenMP 3.0 のワークキューイング・モデルを使用すると、OpenMP 2.0 や 2.5 ではできなかった不規則な並列化が使用できるようになります。

task プラグマは、囲まれた作業 (タスク) 単位が実行される環境を指定します。task プラグマに到達すると、タスクブロック内のコードは、そのタスクに関連付けられている概念キューにキューイングされます。シーケンシャルなセマンティクスを保持するために、タスク完了時に暗黙的なバリアがあります。

開発者は、タスクブロック間、あるいはタスクブロックのコードとそのタスクブロック外の task ブロックのコード間で、依存性が存在しないこと、または適切に同期されることを確認する必要があります。この例は、上の図 4 に示されています。

```
#pragma omp parallel
#pragma omp single
{
 for(int i=0; i<size; i++) {
    // 最初の行のすべての位置を試行します
    // 各再帰で個別の配列を作成します
    // ここから開始です
#pragma omp task
    setQueen(new int[size], 0, i);
  }
}
```

図 4: OpenMP 3.0 のタスク・キューイングのサンプル

上の図 4 の例では、1 つのタスク・キューしか必要ありません。そのため、1 つのスレッド (omp single) のみを起動して、キューを設定する必要があります。setQueens は、それぞれが独立しているため、タスクの概念に合致します。また、次に示す「インテル® Parallel Debugger Extension」も参照してください。OpenMP プログラムでのタスク、チーム、ロック、バリア、タスク待機の状態を専用のウィンドウで簡単に調査することができます。

インテル® Parallel Debugger Extension

インテル® Parallel Composer には、インテル® Parallel Debugger Extension が含まれています。インストール後に、Visual Studio の [デバッグ]メニューからアクセスできます (下の 図 5 を参照)。

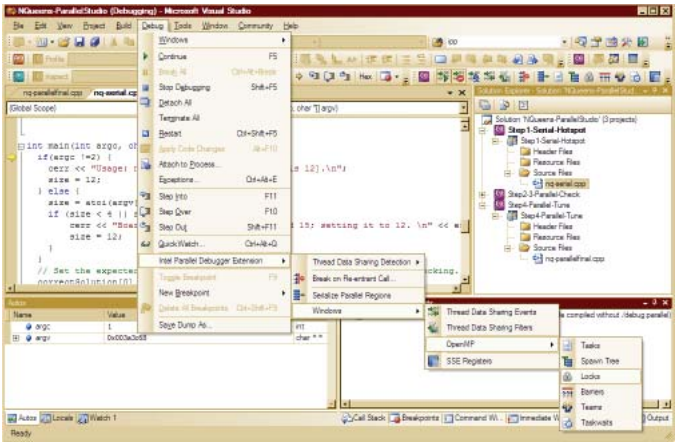


図 5: Microsoft Visual Studio の [デバッグ] プルダウンメニューから利用可能なインテル® Parallel Debugger Extension

インテル® Parallel Debugger Extension は、並列アプリケーションの共有データやデータの依存関係について、詳しい洞察やアクセスを提供します。開発サイクルを短縮し、重大なランタイムエラーを引き起こす可能性のある潜在的なデータアクセス競合を早めに検出します。インテル® Parallel Composer をインストールし、Visual Studio を起動した後、開発アプリケーションが SIMD (Single Instruction Multiple Data) 実行を活用しているとき、実行フローについてさらに洞察を得たいとき、並列化アプリケーションが OpenMP スレッディングを使用している場合に可能性のあるランタイム競合があるときなど、インテル® Parallel Debug Extension を使用できます。

共有データイベント検出、関数の再帰性検出、並列化コードのシリアル実行を含む OpenMP 認識などのインテル® Parallel Debugger Extension の高度な機能を活用するには、インテル® コンパイラーでデバッグ情報をインストルメントする /debug:parallel オプションを使用してコードをコンパイルしてください。

