The Parallel Universe

intel Software

Knights Corner から Knights Landing へ :

次世代のインテル® Xeon Phi™ プロセッサー / コプロセッサーに備える

OpenMP* の非効率性の特定と修正

HPC バイオメディカル研究シミュレーションの最適化

Issue 20
2015

目次

編集者からのメッセージ 3
James Reinders

注目記事

次世代のインテル® Xeon Phi ™ プロセッサー / コプロセッサー、Knights Landing への道のり 4
今年登場予定の、高度にスケーラブルな次世代のインテル® Xeon Phi™ プロセッサー / コプロセッサー、Knights Landing (開発コード名) 向けにアプリケーションを準備しましょう。

インテル® VTune™ Amplifier XE による OpenMP* 領域の解析 16
インテル® VTune™ Amplifier XE は、OpenMP* ユーザーがチューニングの労力をかけるべき場所をより簡単に特定できるように支援します。

ウォーカー分子動力学研究室のバイオメディカル・ソフトウェアの最適化 28
ウォーカー分子動力学研究室では、インテル® VTune™ Amplifier とほかのインテル® ソフトウェア開発製品を使用してインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーの両方でソフトウェアのパフォーマンスを最適化しました。

インテル® ソフトウェア開発製品が HPCwire アワードを獲得 36
2 つのインテル® ソフトウェア開発製品が読者部門と編集者部門の最優秀賞を獲得しました。

© 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 Xeon Phi、VTune、Xeon 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『High Performance Programming Pearls』の共著者で、このほかに、『Multithreading for Visual Effects』(2014)、『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』(2007、日本語 :『インテル スレッディング・ビルディング・ブロック ーマルチコア時代の 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』(2005) などの文献を発表しています。

準備はいいですか？

「我々を制限しているのはテクノロジーではありません。我々が制限しているのです。テクノロジーは我々の知能と創造性を表しています。つまり、テクノロジーの制限は我々の制限を表しているのです。」
–SF 作家、ポッドキャスター、テクノロジー・コメンテーター、Christian Cantrell 氏

ソフトウェア開発者の皆さんは、この言葉が意味することをよくご存じでしょう。今年は、制限を解決するための、待望の新製品が登場します。

インテルは、これまで数年間、次世代のインテル® Xeon Phi™ プロセッサー / コプロセッサー (開発コード名 : Knights Landing) の開発に取り組んできました。そしてついに、出荷の準備が整い、間もなく開発者の皆さんが利用できるようになります。このテクノロジーを使用する最初の商用システムはすでに発表されており、今年後半に市場に登場する見込みです。

「次世代のインテル® Xeon Phi™ プロセッサー / コプロセッサー、Knights Landing への道のり」では、次世代のインテル® Xeon Phi™ コプロセッサー向けにアプリケーションを準備するために、今できることを紹介します。現在のインテル® Xeon Phi™ コプロセッサー (開発コード名 : Knights Corner) を効率的に利用するプログラムは、将来のテクノロジーのために用意されたいくつかの手法を用いてパフォーマンスを向上できます。

「インテル® VTune™ Amplifier XE による OpenMP* 領域の解析」では、インテル® VTune™ Amplifier XE を利用して OpenMP* 開発者がチューニングの労力をかけるべき場所をより簡単に特定する方法を説明します。

「ウォーカー分子動力学研究室のバイオメディカル・ソフトウェアの最適化」では、科学研究者が強力な Amber* 分子動力学ソフトウェアを使用し、どのように実験を次のレベルへ進めたかを説明します。科学者がインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーでより長いシミュレーションを実行して研究を高速化できるように、研究室ではインテル® VTune™ Amplifier、インテル® MPI ライブラリー、インテル® マス･カーネル･ライブラリー (インテル® MKL) の最新機能を利用してソフトウェアのチューニングを行いました。

Parallel Universe では、開発者の皆さんに役立つさまざまな情報を今後も取り上げていきます。どうぞご期待ください。

James Reinders
2015 年 2 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。

次世代のインテル® Xeon Phi™ プロセッサー / コプロセッサー、Knights Landing への道のり

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

2012 年にインテルの最初のメニーコア商用製品、インテル® Xeon Phi™ コプロセッサー (開発コード名 : Knights Corner) を搭載した最初のシステムが登場しました。現在、インテル® Xeon® プロセッサーとこれらの第一世代インテル® Xeon Phi™ コプロセッサーは、ほかの計算方式 (GPU アクセラレーターなど) よりも、TOP500 リストにおける FLOPS (1 秒あたりの浮動小数点演算数) の増加に貢献しています。天河二号 (2013 年 6 月から 4 期連続で TOP500 リストの 1 位) は、インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーを使用しています。

特に並列処理向けに開発された Knights Corner コプロセッサーは、最高 1.10GHz で動作する、最大 61 個の x86 コアを搭載しています。各コアは 4 つのハードウェア・スレッドを実行できます。つまり、コプロセッサーあたり 244 ものスレッドを並列に実行することが可能です。

コアあたりの動作周波数が 1.10GHz とインテル® Xeon® プロセッサーより低いのは、意図的にシングルスレッドのパフォーマンスよりも、非常に高度な並列処理を優先しているからです。高度な並列処理向けに設計されたプログラムは、Knights Corner コプロセッサーの性能を引き出し、プロセッサーの 2.2 倍のパフォーマンスと 4 倍の電力効率を達成します。2 テラ FLOPS の単精度演算または 1 テラ FLOPS の倍精度演算が可能です。

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。

つまり、デバイス全体のパフォーマンスはプロセッサーよりはるかに高くなります。ただし、そのパフォーマンスを引き出すには並列プログラムが必要になります。

Knights Landing: 次世代のインテル® Xeon Phi™ テクノロジー

インテルは、これまで数年間、次世代のインテル® Xeon Phi™ コプロセッサー (開発コード名 : Knights Landing) の開発に取り組んできました。Knights Landing を搭載する最初の商用システムはすでに発表されており、今年後半に市場に登場する見込みです。

Knights Landing の最初のプロジェクトは、米国エネルギー省のローレンス・バークレー国立研究所に設置された国立エネルギー研究科学計算センター (NERSC) の Cori* スーパーコンピューターでした。別の著名なプロジェクトは、クレイが 1 億 7400 万ドルで受注した、インテル® Xeon® プロセッサー (開発コード名 : Haswell) と Knights Landing コプロセッサーで構成される国家核安全保障局 (NNSA) の Trinity* スーパーコンピューターです。

「多くのソフトウェア開発者は、この素晴らしさに感銘を受け、アプリケーションの並列性と新しいレベルの計算能力をどのように利用できるか調査して、アプリケーションの改良に取り組んでいます。プログラムをインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーのどちらで実行する場合でも、作業の結果としてより高いパフォーマンスを得ることができます。」

新しい機能

Knights Landing は、インテル® Xeon Phi™ コプロセッサー・ファミリーに大幅な拡張をもたらします。次に例を示します。

- Knights Landing のコア数は Knights Corner デバイスと同程度ですが、パフォーマンスと電力効率がさらに向上した、インテルの 14nm プロセスで製造されます。(Knights Corner 製品ラインは 22nm プロセスで製造されています。)
- パフォーマンスは、第一世代のピーク・パフォーマンス 1 テラ FLOPS から単一ソケットノードあたり 3 テラ FLOPS 倍精度ピーク・パフォーマンスへと大幅に向上しました。
- Knights Landing は、スタンドアロンのブート可能なプロセッサー (ホスト OS を実行) またはコプロセッサー (PCIe* エンドポイント・デバイス) として動作します。プロセッサー・バージョンでは、Knights Corner のように別のホスト・プロセッサーを用意する必要がなくなりました。この汎用性には多くの意味があります。一例として、ノードまたはシステムのすべてのメモリーを、メニーコアやマルチコア処理デバイス間で分割するのではなく、Knights Landing 専用にできます。

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。

- デバイスの個々のコアでアウトオブオーダー実行をサポートすることにより、Knights Landing はパフォーマンスとプログラムの容易性、特にプログラムのシリアル領域の実行効率を向上します。
- 大きく、高帯域のオンパッケージ・メモリーは、高い帯域幅と大容量のメモリー空間を必要とするアプリケーションのパフォーマンスを向上します。オフパッケージ・メモリーを使用することもできます。非常に多くのメモリーを必要とするアプリケーションでは、パフォーマンスを向上するため、通常のプロセッサーのように、オンデバイス・メモリーとオフデバイス・メモリーの両方を利用します。
- 標準化され移植性がある MPI ライブラリーを使用するクラスター･アプリケーションの効率を高める、統合ファブリック・バージョンが利用できます。
- Knights Landing は、3 分の 1 のスペースで 5 倍の帯域 (DDR4 と比較して) を有します。電力効率は、ディスクリート･コプロセッサーの 1.25 倍、以前のコプロセッサーの 5 倍です。

Knights Landing は、インテル® AVX-512 仕様に基づいてベクトル命令を標準化します。Knights Landing はプロセッサーとしても利用できるため、命令セットをプロセッサーと完全に同じにできることは大きな利点です。(これは Knights Corner と Knights Landing で最も重要な違いです。テスト用に、Knights Landing で実装された インテル® AVX-512 命令をフルサポートするソフトウェア・エミュレーターを提供しています。)

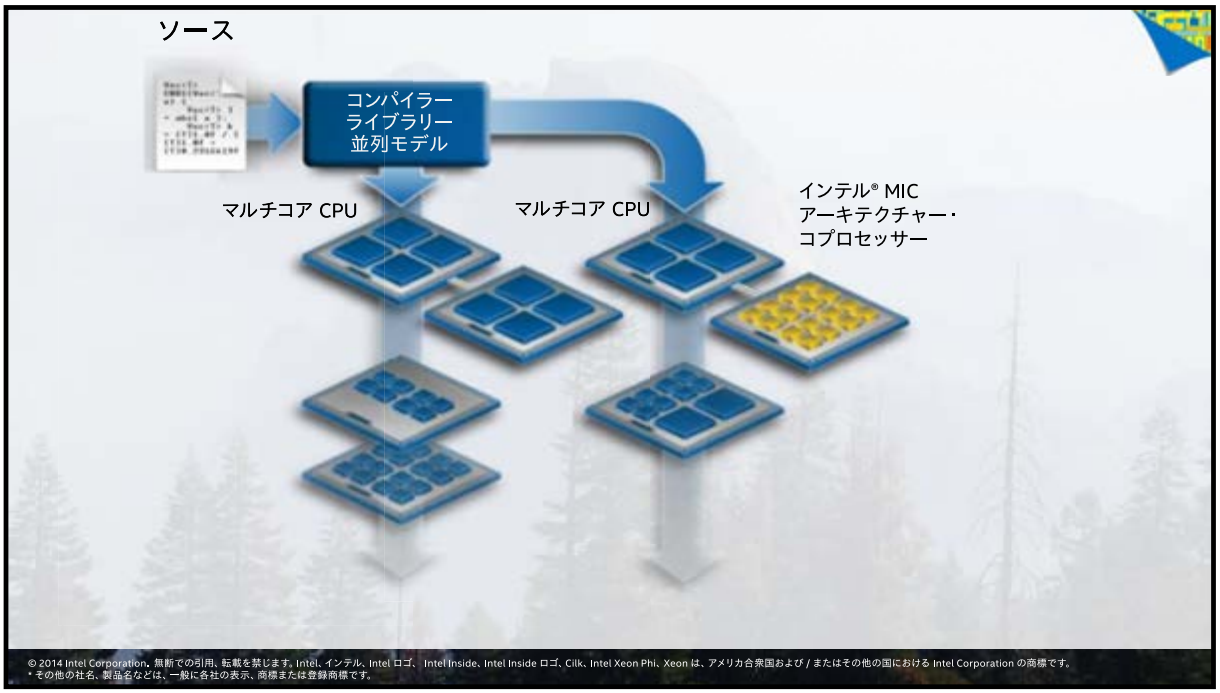
インテル® ソフトウェア製品のパフォーマンスおよび最適化に関する注意事項については、
software.intel.com/en-us/articles/optimization-notice/#opt-jp を参照してください。

さらに、 GNU* コンパイラー・コレクション (GCC*) の開発を担当する GNU* プロジェクトやほかのベンダーとも協力して作業を進めています。Knights Corner 向け GCC の現在のバージョンでは、 これらの 512 ビット・ベクトル命令を使用できませんが、Knights Landing のリリース時には使用できるように変更される予定です。

継続して利用できる機能 : Knights Landing の主なメリット

インテルのビジョンは、マルチコアからメニーコア･アーキテクチャーまで、一貫したプログラミング･モデル、言語、ツール、 手法を提供することであり、 プログラマーが (使い慣れたコンパイラー、 ライブラリー、 抽象化並列プログラミング・モデル、 標準規格を使用して) 1 つのプログラムを記述することで、 マルチコア CPU とインテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー、 つまり、 第 1 世代と第 2 世代インテル® Xeon Phi™ 製品の両方で最高のパフォーマンスを達成できるようにすることを目標としています。

高度な並列処理を実装するにはソースコードを修正する必要がありますが、 その投資は一貫した方法で行うことができます。 コア数の多い少ないにかかわらず、 同じ方法でコードを記述することができます **(図 1)**。



1 一貫したプログラミング・モデル、 言語、 ツール、 手法

このビジョンに基づき、インテル® Xeon Phi™ 製品は、OpenMP*/MPI、Fortran、C/C++、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) などの主要な標準規格をサポートしています。

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。

これらの標準規格は、長年にわたって利用されており、多くのベンダーからサポートされ、さまざまなプラットフォームで動作します。多くのアプリケーションが、これらの標準規格を使用して汎用デバイス (特にさまざまな種類の CPU) 向けに記述されています。Knights Landing は非常に広い用途向けに (CPU が行うあらゆることを行えるように) 設計されているため、これらの標準規格をすべてサポートしています。

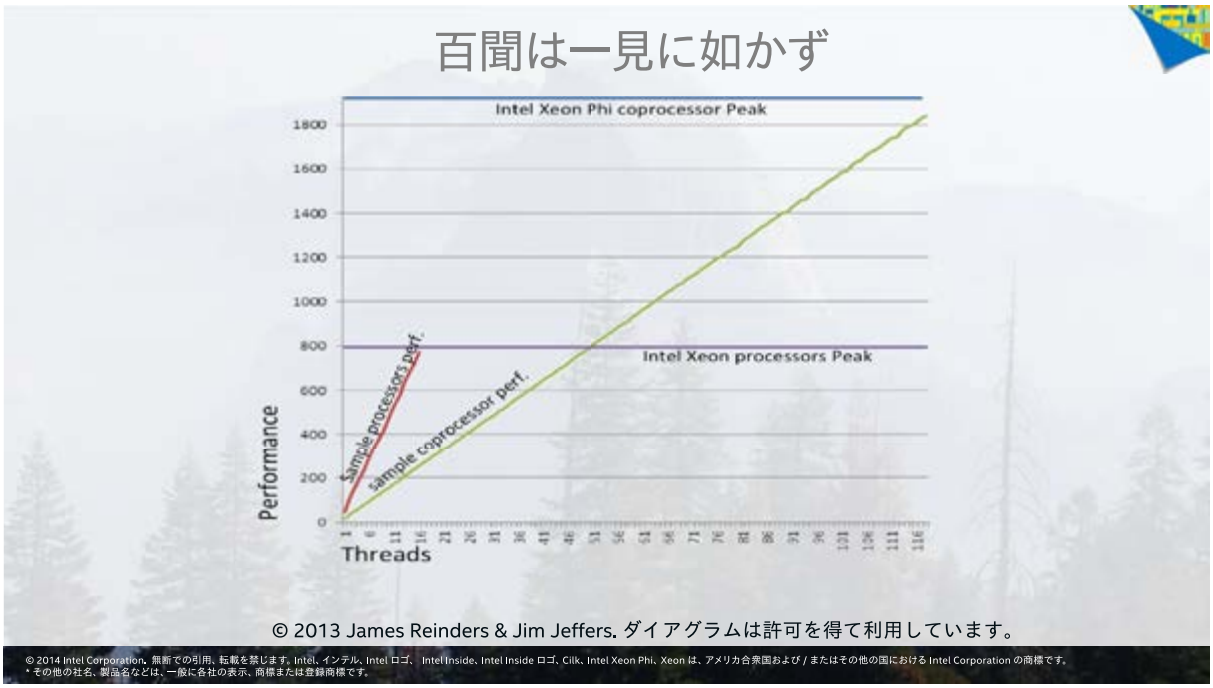
Knights Corner は、インテルのビジョンを実現しました。Knights Landing は、新しい革新技術を追加して、ビジョンをさらに高めます。

高度な並列処理プログラミング

Knights Landing ベースのシステムは 2015 年後半に登場しますが、その前にさまざまな準備を行うことができます。最も考慮すべき重要な概念は高い並列性の導入です。

ほかの高度な並列デバイスと同様に、個々のスレッドのパフォーマンスは CPU より低くなります。しかし、実行スレッドが数百になると、Knights Landing の全体的なパフォーマンスは CPU よりはるかに高くなります。優れたパフォーマンスを引き出すには、高度な並列処理を行う必要があります。

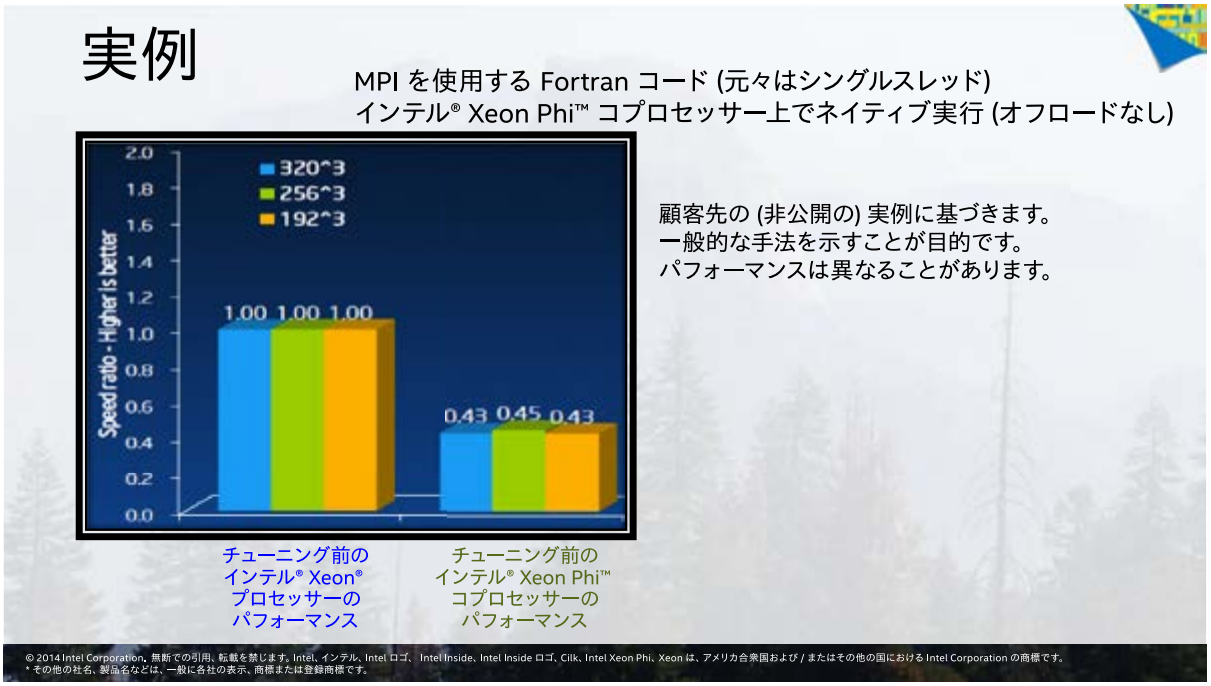
図 2 に示すように、プログラムが 4 つまたは 8 つの実行スレッドのみを使用する場合は、インテル® Xeon® プロセッサーのほうがパフォーマンスが高くなります。数十スレッドを使用すると、インテル® Xeon Phi™ コプロセッサーのパフォーマンスがインテル® Xeon® プロセッサーのパフォーマンスを上回ります。数百スレッドを使用した場合も、同様のパフォーマンスが期待できるでしょう。



2 スレッド数が少ないときはインテル® Xeon® プロセッサーのパフォーマンスのほうが高いが、スレッド数が多くなるとインテル® Xeon Phi™ コプロセッサーのパフォーマンスのほうがはるかに高くなる。この現実に沿って設計する。

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 3 は、実際にユーザーが MPI を使用して Fortran で記述したシングルスレッド・アプリケーションの相対パフォーマンスの例です。インテル® Xeon® プロセッサーのパフォーマンスが、インテル® Xeon Phi™ コプロセッサー (Knights Corner) のパフォーマンスより高くなっています。