詳細は、インテル® Parallel Debugger Extension のホワイトペーパー (http://software.intel.com/en-us/articles/parallel-debugger-extension/ (英語)) を参照してください。このホワイトペーパーでは、デバッガー拡張の詳細と利点をさらに詳しく説明し、また機能をより良く活用するための方法を示しています。

並列アプリケーションをデバッグする製品をお探しの場合は、インテル® Parallel Studio を是非ご検討ください。この製品には、メモリーリーク分析ツールとスレッド検証ツールを備えたインテル® Parallel Inspector が含まれています。また、hotspot (パフォーマンス・ボトルネック) 分析や、追加された並列化コードとデータ認識機能でコードの正当性をデバッグする並列性チェックツール、インテル® Parallel Amplifier も同梱されています。インテル® Parallel Studio は、インテル® Parallel Debugger Extension を含む、これらすべての機能を提供しています。

驚異的並列ループの最適化

独立した反復を持つデータ並列化を示すアルゴリズムは、「驚異的並列」コードを示すループに役立ちます。インテル® Parallel Composer は、そのようなループのパフォーマンスを最小限の労力で、最大限に引き出す 3 つの手法 (自動ベクトル化、インテルにより最適化された valarray コンテナーの使用、自動並列化) をサポートしています。インテル® Parallel Composer は自動ベクトル化に適したループを自動で検出します。これには、静的配列または動的配列を持つ明示的な for ループ、ベクトルコンテナーと valarray コンテナー、または明示的なループを持つユーザー定義の C++ クラスなどがあります。例外として、暗黙的な valarray ループは、自動ベクトル化を行うか、あるいは最適化されたインテル® IPP ライブラリー・プリミティブを起動するよう指定することが可能です。自動ベクトル化と最適化された valarray ヘッダーの使用で、アプリケーションのパフォーマンスを最適化し、ストリーミング SIMD 拡張命令対応のプロセッサーを十分に活用することができます。

インテルにより最適化された valarray ヘッダーの使用方法を次に説明します。まず最初に、図 6 を見てみます。この図は、明示的な valarray、ベクトルループ、暗黙的な valarray ループの例を示しています。

```
valarray<float> vf(size), vfr(size);
vector<float> vecf(size), vecfr(size);


//対数関数、ベクトル、明示的なループ
(int j = 0; j < size-1; j++) {
        vecfr[j]=log(vecf[j]);

}

//対数関数、valarray、明示的なループ
(int j = 0; j < size-1; j++) {
        vfr[j]=log(vf[j]);
}

//対数関数、valarray、 暗黙的なループ vfr=log(vf);
```

図 6: 明示的な valarray、ベクトルループ、暗黙的な valarray ループ

最適化された valarray ヘッダーを使用するには、[Build Component Selection (ビルド・コンポーネントの選択)] ダイアログで、インテグレーテッド・パフォーマンス・プリミティブを指定し、コマンドライン・オプションを設定する必要があります。これを行うには、最初にプロジェクトを Visual Studio にロードして、プロジェクト・プロパティーのポップアップ・ウィンドウを表示します。
図 7 で示すように、[Additional Options (追加のオプション)] ボックスで “/Quse-intel-optimized-headers” を追加して、[OK] をクリックします。