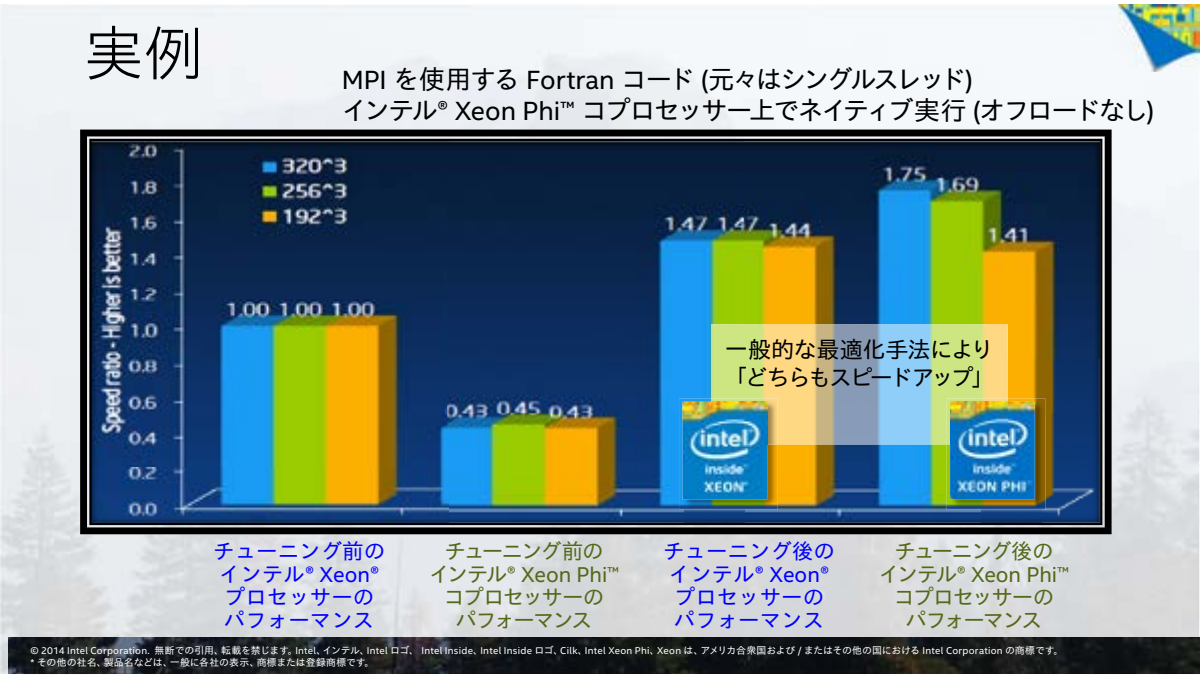


3 チューニングしていないアプリケーションの相対的なパフォーマンス

しかし、このプログラムを並列処理向けにチューニングすると、Knights Corner コプロセッサーのパフォーマンスは、インテル® Xeon® プロセッサーのパフォーマンスを大きく上回ります。インテル® Xeon Phi™ コプロセッサー向けに最適化を行った後の、インテル® Xeon® プロセッサーのパフォーマンスに注目してください **(図 4)**。パフォーマンスが大幅に向上しています。これは、あるユーザーと取り組んだ単純な最適化の例ですが、アプリケーションで並列化可能な領域を特定し、並列化するのにそれほど時間はかかりませんでした。

これらの結果からいくつかの疑問が生じます。なぜ数年前にこの作業を行わなかったのでしょうか。なぜプロセッサー・ベースのシステムでこのパフォーマンス向上が得られるようにアプリケーションを最適化しなかったのでしょうか。そして、なぜこの方法でチューニングされたスーパーコンピューターのワークロードが少ないのでしょうか。

おそらく、その答えは、61 コアを利用することが大きな動機付けになったからでしょう。いったんアプリケーションを移植して、61 コアを活用すれば、プロセッサーを使用するほとんどの並列プラットフォームでも利益が得られることに気付くでしょう。



4 並列処理用にチューニングした、同じアプリケーションの相対的なパフォーマンス

多くのソフトウェア開発者は、この素晴らしさに感銘を受け、アプリケーションの並列性と新しいレベルの計算能力をどのように利用できるか調査して、アプリケーションの改良に取り組んでいます。プログラムをインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーのどちらで実行する場合でも、作業の結果としてより高いパフォーマンスを得ることができます。

Knights Landing に対する準備の第一歩は、アプリケーションが Knights Corner のような高度な並列デバイスでスケーリングできることを認識することかもしれません。

幸いにも、現在利用可能な手法によって、より高いレベルの並列処理を簡単に達成できます。OpenMP* やインテル® TBB は非常に適した手法です。デバイスのスレッド数が非常に多い場合は、OpenMP* やインテル® TBB に加えて、デバイスで特定の MPI ランクを使用することもできます。これらの手法により、CPU よりもさらに上のパフォーマンスが期待できます。ただし、そのパフォーマンスを引き出すには並列処理が必須です。

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® Xeon Phi™ 製品が GPU でない理由

インテル® Xeon Phi™ 製品は GPU (グラフィックス・プロセシング・ユニット) アクセラレーターではありません。この決定は意図的に行われたもので、マルチコアからメニーコアまでさまざまなデバイスにおいて、一貫性のある CPU に限定されないプログラミング・モデルを採用しています。GPU として、あるいは GPU 手法を用いてインテル® Xeon Phi™ 製品のプログラムを行った場合、GPU プログラミングの制限により、製品のすべての機能を利用することはできません。

Knights Landing の準備はできていますか？

すでに準備できているケースもあるかもしれませんが、ほとんどのアプリケーションはまだ動作検証を行っていないでしょう。これまで高度な並列デバイスで動作していないアプリケーションは、おそらくスケーリングの妨げとなる要因が取り除かれていません。多くの場合、次の 3 つのスケーリング基準に注目する必要があります。

- スレッド・スケーリング。コア数の増加に伴いスケーリングします。
- ベクトル化スケーリング。データ並列化によりベクトル化が適切に使用されることを保証します。
- ファブリック・スケーリング (クラスター・アプリケーションの場合)。

キャッシュ関連の最適化もありますが、多くのスレッド・スケーリングおよびベクトル化手法は、優れたキャッシュ効率の概念を必要とし、採用しています。そのため、これらの 3 つの要素に注目することが最も適切です。

アプリケーションが高いスケーリングに対応しているかどうか判断する最適な方法は、Knights Corner やインテル® Xeon® EX プロセッサー・ベースのシステムのように、コア数の多いシステムでアプリケーションのパフォーマンスを評価することです。次のように、ワークロードのパフォーマンスと構成基準およびクラスター・ベースラインを決定します。

- スレッド・スケーラビリティーを調べるには、ノードのランク数に対するパフォーマンスを評価します。
- ベクトル・スケーラビリティーを調べるには、ベクトル化の有効性を評価します。ベクトル化のオン / オフを切り替えるか、最小のベクトル化と最大のベクトル化でパフォーマンスを比較します。
- ファブリック・スケーラビリティーを調べるには、ノードのランク数に対するパフォーマンスを評価します。ランク数を上げるとパフォーマンスは向上しますか？

パフォーマンスの評価は、より高いパフォーマンスを引き出すチューニングの可能性があるかどうかを理解する基本であり、非常に有効なステップです。

例えば、OpenMP* やインテル® TBB を使用すると、ランクを増やしてデバイス全体の使用率を上げることで、スレッド・スケーリングを徐々に増やすことができます。ワーカースレッドを追加するとパフォーマンスはどう変わるでしょう？Knights Corner では、コアあたりの最適なスレッド数は通常 3 つですが、この値はアプリケーションに依存します。そして、ピーク・パフォーマンスを達成するには、コアあたり少なくとも 2 つのスレッドが必要です。

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。

ベクトル化のスケーリングを実装する場合、多くの開発者は、組込み関数、自動ベクトル化、あるいは OpenMP* 4.0 標準規格に含まれる SIMD (Single Instruction Multiple Data) 宣言子を使用します。さらに、インテル® コンパイラーには、ベクトル化を支援する革新的なレポート機能も含まれています。

重要なことは、アプリケーションのこれら 3 つの特性を理解すれば、さまざまな機能を利用してすべての (例えば、インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーを使用する) マシンでパフォーマンスを向上できることです。

準備に最も適したシステムは？

現時点で Knights Landing の準備に最適なシステムが Knights Corner であることに疑いはありません。アプリケーションが Knights Corner に適している場合、Knights Corner で目標とするパフォーマンスが得られるように 3 つのスケーリング要素をチューニングして、Knights Landing 向けに準備します。

例えば、Knights Corner では、メモリー不足の CPU との間で頻繁にデータの移動を行っている場合、Knights Landing で実行するとデータの移動が少なくなりパフォーマンスが大幅に向上します。

「Knights Landing は、Knights Corner の組込み関数と非常に高いレベルのソース互換性を保ちながら、ベクトル命令のパフォーマンスを高める多くの拡張を追加しました。」

アプリケーションが Knights Corner に適していない可能性は？

十分考えられます。主な理由はメモリーです。16GB はカードとしては大容量ですが、ノードとしては大容量とは言えません。そのため、Knights Corner でアプリケーションが希望どおりに動作しない可能性があります。これは、カード向けに設計を行っているすべてのアプリケーションに当てはまる制限です。アプリケーションを制限するそのほかの要因は、入出力 (I/O) や通信要件の高さです。

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。

Knights Corner のシリアル・パフォーマンスはホスト・プロセッサーに比べ相対的に低いため、コードの重要な領域をより高速な (約 10 倍) ホスト・プロセッサーで実行するアプローチを取ることがあります。このアプローチでは、PCI バスを介してより多くのデータ移動が発生します。

Knights Corner が演算プラットフォームとして適切な選択肢でない場合は、コア数の多い別のシステムでアプリケーションの 3 つのスケーリング要素について再度考える必要がありますが、8 から 10 コアでは、極めて高度なスケーリング (スレッドとベクトルの効率的な利用) には対応できないため十分とは言えないでしょう。

理論的には、8 コアまたは 16 コアのシステムのパフォーマンス・データを基に数百コアのパフォーマンスを予測できます。実際には、スケーリングに十分な注意が払われないためか、この方法で正しく予測されたケースを見たことはありません。

実際のテスト環境でアプリケーションをテストすることは必須です。61 コアの Knights Corner システムを使用しない場合、予算的に問題なければ、コア数の多いインテル® Xeon® プロセッサー・ベースのシステム (インテル® Xeon® EX プロセッサー・ベースのシステム) を使用します。ただし、Knights Corner を使用できる場合は、Knights Corner を使用してください。コードを Knights Corner 対応にするために費やした労力は、インテル® Xeon® プロセッサー・ベースのマシンだけでなく、将来の Knights Landing でも必ず役立ちます。

将来に向けて

Knights Landing における拡張により、これまで説明した問題点と制限の多くは軽減もしくは解消されます。次に例を示します。

- 汎用性とメモリーの増加 (プロセッサーの能力とプロセッサー・サイズのメモリー) により、より多くのアプリケーションとモデルの適用が可能になります。
- 統合ファブリックとオンボードの高帯域メモリーにより、Knights Landing で扱うことができるワークロードの種別が広がります。
- 極めて高いシリアルコードのパフォーマンスにより、並列演算とシリアル演算の間でデータ移動 (および大規模なアプリケーションで移動を制御する処理) を行う必要がなくなります。

Knights Landing は、Knights Corner の組込み関数と高いレベルのソース互換性を保ちながら、ベクトル命令のパフォーマンスを高める多くの拡張機能を提供します。アライメントされていないデータアクセスの強化のような、インテル® AVX-512 仕様の拡張により、Knights Landing では Knights Corner よりも容易にベクトル化機能をプログラミングできます。(Knights Corner 向けのコードは、ここで多少の作業が必要でした。)

つまり、Knights Corner 向けのアプリケーションがすでにある場合、Knights Landing 向けのアプリケーションも用意できている、ということを忘れないでください。そして、Knights Corner で制約を受けているアプリケーションは、Knights Landing でその制約が軽減されます。

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。

BLOG HIGHLIGHTS

インテル® Xeon Phi™ コプロセッサー上での Lustre* のコンパイル、設定および実行

SUNNY G. » および DMITRY EREMIN »

この記事では、インテル® Xeon Phi™ コプロセッサーで Lustre* を動作するための推奨するステップを説明します。この記事のステップは、インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) バージョン 3.3.x/3.4.x および Lustre* バージョン 2.5.x/2.6.0 で検証しました。

MPSS 3.3.x および 3.4.x で Lustre* をコンパイルする

インテル ® Xeon Phi ™ コプロセッサーの Lustre* クライアントを設定する

インテル ® Xeon Phi ™ コプロセッサーの Lustre* クライアントをマウントする

トラブルシューティング

この記事の続きはこちらでご覧になれます。 >

61 コアの魅力

このテーマは繰り返し取り上げられています。61 コアを利用できるということは大きな動機付けになります。Knights Corner マシンを入手するか、非常にコア数の多いインテル® Xeon® プロセッサー・ベースのマシンを構築して、そのパフォーマンスを理解したのであれば、Knights Landing への道のりは順調と言えます。さらに多くのヒントを得たい方は、新しい書籍、『High Performance Parallelism Pearls』を一読ください。世界中の 69 人の専門家による、経験に基づく例と実際のコードを紹介しています。

以下に概要を説明します。

> **並列性を特定する。**インテル ® Xeon Phi ™ コプロセッサーを搭載したシステムでワークロードの真の特性を理解するには、さまざまな要素を考慮した優れた並列性が求められます。そのため、Knights Landing の準備をすることは、プラットフォームにかかわりなく、ツールとハードウェアを活用して並列性を特定することを意味します。

> **向上の機会を見つける。**多くのコアを使用し、異なる並列性の要素 (スレッド･スケーリング、ベクトル･スケーリング、ファブリック･スケーリング) を評価して、チューニングにより高度な並列処理を活かせるアプリケーション領域を把握します。

> **よく知られている実証された方法を使用する。**インテルは、一貫したプログラミング・モデル、言語、ツール、手法を使用した高度な並列コンピューティングを提供します。少数のコアから数百コアへの移行時に、新しい手法を導入したり、同じことを最初からやり直す必要はありません。

プログラムをインテル® Xeon® プロセッサー、Knights Corner、そして新しい Knights Landing のどの環境で実行する場合でも、これらの投資は高いパフォーマンスをもたらし、今後数世代にわたってその価値が損なわれることはありません。

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。

関連情報

Knights Corner: Your Path to Knights Landing James Reinders による Web セミナー (英語)

『Intel® Xeon Phi™ Coprocessor High-Performance Programming』(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**』) Jim Jeffers、 James Reinders、 2013 年

『High Performance Parallelism Pearls: Multicore and Many-Core Approaches』 Jim Jeffers、James Reinders、 2014 年

ブログ : **追加された AVX-512 命令** James Reinders、 2014 年 7 月 17 日

『Structured Parallel Programming』(日本語 :『**構造化並列プログラミング**』)、 Michael McCool、Arch Robinson、 2012 年

オンライン・デベロッパー・リソース : **http://www.isus.jp/article/idz/mic-developer**

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 Composer Edition/Professional Edition/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。

インテル® VTune™ Amplifier XE による OpenMP* 領域の解析

Kirill Rogozhin 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
Dmitry Prohorov インテル® VTune™ Amplifier XE エンジニアリング・マネージャー

OpenMP* は、特にハイパフォーマンス・コンピューティング (HPC) で広く使用されているプログラミング・モデルです。OpenMP* 構文は、複数のスレッド間に作業を分配してパフォーマンスを向上します。ただし、コードに OpenMP* 宣言子を追加しても必ずしもパフォーマンスが向上するとは限らず、プログラマーによるチューニングが必要な場合もあります。チューニング手法はパフォーマンス解析と密接に関係しています。ボトルネックを解消するには、効率の悪い原因を特定する必要があります。

主要なパフォーマンス・ツールの多くは、特定の OpenMP* 並列領域ではなく、ボトムアップやトップダウン方式で関数 / ループに関連付けられたパフォーマンス関連の情報を表示します。その結果、並列領域 / 並列ループのコンテキストが失われ、インバランスやオーバーヘッドのような非効率性を理解することが困難になります。

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。

インテル® VTune™ Amplifier XE 2015 は、プログラマーが記述する OpenMP* 構文に即したプロファイル結果を生成できます。アプリケーションの並列時間とシリアル時間、並列領域の実行状況と理想的な実行との違い、並列ループごとの統計、領域ごとの CPU 使用率ヒストグラムを表示できます。プログラマーは、各問題を修正することで理論上達成可能なアプリケーションのウォールクロック時間を把握し、どこにチューニングの労力をかけるべきか、簡単に特定することができます。さらに、オーバーヘッドやスピン時間の分類は、効率の悪い原因 (例えば、ロード・インバランスによるバリアでの待機、同期によるロックでのスピンなど) を理解するのに役立ちます。

この記事では、インテル® VTune™ Amplifier により特定できる OpenMP* の非効率性の種類と、それらの対応方法を説明します。

BLOG HIGHLIGHTS

インテル® Xeon® プロセッサー・ベースのホストとインテル® Xeon Phi™ コプロセッサー間の MPI 通信の改善

LOC-NGUYEN »

MPI 対称モードは、インテル® Xeon Phi ™ コプロセッサーを搭載したシステムで広く利用されています。1 つ以上のコプロセッサーが装着されたインテル® Xeon® プロセッサー・ベースのホストシステムでは、ホストとコプロセッサー間または同じホストのコプロセッサー間の MPI メッセージの転送に伝送制御プロトコル (TCP) が使用されます。一部の転送速度を重視するアプリケーションでは、この MPI 通信の速度は十分ではありません。

このブログでは、OFED スタックをインストールして、DAPL ファブリックを代わりに使用することにより (インテル® Xeon® プロセッサー・ベースのホストとインテル ® Xeon Phi ™ コプロセッサー間の) MPI ノード内通信を向上する方法を示します。ホストに InfiniBand* ホスト・チャネル・アダプター (HCA) がない場合でも、DAPL ファブリックを使用して scif0 経由で仮想 InfiniBand* インターフェイスに MPI メッセージを転送できます。

Linux* カーネルバージョン 2.6.32-279 を実行する、2 つのインテル® Xeon Phi ™ コプロセッサー 7120 (C0 ステッピング、mic0 および mic1) を搭載したインテル® Xeon® プロセッサー E5-2670 ベースのホストシステムに、インテル ® MPSS 3.3.2 およびインテル® MPI ライブラリー 5.0 をインストールしました。インテル® MPI ライブラリーには、ベンチマーク・ツール IMB-MPI1 が含まれています。結果を比較するため、OFED スタックのインストール前とインストール後にインテル® MPI ベンチマーク sendrecv を実行しました。このテストでは 2 つのプロセスを使用し、各プロセスはメッセージを送信し、ほかのプロセスからメッセージを受信します。ツールは、双方向の帯域幅をレポートします。

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。

単純な構成

最新バージョンのインテル® コンパイラーとインテル® VTune™ Amplifier XE 以外に特別な構成要件はありません。この記事で説明するすべての機能を利用するには、次のバージョンを入手してください。

- インテル® VTune™ Amplifier XE 2015 Update 2 以降
- インテル® Parallel Studio XE 2015 Composer Edition Update 2 以降

OpenMP* アプリケーションをプロファイルするには、Advanced Hotspots (高度な hotspot) などの解析を実行するだけです。唯一必要な追加の設定は、パフォーマンス収集を実行する前に次の環境変数を設定することです。次に例を示します (Linux* の場合)。

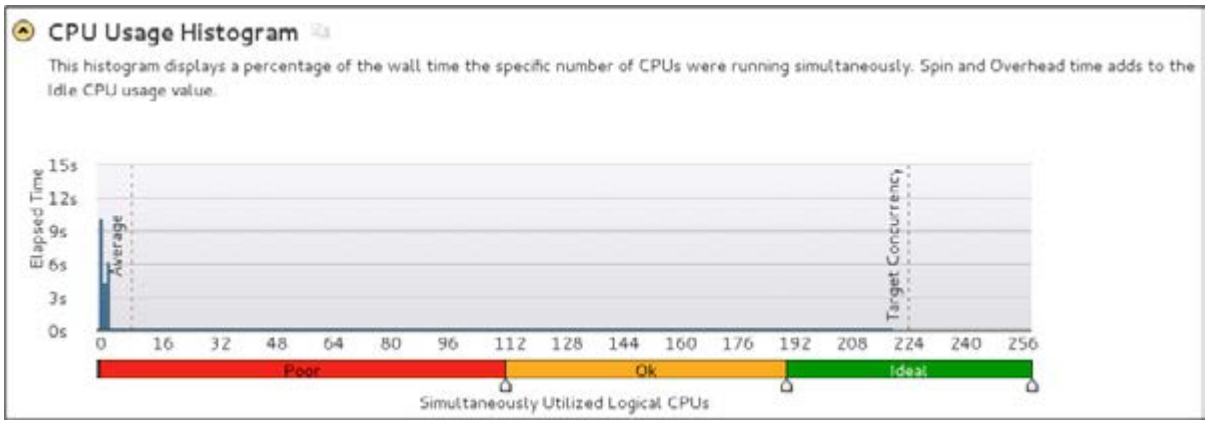
```
export KMP_FORKJOIN_FRAMES_MODE=3
```

この記事の執筆時点では、インテルの OpenMP* ランタイムの (バリアごとの統計に関連する) インストルメンテーションがまだ試験的な状態であり、環境変数を使用して明示的に機能を有効にする必要があります。

この試験的な機能が、将来正式リリースに含まれるかどうかは未定です。また、試験的な機能を有効にして収集したデータは、将来のリリースで下位互換性が保証されません。

非効率性の特定 : シリアルコードと CPU 使用率

最初に、解析結果の [Summary] ペインで CPU 使用率ヒストグラムを確認することを推奨します。アプリケーションの経過時間が、CPU 使用率のレベル別に表示されます (**図 1**)。ヒストグラムには、有効な使用率のみ表示されます。アプリケーションがスピンループ (アクティブウェイト) で費やした CPU サイクルはこのヒストグラムに表示されません。