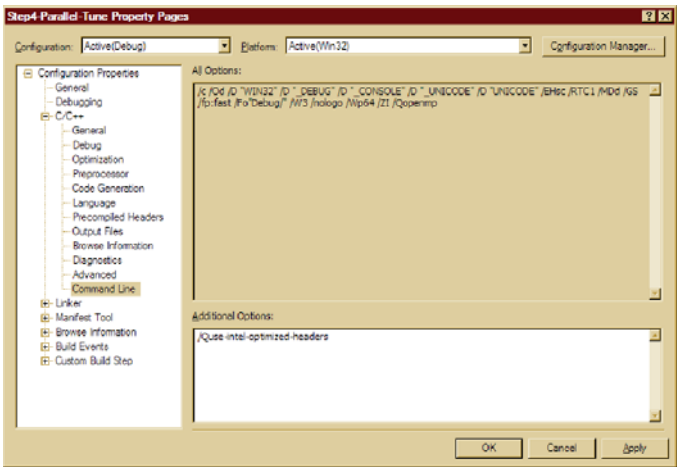


図 7: 最適化ヘッダーファイルを使用するコマンドを Visual C++ のコマンドラインに追加

次に、[プロジェクト] メニューから、[Build Component Selection (ビルド・コンポーネントの選択)] ポップアップを表示し、[Intel Integrated Performance Primitives (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)] の右のボックスで、[Common (一般)] を選択し、[OK] をクリックします。図 8 は、この画面を示しています。この操作を終えたら、アプリケーションをリビルドして、アプリケーションを変更した場合と同じようにパフォーマンスと動作をチェックします。

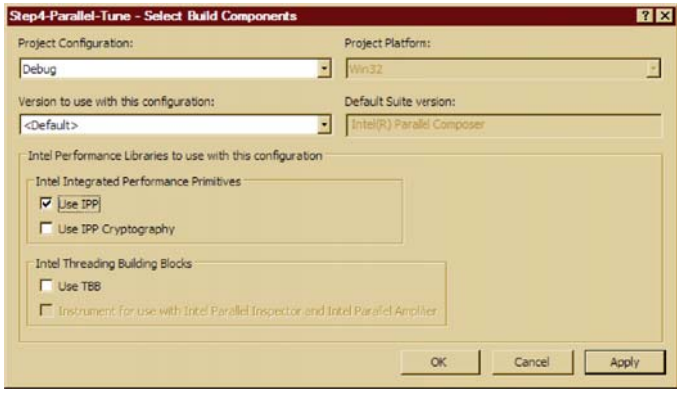


図 8: Visual Studio でインテル® IPP を使用する

自動並列化では、並列で安全に実行できる並列ループを検出し、自動でマルチスレッド・コードを生成して、マルチコア・プロセッサーを活用することで、アプリケーション・パフォーマンスが向上します。反復のパーティショニング、データの共有、スレッドのスケジューリング、および同期化に関する下位の詳細レベルを処理して、ユーザーの負担を軽減します。

自動並列化は、自動ベクトル化と最適化された valarray ヘッダーの使用を補い、SSE 対応のマルチコアシステム上で最適なパフォーマンスを得られるようにします。マルチスレッド・アプリケーションのサポートに関する詳細は、ユーザーガイド (http://software.intel.com/en-us/intel-parallel-composer/ (英語) でドキュメントのリンクをクリック) を参照してください。

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® スレッディング・ビルディング・ブロック (インテル® TBB) は、受賞歴もある C++ テンプレート・ライブラリーで、スレッドをタスクに抽象化し、信頼性と移植性に富んだスケーラブルな並列アプリケーションを作成できます。インテル® Parallel Composer に含まれるインテル® TBB は、インテル® C++ コンパイラーまたは Microsoft Visual C++ で使用できる標準テンプレート・ライブラリー (STL) です。インテル® TBB は、次の並列プログラミングの 3 つの主要な課題を解決します。

- **生産性:** 並列化の実装を簡素化
- **正当性**: 並列の同期問題の排除を支援
- **保守**: 現在だけではなく、将来にも対応できるアプリケーションの作成を支援

インテル® TBB の利点:

- **将来も安心のアプリケーション**: インテル® TBB の洗練されたタスク・スケジューラーを使用することで、コア数 (およびスレッド数) が増加するにつれ、アプリケーションが高速化します。
- **移植性**: 並列化を一旦実装してしまえば、複数のプラットフォームでスレッド化コードを実行できます。
- **相互運用**: 多種多様なスレッド化手法、ハードウェア、オペレーティング・システムで動作します。
- **活発なオープンソース・コミュニティー**: インテル® TBB には、オープンソース版もあります。opentbb.org (英語) は、フォーラム、ブログ、コードサンプルなどが用意されている役立つ Web サイトです。