1 CPU 使用率ヒストグラム

インテル® ソフトウェア開発製品

intel Software

革新的なアプリケーションを
実現する業界最先端のツール

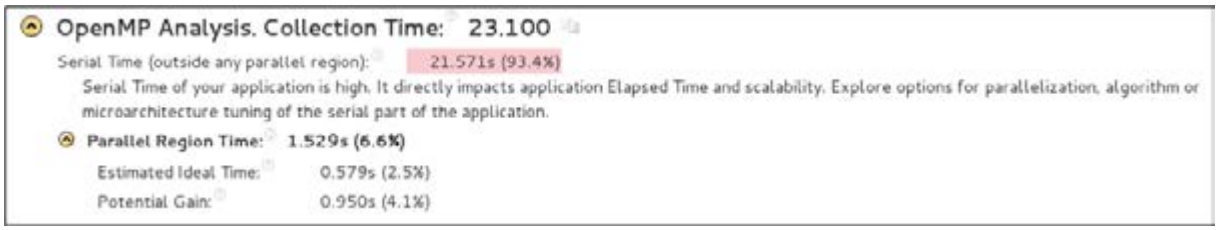
インテル® Parallel Studio XE
インテル® System Studio
インテル® Media Server Studio
インテル® INDE
インテル® XDK

intel.co.jp/software >

並列アプリケーションでは、大部分の CPU コアが利用され、ほとんどの経過時間が " 緑 " の領域に集中することが理想的です。図 1 は、インテル® Xeon Phi™ コプロセッサーで実行したテストの結果を示しています。大部分の計算リソースが十分に活用されていないことは明らかです。2 つの主な理由が考えられます。

- シリアル領域が大きい : ほとんどのコードは並列で動作していません。
- 並列領域の効率が悪い : コードは並列ですが、ボトルネックによりスケーラビリティーが制限されています。

次に、同じ [Summary] ペインの [OpenMP Analysis] セクションを確認します (**図 2**)。

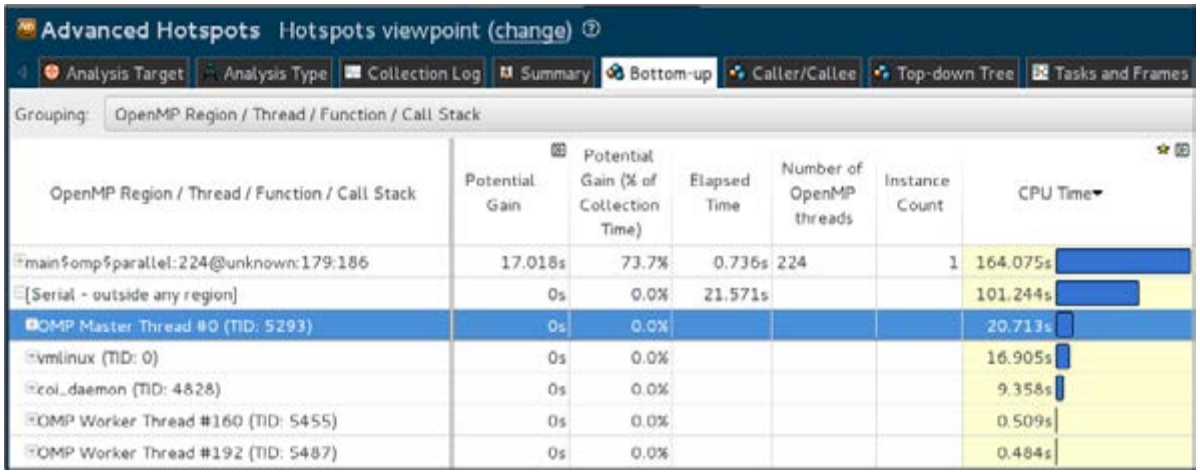


2 [OpenMP Analysis] ペインにアプリケーションのシリアル領域と並列領域の経過時間が表示される

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。

このセクションには、プログラムのシリアル領域と並列領域の経過時間が含まれます。シリアル領域が多い場合は、まずそこから確認すると良いでしょう。シリアル処理を並列処理に移行するか、シリアルでなければならない領域はアルゴリズムまたはマイクロアーキテクチャーのチューニングを行い、シリアル実行を最小化する方法を調べます。スレッド数の多いマシンでは、シリアル領域は (アムダールの法則により) スケーリングに重大な悪影響を及ぼすため、可能な限り最小化すべきです。テストでは、ウォールクロック時間の 93.4 パーセントがシリアル領域で費やされており、ボトルネックになっていることは明白でした。

シリアルに実行されるコードは、[Bottom-up] タブで、[OpenMP Region/Thread/Function/Call Stack] グループを選択し、[Serial - outside any region] 行の OMP Master Thread でフィルターして確認できます (**図 3**)。このシリアル領域の実行には 21.571 秒かかっており、経過時間のほとんどを占めています。しかし、CPU 時間でソートすると、行 179 の OpenMP* 並列領域が多数を占めています。この領域の CPU 時間は 164 秒で、シリアル領域の ~101 秒よりも長いためです。シリアル領域を確認するときは、必ずマスタースレッドのみでフィルターするようにします。シリアル領域の実行中は、多くの OpenMP* ワーカースレッドがバリアで待機またはスピンします。純粋なシリアルコードを調べるためには、これらの時間を除外すべきです。この時間は CPU 時間の 20.713 秒、つまり最上位の並列領域の数分の一にすぎませんが、高度な並列デバイスであるインテル® Xeon Phi™ コプロセッサーでは、パフォーマンスに大きく影響します。アムダールの法則はここでも例外なく適用されます。

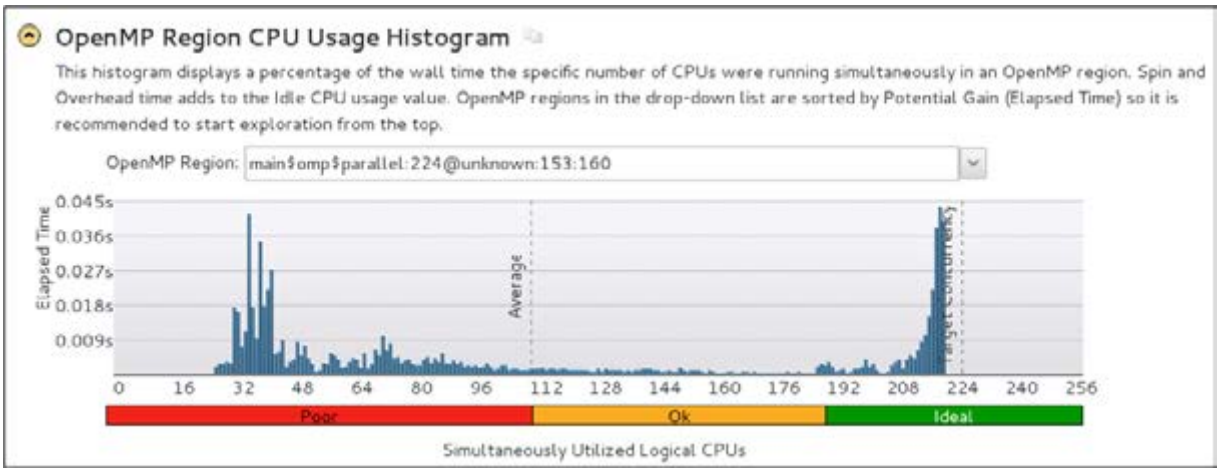


OpenMP Region / Thread / Function / Call Stack	Potential Gain	Potential Gain (% of Collection Time)	Elapsed Time	Number of OpenMP threads	Instance Count	CPU Time
mainompparallel:224@unknown:179:186	17.018s	73.7%	0.736s	224	1	164.075s
[Serial - outside any region]	0s	0.0%	21.571s			101.244s
OMP Master Thread #0 (TID: 5293)	0s	0.0%				20.713s
vmlinux (TID: 0)	0s	0.0%				16.905s
coi_daemon (TID: 4828)	0s	0.0%				9.358s
OMP Worker Thread #160 (TID: 5455)	0s	0.0%				0.509s
OMP Worker Thread #192 (TID: 5487)	0s	0.0%				0.484s

3 [Bottom-up] タブのシリアル時間

並列領域の効率は、[Summary] ペインの [OpenMP Region CPU Usage Histogram] で確認できます。これは CPU 使用率ヒストグラムと本質的に同じですが、全体ではなく選択した並列領域の CPU 使用率を示します。**図 4** は、**図 1** のインテル® Xeon Phi™ コプロセッサーの CPU 使用率ヒストグラムから、行 153 の並列領域のみを示したものです。理想値の 224 に近いハードウェア・スレッドが使用されている個所もありますが、多くの時間は 32 から 40 ハードウェア・スレッドで費やされています。

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。



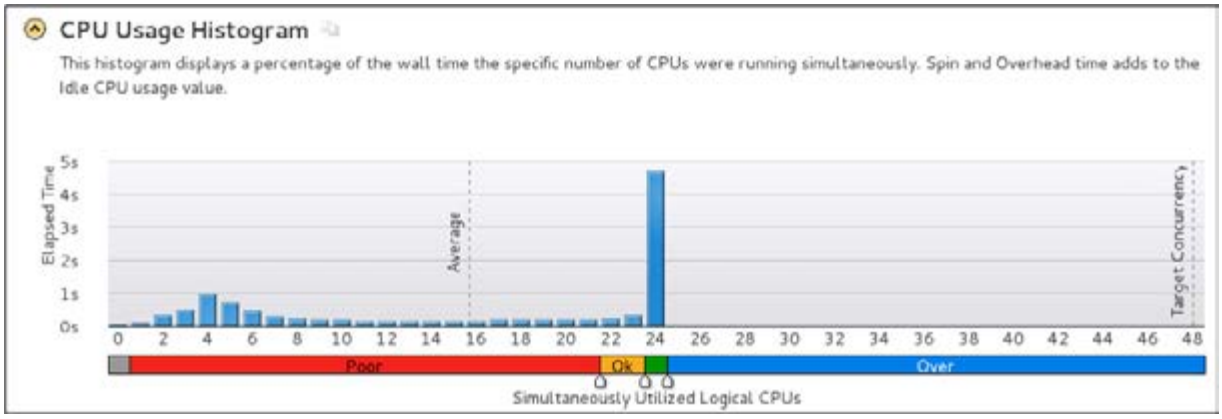
4 選択した並列領域の CPU 使用率ヒストグラム

パフォーマンス向上の可能性の特定 : ポテンシャル・ゲイン

別のテストケースを見てみましょう。 このテストでは、 NAS 並列ベンチマーク (NPB) を解析して、 OpenMP* 並列領域実行で発生する可能性のある問題を調べました。