インテル® TBB には、包括的な、並列化のための抽象化されたテンプレート、コンテナー、クラスが用意されています。図 9 は、インテル® TBB 内の主な関数グループを示しています。

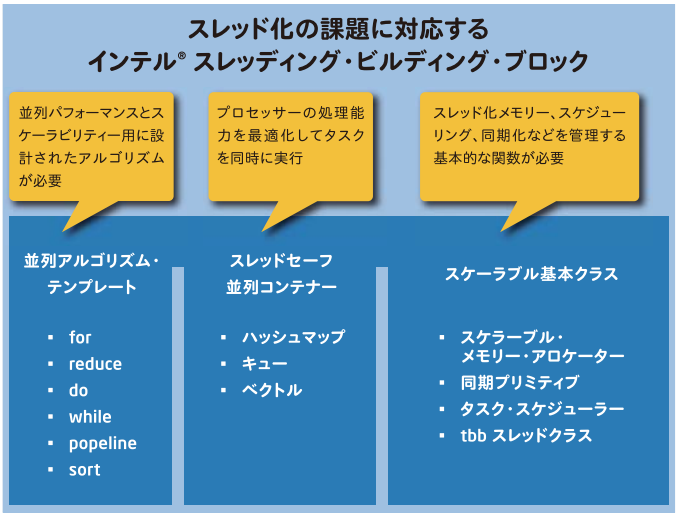


図 9: インテル® TBB の主な関数グループ

問題	解決方法
並列化を簡単に追加する方法	インテル® TBB の parallel_for コマンド ▪ 単刀直入な for/next ループの置換で、並列化の利点を活用 ▪ 決まった回数の独立したループ反復による負荷分散された並列実行
最良のスケーラビリティーを得るためのスレッドの管理	インテル® TBB タスク・スケジューラー ▪ スレッドプールを管理し、ネイティブスレッドの複雑性を隠避 ▪ 並列プログラミングにおける一般的なパフォーマンスの問題に対応するよう設計 - オーバーサブスクリプション: ハードウェア・スレッドごとに 1 つのスケジューラー・スレッド - 高いオーバーヘッド:プログラマーはスレッドではなくタスクを指定 - ロード・インバランス: ワークスチールにより負荷を調整
並列環境におけるボトルネックのメモリー割り当て	インテル® TBB は、スレッドごとのメモリー管理アルゴリズムに基づいて、テスト、チューニングされたスケーラブルなメモリー・アロケーターを提供 ▪ STL テンプレート・クラスへのアロケーター引数として ▪ malloc/realloc/free 呼び出し (C プログラムの) の代替として ▪ グローバルな new 演算子と delete 演算子の代替として (C++ プログラム)

図 10: 並列化の 3 つの主要な問題に対応するインテル® TBB

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)

インテル® Parallel Composer には、インテル® IPP 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。ビデオ・コーディング、信号処理、オーディオ・コーディング、画像処理、音声コーディング、JPEG コーディング、音声認識、コンピューター・ビジョン、データ圧縮、イメージカラー変換、暗号化、ストリング処理/正規表現、ベクトル/行列の演算などの分野で頻繁に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。

インテル® IPP には、手動で最適化されるプリミティブ・レベルの関数とコーデックなどのハイレベルなスレッド化サンプルの両方が含まれ、Visual C++ 環境と .NET 環境の両方で使用することが可能です。これらの関数やサンプルはすべてスレッドセーフで、多くは内部的にスレッド化され、今日のマルチコア・プロセッサーを活用し、将来のメニーコア・プロセッサーにスケーリングします。

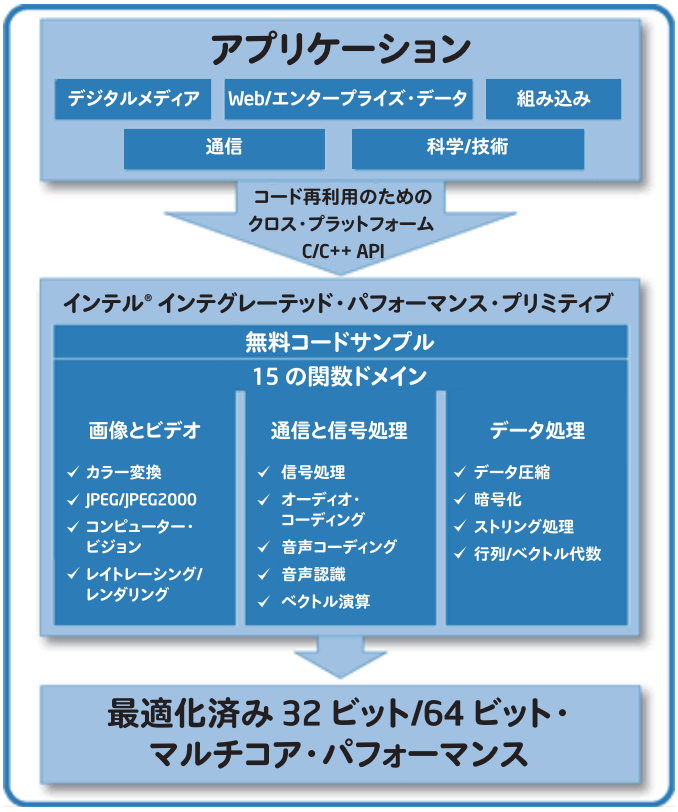


図 11: インテル® IPP は、インテル® Parallel Studio のコンポーネント、インテル® Parallel Composer に含まれ、先にリストされた多様なドメインにおけるスレッド化済みのスレッドセーフなライブラリー関数を提供します。

インテル® IPP のパフォーマンス

アプリケーションと作業負荷に応じて、インテル® IPP 関数は、同等のコンパイル済み C コードと比べ、何倍も高速に動作します。次に示す画像のサイズ変更の例では、コンパイル済み C++ コードで実行した場合は 338 マイクロ秒かかり、インテル® IPP の画像処理関数を使用して実行すると、111 マイクロ秒しかかからなかったことを示しています。パフォーマンスが 300% ほど向上しています。

図 12: この画像サイズの変更例 (256x256 ビットから 460x332 ビット) では、インテル® IPP を使用したアプリケーションが 111 マイクロ秒であるのに対し、コンパイル済み C++ コードでは 338 マイクロ秒を要しています (システム構成: インテル® Xeon® プロセッサー、2.9 GHz、2 プロセッサー、4 コア/プロセッサー、2 スレッド/プロセッサー)

Visual Studio でのインテル® IPP の使用

インテル® IPP サポートを Microsoft Visual Studio プロジェクトに簡単に追加できます。インテル® Parallel Composer には、インテル® IPP ライブラリーや Visual Studio プロジェクトへのパスを追加するメニューとダイアログが用意されています。[ソリューション エクスプローラ] でプロジェクト名をクリックし、[Intel® Build Components Selection (インテル® ビルド・コンポーネントの選択)] を選択して、[Build Component Selection (ビルド・コンポーネントの選択)] ダイアログでインテル® IPP を追加します。そして、ヘッダーコードと機能コードを含むインテル® IPP コードをプロジェクトに追加するだけです。ビルドの選択ダイアログで、IPP 用に自動的にライブラリー名がリンカーに追加され、インテル® IPP ライブラリーへのパスが追加されます。

C++ プロジェクトのほか、インクルードされたラッパークラスを使用し、ストリング処理、イメージ処理、信号処理、カラー変換、暗号化、データ圧縮、JPEG、行列、ベクトル演算のドメインにおいて、C# からインテル® IPP 関数への呼び出しをサポートすることで、C# プロジェクトでも使用できます。