テスト構成 :

- CPU: インテル® Xeon® プロセッサー E5-2697 v2 (2.70GHz)、 24 コア /48 スレッド
- OS: RHEL* 7.0 x64
- コンパイラー : インテル® Parallel Studio XE 2015 Composer Edition Update 2
- ワークロード : NPB 3.3.1、 "CG - 共役勾配、 不規則メモリーアクセスと通信 " モジュール、 クラス B



5 CPU 使用率 : NPB ベンチマーク

OpenMP* スレッドの数は、 物理コアの数と一致するように 24 に設定しました。**図 5** は、 ベンチマークを実行した場合の CPU 使用率ヒストグラムです。 長いシリアル領域を含むコードをインテル® Xeon Phi™ コプロセッサーで実行した前述の例よりも CPU 使用率が良いことが分かります。 しかし、 同時実行コア数が 2 から 6 の間でかなりの時

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。

間が占められているため、 まだ理想的ではありません。 これは、 並列コードが利用可能な 24 コアすべてを常に使用していないことを示しています。NPB のシリアル時間は無視できるため、問題ではありません (**図 6**)。ピンクでハイライトされている、[Potential Gain] の値を確認してください。

6 OpenMP* 解析 : NPB ベンチマーク

ポテンシャル・ゲインは、 スレッドのバランスが完全にとられており、OpenMP* ランタイムのワーク・スケジュールでオーバーヘッドが発生しないと仮定して、 並列領域での経過時間の実測値と理想値の差を評価します。 つまり、 ポテンシャル・ゲインとは、 チューニングによって得られる可能性がある利点であり、並列実行効率の向上により短縮できる最大時間を示します。 ポテンシャル・ゲインは、 最も時間が費やされた領域ではなく、 チューニングにより最も利点が得られる領域に注目するため、 CPU や経過時間よりもはるかに重要です。

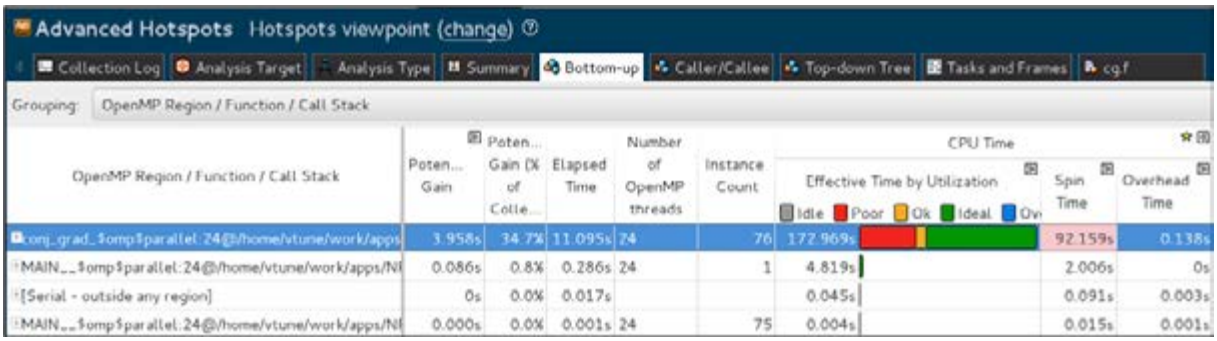
テストでは、 すべての並列領域を理想的な状態に最適化すると、 実行時間を 3.975 秒 (合計アプリケーション実行時間の 34.9%) 短縮できる可能性があると示されました。

これは、アプリケーション全体のポテンシャル・ゲインなので、より詳しく並列領域レベルで見てみましょう。[Summary] タブに、 ポテンシャル・ゲインでソートした上位 5 つの OpenMP* 並列領域が表示されています。 テストでは、 行 514 の並列領域がアプリケーション全体のポテンシャル・ゲインのほぼすべてである 3.958 秒を占めていました (**図 6**)。そのため、問題をこの 1 つの並列領域に絞り込むことができました。

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。

非効率性の原因の特定

特定の並列領域に絞り込むことができたら、領域名のリンクをクリックして [Bottom-up] ビューを開きます。OpenMP* 領域と選択した領域のグループが表示されます (**図 7**)。



7 OpenMP* 領域でグループ化したグリッドビュー

[Bottom-up] ビューに、並列領域に関するさまざまな統計 - 経過時間とポテンシャル・ゲイン、選択した領域で使用された OpenMP* ワーカースレッド数、領域のインスタンス数 (領域が外部ループや関数から呼び出された回数) - が表示されます。CPU 時間は、有効時間 (ユーザーコードの実行時間)、スピン時間、オーバーヘッド時間に分類されます。スピン時間が 92.159 秒と非常に長いことがわかります。この原因を調査する前に、**図 8** のソースコード (Fortran) を見てみましょう。行 514:695 の並列領域には、"**`!$omp do`**" 構文で指示された複数の並列ループが含まれているため、どの並列ループに問題があるか分かりません。

```
514 !$omp parallel default(shared) private(j,k,cgit,suml,alpha,beta)
515 !$omp&  shared(d,rho0,rho,sum)
516
517 c---------------------------------------------------------------------
518 c  Initialize the CG algorithm:
519 c---------------------------------------------------------------------
520 !$omp do
521       do j=1,naa+1
522          q(j) = 0.0d0
523          z(j) = 0.0d0
524          r(j) = x(j)
525          p(j) = r(j)
526       enddo
527 !$omp end do
528
529
530 c---------------------------------------------------------------------
531 c  rho = r.r
532 c  Now, obtain the norm of r: First, sum squares of r elements locally...
533 c---------------------------------------------------------------------
534 !$omp do reduction(+:rho)
535       do j=1, lastcol-firstcol+1
536          rho = rho + r(j)*r(j)
537       enddo
538 !$omp end do
539
```

8 行 514 の OpenMP* 並列領域

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。

幸いにも、インテル® VTune™ Amplifier XE は、並列領域ではなく OpenMP* バリアでも情報を分類することができます。"nowait" 節を指定しない限り、すべての "**#omp for**" 構文と "**!$omp do**" 構文には同期バリアが含まれます。インテル® VTune™ Amplifier はこれらのバリアを特定できるため、同じ並列領域内の各並列ループの CPU 時間とウォールクロック時間を見ることができます。バリアごとのデータを見るには、[OpenMP Region/OpenMP Barrier Type/OpenMP Barrier/...] のようなカスタムグループを作成します。

OpenMP* バリアでグループ化すると、状況を把握しやすくなります。まず、ポテンシャル・ゲイン、経過時間、CPU 時間のほとんどは、**図 9** でハイライトされている行 572 の並列ループからであることが分かります。次に、[Spin Time] 列の分類から、すべてのスピンはインバランスが原因であることが分かります。最後に、[OpenMP Loop Schedule Type] 列から、行 572 のループが静的にスケジュールされていることが分かります。

Grouping: (custom) OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack

OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack	Poten... Gain	Elapsed Time	Number of OpenMP threads	OpenMP Loop Chunk	OpenMP Loop Schedule Type	CPU Time: Effective Time ... (Idle, Poor, C)	CPU Time: Spin Time: Imbalance...	CPU Time: Spin Time: Lock C...	CPU Time: Spin Time: Other
conj_grad_ompparallel::	3.958s	11.095s	24			172.969s	91.947s	0s	0.212s
Loop barriers	3.954s	11.086s	24			172.911s	91.825s	0s	0.211s
conj_grad_omploop_	3.734s	10.445s	24	3125	Static	163.287s	86.096s	0s	0.198s
conj_grad_omploop_	0.150s	0.418s	24	3125	Static	6.528s	3.449s	0s	0.007s
conj_grad_omploop_	0.036s	0.068s	24	3125	Static	0.450s	1.116s	0s	0.004s

9 バリアごとのパフォーマンス評価基準

インバランスの修正

行 572 のループのソースコードを**図 10** に示します。"Schedule" 節が指定されていないため、デフォルトで静的にスケジュールされます。並列ループにインバランスの問題がある場合、OpenMP* ランタイムによってワークロードが自動的に再分配される可能性がある動的スケジュールを試してみることは自然な流れです。変更後のコードを**図 11** に示します。

```
572 !$omp do
573          do j=1,lastrow-firstrow+1
574             suml = 0.d0
575             do k=rowstr(j),rowstr(j+1)-1
576                suml = suml + a(k)*p(colidx(k))
577             enddo
578             q(j) = suml
579          enddo
580 !$omp end do
```

10 行 572 の OpenMP* 並列ループ (オリジナル)

```
572 !$omp do schedule (dynamic)
573          do j=1,lastrow-firstrow+1
574             suml = 0.d0
575             do k=rowstr(j),rowstr(j+1)-1
576                suml = suml + a(k)*p(colidx(k))
577             enddo
578             q(j) = suml
579          enddo
580 !$omp end do
```

11 行 572 の並列ループ (動的スケジュール)

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。

スケジュール・オーバーヘッドの解析と修正

動的スケジュールに変更すると、 並列ループのパフォーマンスはさらに悪くなりました。 経過時間はオリジナルの 10.445 秒から 11.102 秒に増えました (**図 12**)。 ただし、結果をよく見ると、インバランスがなくなっていることが分かります。 つまり、 インバランスは解消されたのです。 しかし、 別の列がピンクでハイライトされ、 74.99 秒のスケジュール・オーバーヘッドが発生しています。 これは、 OpenMP* ランタイム・ライブラリーが一部の内部処理を過度に実行していることを意味します。

Grouping: (custom) OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack

OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack	Poten... Gain	Elapsed Time	OpenMP Loop Chunk	OpenMP Loop Schedule Type	CPU Time: Effective Time... (Idle, Poor)	CPU Time: Spin Time	CPU Time: Overhead Time: Creation ...	CPU Time: Overhead Time: Scheduling...	CPU Time: Overhead Time: Reduction...	CPU Time: Overhead Time: Other
conj_grad_ompparallel:24	3.338s	11.758s			199.298s	5.866s	0.001s	75.051s	0.012s	0.083s
Loop barriers	3.333s	11.748s			199.272s	5.709s	0.001s	75.021s	0.012s	0.071s
conj_grad_omploop_bar	3.133s	11.102s	1	Dynamic	189.320s	0.368s	0s	74.990s	0s	0.009s
conj_grad_omploop_bar	0.128s	0.412s	3125	Static	6.880s	2.969s	0s	0s	0s	0s
conj_grad_omploop_bar	0.031s	0.071s	3125	Static	0.512s	1.094s	0s	0.024s	0.010s	0.027s