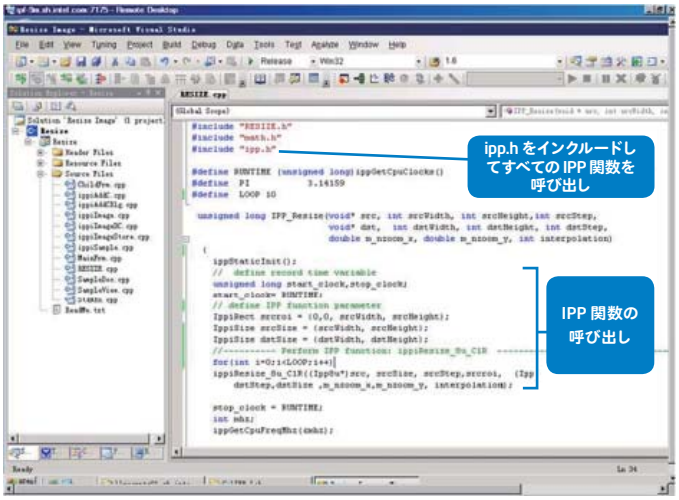


図 13: インテル® IPP ライブラリー呼び出しを Visual Studio コードに簡単に統合

機能

- Microsoft Visual Studio を C/C++ 並列化対応にシームレスにアップグレード。Visual Studio に統合し、IDE 投資を保護しながら、並列化機能を追加できます。
- Microsoft デバッガーと統合するインテル® Parallel Debugger Extension。並列化の問題を発見し、対処できるよう Visual Studio を強化します。アプリケーションの使用までの時間が短縮されます。
- 簡単な並列関数、並列データ配列、そして多数のスレッド化ライブラリー関数が同梱。タスクのスレッド化を容易にし、アプリケーション開発を高速化します。
- 自動並列化オプションや自動ベクトル化オプションにより、開発を簡単にし、時間を節約
- 統合された配列表記、データ並列のインテル® IPP 関数でオーディオ、ビデオ、信号分析やその他のアプリケーション・クラスを高速処理
- 並列アプリケーションの実装に最も効率的で、マルチコア・プラットフォームの潜在能力を引き出すインテル® TBB が同梱
- 並列化をすばやく導入できるコードサンプルを含む広範囲なドキュメントを提供。短時間で習得可能な入門ガイドも用意されています。
- コミュニティー・サポート。拡大する開発者コミュニティーに是非ご参加ください。多くの開発者がコードの並列化に取り組んでいます。ほかの開発者の経験から学び、そしてご自身の知識や経験を共有してください。特典も用意されています。

システム要件

- Microsoft Visual Studio
- 最新のシステム要件については、次の Web ページを参照してください。

www.intel.com/software/products/systemrequirements/ (英語)

サポート

インテル® Parallel Studio 製品には、コミュニティー・フォーラムおよびテクニカルノート、アプリケーション・ノート、ドキュメント、すべての製品アップデートを含む、技術サポートに関する情報を掲載したナレッジベースへのアクセスが提供されます。

詳細は、http://software.intel.com/en-us/articles/intel-parallel-studio/ (英語) を参照してください。

評価版のダウンロード

評価版のダウンロードについての詳細は、次の Web サイトを参照してください。

http://www.intel.co.jp/jp/software/products/intel-parallel-studio-home/

インテル® Parallel Studio

今日のシリアル・アプリケーション、そして明日のソフトウェア・イノベーションのための設計

インテルは、シリアル・アプリケーションおよびマルチコアやメニーコア向けの新しい並列アプリケーション用に設計された生産性ソリューション・セットにより、Microsoft Visual Studio C++ 開発者に簡素化された並列化を提供します。

インテル® Parallel Studio: 究極のオールインワン並列化ツールキットで最適化されたシリアル・アプリケーションと並列アプリケーションを作成

インテル® Parallel Composer: C/C++ コンパイラーと高度なスレッド化ライブラリーで効率的なアプリケーションを開発

インテル® Parallel Inspector: 並列メモリーエラーとスレッド化エラーを未然に防ぐ検証機能でアプリケーションの信頼性を確保

インテル® Parallel Amplifier: スケーラブルなマルチコア・パフォーマンスのためのボトルネックの迅速な検出と並列アプリケーションのチューニング

JPN/0909/PDF/XL/SSG/TT 321553-001JA