12 行 572 のループのスケジュール・オーバーヘッド

[OpenMP Loop Chunk] 列を見てみましょう。 値がオリジナルの 3125 から 1 に変更されています。 これは、 各反復が個別にスケジュールされることを意味し、 ワーカースレッドのワーク項目が非常に小さいため、 OpenMP* ライブラリーは非常に頻繁にスケジュールを行う必要があります。 この並列処理の粒度は細かすぎます。

粒度 1 は動的スケジュールのデフォルト値で、 最小の値です。 粒度が細かすぎることを証明するため、 粒度を 20 に変更してみました (**図 13**)。

```
572 !$omp do schedule (dynamic,20)
573          do j=1,lastrow-firstrow+1
574             suml = 0.d0
575             do k=rowstr(j),rowstr(j+1)-1
576                suml = suml + a(k)*p(colidx(k))
577             enddo
578             q(j) = suml
579          enddo
580 !$omp end do
```

13 行 572 の並列ループ (動的スケジュール、粒度 = 20)

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。

パフォーマンスは**図 14** のようになり、インバランスとスケジュール・オーバーヘッドはどちらも約 1 秒になりました。ポテンシャル・ゲインがオリジナルの 3.133 秒から 0.077 秒に低下したため、行 572 のループのバリアは hotspot リストの下位に移動しました。また、このループの経過時間は、オリジナルの 10.445 秒から 8.928 秒に短縮されました。並列領域全体の CPU 時間は、オリジナルの約 250 秒から約 213 秒に短縮されました。約半分ではありますが、ポテンシャル・ゲインの一部は達成できたことになります。

Grouping: (custom) OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack

OpenMP Region / OpenMP Barrier Type / OpenMP Barrier / Function / Call Stack	Potenti... Gain	Potential Gain (% of Collection Time)	Elapsed Time	Number of OpenMP threads	OpenMP Loop Chunk	OpenMP Loop Schedu... Type	CPU Time: Effective Time by ... (Idle / Poor / Ok)	CPU Time: Spin Time	CPU Time: Over... Time
conj_grad_ompparallel:24@/h	0.264s	2.7%	9.568s	24			220.930s	6.135s	1.061s
Loop barriers	0.258s	2.6%	9.557s	24			220.904s	5.982s	1.019s
conj_grad_omploop_barrier	0.119s	1.2%	0.406s	24	3125	Static	6.963s	2.769s	0.001s
conj_grad_omploop_barrier	0.077s	0.8%	8.928s	24	20	Dynamic	210.848s	1.083s	0.893s
conj_grad_omploop_barrier	0.028s	0.3%	0.067s	24	3125	Static	0.538s	1.010s	0.083s
conj_grad_omploop_barrier	0.015s	0.2%	0.080s	24	3125	Static	1.365s	0.537s	0.022s

14 最適化されたループのパフォーマンス評価基準

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。

より効率良いパフォーマンス解析方法

インテル® VTune™ Amplifier XE を利用することで、OpenMP* アプリケーションのパフォーマンス解析をより自然に行うことができます。パフォーマンス効率とボトルネックをトップダウン方式 (一般的な CPU 使用率解析から開始して純粋なシリアルコードを確認) で調査し、アプリケーション全体または特定の並列領域の CPU 使用率を確認できます。ポテンシャル・ゲインは、パフォーマンス向上の可能性の点から最も注目すべき並列領域を知るのに役立ちます。また、バリアごとのデータ分類を使用すると、同じ並列領域内の複数の並列ループが確認しやすくなります。

[Bottom-up] ビューからは、詳細な OpenMP* 統計 (スケジュール・タイプ、チャンクサイズ、OpenMP* スレッド数、並列領域のインスタンス数) が分かります。パフォーマンス評価基準 (スピン時間やオーバーヘッド時間の詳細な分類など) は、パフォーマンス問題の根本的な原因をプログラマーに知らせます。これらの評価基準は、パフォーマンスを制限している原因 (ロード・インバランス、最適でない粒度、ロックでの待機など) を理解するのに役立ちます。

これらはすべて、インテルの OpenMP* ランタイム・ライブラリーとインテル® VTune™ Amplifier XE の機能によってもたらされる相乗効果です。どちらも、インテル® Parallel Studio XE 2015 Professional Edition に含まれており、評価版は無料でダウンロードできます。インテル® VTune™ Amplifier XE および関連資料は、インテル® VTune™ Amplifier XE 製品サイトからも利用できます。ツールを利用して OpenMP* アプリケーションのパフォーマンス問題を確認してみてください。

インテル® VTune™ Amplifier 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 Edition/Cluster Edition >

インテル® System Studio 2015 >

インテル® Media Server Studio Professional 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。

ウォーカー分子動力学研究室のバイオメディカル・ソフトウェアの最適化

インテル® ソフトウェア開発製品によるアプリケーションのパフォーマンスと生産性の向上

はじめに

サンディエゴ・スーパーコンピューター・センター (SDSC) のウォーカー分子動力学研究室では、高度な古典分子力学 (MM) と複合量子 / 分子力学 (QM/MM) シミュレーションの研究を行っています。創薬、生体触媒、酵素工学、高度なバイオマテリアル開発を含む、生体分子系研究のシミュレーションを可能にする、ウォーカーの Amber* 分子動力学ソフトウェアは、学術界および産業界の多くの科学研究者が利用しています。ウォーカー研究室は、多くの研究開発結果を Amber* ソフトウェアに適用し、分子動力学シミュレーションの最先端の手法をユーザーに提供しています。

ウォーカー研究室の研究対象は広範で、計算化学、分子生物学、ハイパフォーマンス・コンピューティング (HPC) の分野を含み、特に古典 MM と複合 QM/MM 手法の並列計算および高速計算用の効率的なアルゴリズムの開発に力を入れています。研究では、酵素の物理的・化学的性質の決定における MM および QM/MM 分子動力学アルゴリズムの使用にも注目しています。

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。

Amber* が研究者に提供するもの：

1. 生体分子シミュレーション用分子動力学力場のセット。これらはパブリックドメインで、さまざまなシミュレーション・プログラムで使用されています。
2. ソースコードとデモを含む、分子シミュレーション・プログラムのパッケージ。

ソフトウェアと力場は、酵素、脂質膜、高度なバイオマテリアル、触媒のシミュレーションに使用されます。脂質結合タンパク質との相互作用に加えて、薬物結合の親和性を予測し、セルの薬物透過性を正確に判定する能力が提供されるため、創薬に効果的です。

「インテル® MKL とインテル® MPI ライブラリーを実装したことにより、開発者は最小限の労力で大幅な速度向上を達成できました。エンドユーザーにとってもこの投資は非常に価値のあるものです。」

– カリフォルニア大学サンディエゴ校 (UCSD)
サンディエゴ・スーパーコンピューター・センター (SDSC)

ウォーカー分子動力学研究室
Ross Walker 准教授

課題

ウォーカー研究室は、Amber* の高度に最適化された分子動力学エンジン (PMEMD) 内の精度混在モデルのベクトル化を進めることで、インテル® アーキテクチャー上でさらに高いパフォーマンスを引き出そうとしていました。並列スケーラビリティーを高め、強固な基盤を構築するため、インテル® Xeon Phi™ コプロセッサーのオフロード・アクセラレーションをサポートすることにしました。さらに、ユーザーの要件を満たすため Amber* をインテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー対応にすることにしました。

ウォーカー研究室は、ソフトウェアを最新のインテル® プロセッサーで実行できるように、以前の経験に基づいて Amber* を変更しました。しかし、そのパフォーマンスは予想を大幅に下回るもので、どうすればパフォーマンスを向上できるのか特定することは困難でした。コード内の単精度から倍精度へのキャスト動作において、最初の仮定が正しくないことが分かりましたが、エンジニアはなぜ予想したパフォーマンスの向上が得られないのか原因が分からず困っていました。

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。

Amber* は、計算負荷の高いアプリケーションです。そのため、ユーザーは、ソフトウェアを最先端の HPC コンピューター・アーキテクチャーで実行することを要求します。ウォーカー研究室はインテルと協力して、インテル® Parallel Computing Center Program の下で、Amber* をインテル® Xeon Phi™ コプロセッサーで実行できるようにし、その後インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサー両方のアーキテクチャーでパフォーマンスを最適化するプロジェクトを開始しました。

「インテル® VTune™ Amplifier XE は、コードの最適化で hotspot の特定に役立つ非常に優れたツールです。ユーザー・インターフェイスは使いやすく、詳細な情報を基に開発を迅速に進めることができます。インテル® VTune™ Amplifier XE の行単位で表示されるパフォーマンス・カウンターがなければ、精度が混在したコードがオリジナルの倍精度コードより遅くなっていた理由を特定することはできなかったでしょう。」

– カリフォルニア大学サンディエゴ校 (UCSD)
サンディエゴ・スーパーコンピューター・センター (SDSC)

ウォーカー分子動力学研究室
Perri Needham 博士研究員

ソリューション

インテルのアプリケーション・エンジニア Ashraf Bhuiyan と共同作業を行うにあたり、ウォーカー研究室チームはインテル® VTune™ Amplifier を導入し、インテル® MPI ライブラリー (インテル® MPI) とインテル® マス・カーネル・ライブラリー (インテル® MKL) の最新機能を利用することにしました。インテル® MPI ライブラリーをサポートすることにより、並列パフォーマンスがすぐに 10 ～ 20 パーセント (MPI プロセス数に依存) 向上しました。また、インテル® MKL 内のインテルの最適化ルーチンを利用することで、ウォーカー研究室が元々使用していた pubFFT の修正バージョンのパフォーマンスが向上しました。

さらに、インテル® VTune™ Amplifier XE を利用することにより、ウォーカー研究室チームはコードを詳しく調査して、多くのパフォーマンス・ボトルネックを特定することができました。特に、単精度コードから倍精度コードへの過度なキャストが行われていたため、精度混在モデルで予想した速度向上を妨げていたコードの場所を特定することができました。ウォーカー研究室は、インテル® VTune™ Amplifier XE を利用してこの問題が発生しているコード行を特定し、インテル® コンパイラーのエンジニアに連絡してコンパイラーが不要

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。

なキャストを行っていた原因の調査を依頼しました。インテル® VTune™ Amplifier XE のタイミングを行単位に分割して提供する機能がなければ、このパフォーマンス hotspot の場所を特定することは不可能だったでしょう。

Ashraf は、最適なパフォーマンスが得られるようにインテル® MPI ライブラリーをチューニングするだけでなく、Amber* の hostpot を特定するためインテル® VTune™ Amplifier XE 環境のセットアップに協力し、現在使用しているオフロードコードも開発しました。この共同作業は現在も続けられています。Amber* v14 は、ビルドオプションを使用してインテル® MPI ライブラリーをサポートしました。さらに、Ashraf は、インテル® MPI ライブラリーをサポートするための正しいパスと環境の構築、SDSC のインテル® Xeon Phi™ コプロセッサー開発ハードウェアおよび関連ソフトウェアの準備も支援してくれました。

結果

パフォーマンスの向上

ウォーカー研究室は、インテル® VTune™ Amplifier XE でチューニング作業を行い、インテル® MPI ライブラリーおよびインテル® MKL を使用するようにシミュレーションを変更しました。インテル® MPI ライブラリー、インテル® コンパイラー、インテル® MKL を使用することにより、陽溶媒 PME (Particle Mesh Ewald) シミュレーションの並列スケーリングのパフォーマンスは約 20 パーセント、シリアル領域のパフォーマンスは (シミュレーションに応じて) 10 ～ 15 パーセント向上しました。また、暗溶媒 GB (Generalized Born) シミュレーションのパフォーマンスは 2.7 倍から 4.0 倍と大幅に向上しました。

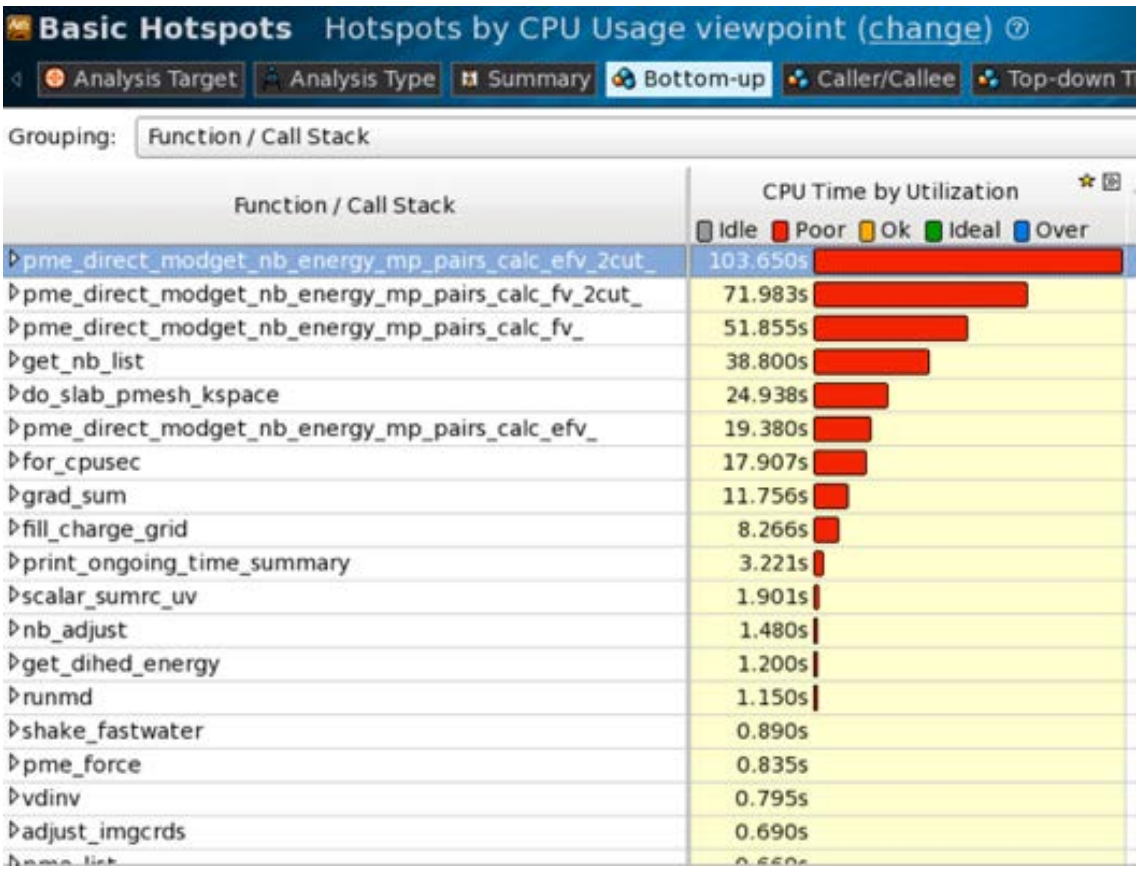
次の表は、サンプル・データ・セットのパフォーマンス向上の詳細を示しています。
インテル® Xeon Phi™ コプロセッサーのオフロードサポートに対応することで、陽溶媒シミュレーション (原子数 > 400,000) のパフォーマンスはさらに 10 パーセント向上しました。インテル® Xeon Phi™ コプロセッサー・アーキテクチャーからより長期的なパフォーマンス・ゲインを得るため、インテル® VTune™ Amplifier XE によるチューニング作業は現在も続けられています。

Amber*			MPI/ コンパイラー		
			GNU* - Red Hat* 4.4.7-3 / MPICH2 v1.5	インテル (2013/impi 4.1.1.036)	
			シミュレーション結果	シミュレーション結果	
モデル	システム	システムサイズ	ナノ秒 / 日	ナノ秒 / 日	スピードアップ (インテル /GNU*)
Generalized Born (GB)	TRPCAGE (304)	304	63.57	176.12	2.77 倍
	MYOGLOBIN (2,492)	2,492	1.17	4.43	3.79 倍
	NUCLEOSOME (25,095)	25,095	0.01	0.04	4.00 倍
Particle Mesh Ewald-NVE アンサンブル	JAC - 4fs (23,558)	23,558	13.09	15.36	1.17 倍
	JAC -2fs (23,558)	23,558	6.88	8.01	1.16 倍

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。

生産性

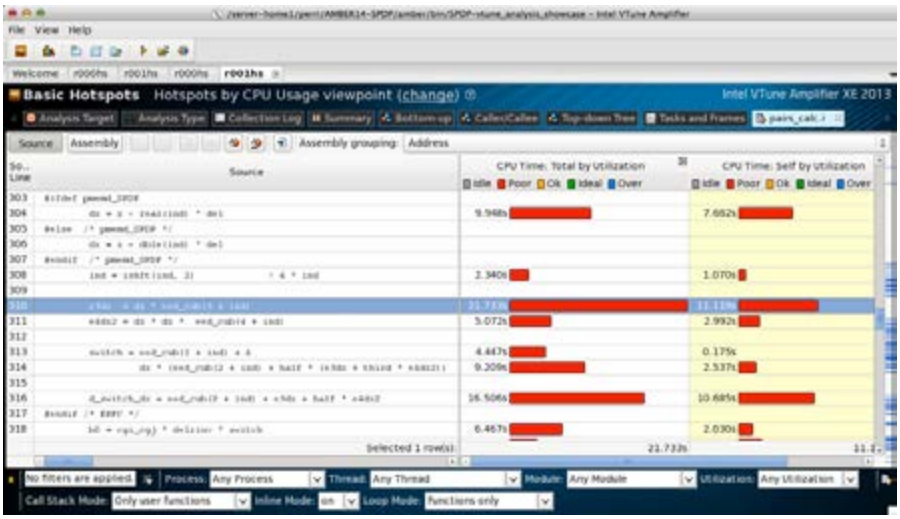
インテル® VTune™ Amplifier XE は、コードの hotspot をすばやく特定し、開発期間を短縮しました。具体的には、不要なキャストによりパフォーマンスが低下している単精度 / 倍精度の精度混在モデルの領域を特定しました。インテル® VTune™ Amplifier XE により、ウォーカー研究室は、なぜコンパイラーがそのような方法で最適化を行っていたのか、異なるコードレイアウトで実験して調査することができました。また、異なるカットオフサイズや異なる FFT 格子サイズのような、異なる入力設定を用いて、コードの重要な hotspot を特定することができました。



1 特定のコードの hotspot の特定

このスクリーンショットから、pme_direct_modget_nb_energy には 4 つのバリエーションがあることが分かります。ウォーカー研究室は、インテル® VTune™ Amplifier XE を利用して、各関数のコスト追跡と最適化の優先度付けを行いながら、1 つのルーチンを 4 つの機能別のルーチンに分割して、内部ループで if 文を回避することができました。

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。



2 CPU の hotspot の特定

インテル® VTune™ Amplifier XE は、 予想よりも多くの CPU 時間がかかっていたコード行の特定に大いに役立ちました。 ウォーカー研究室でこれらの行に対応するコンパイル済みマシンコードをさらに調査した結果、 コンパイラーが単精度から倍精度コードへのキャストを行い、 再度逆のキャストを行っていたことがわかりました。 コンパイル済み実行ファイルの hotspot の場所を特定した後、 ウォーカー研究所はインテル® コンパイラーのエンジニアに問題の調査を依頼しました。

MPI パフォーマンスの向上により、 シミュレーションの時間が短縮され、(より重要なことに) シミュレーションをより長く実行できるようになりました。 ユーザーは、 より広範な創薬ターゲットのような、 異なる系のより多くのシミュレーションを実行できるようになりました。 また、 個々のシミュレーションの収束が改善され、 まれに発生するイベントやより長いタイムスケールの動きを見ることができるようになりました。 これにより、 科学的な発見の可能性が高まることが期待されます。

まとめ

Amber* は現在 GPU、 そしてインテル® Xeon Phi™ コプロセッサー・アーキテクチャーをサポートしています。 ウォーカー研究室は、 インテル® Xeon Phi™ コプロセッサーの柔軟性により、 ユーザーがネイティブモードで多くのシミュレーションを同時に実行したり、 オフロードモードで Amber* GPU エンジンと同等のスピードアップが得られると予想しています。

ウォーカー研究室は、 優れた最適化とパフォーマンスがもたらされると確信しています。 特に、 最新のインテル® Xeon Phi™ コプロセッサーに焦点を当てたことで、より高速なシミュレーションが可能になるでしょう。 Amber* のユーザーは、 より長いシミュレーションを実行でき、 より速いペースで科学研究を行うことができることから、 多くの利点が得られるでしょう。 また、 デスクトップで対話型の計算や計算の試行が可能なシミュレーションからリアルタイムのフィードバックを行うような、科学研究の新しいアプローチを切り拓くことを期待しています。

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。

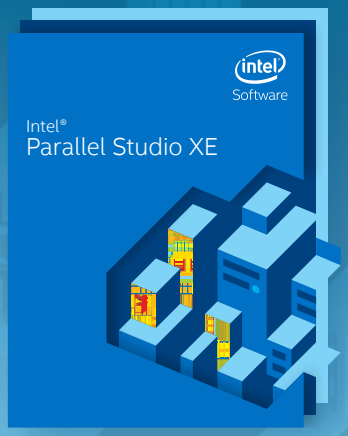
ウォーカー分子動力学研究室について

カリフォルニア大学サンディエゴ校サンディエゴ・スーパーコンピューター・センターのウォーカー分子動力学研究室の研究対象には、計算化学、分子生物学、ハイパフォーマンス・コンピューティング (HPC) の分野が含まれます。研究所は、古典 MM 動的シミュレーションの計算効率と精度の向上に加えて、特に量子力学と複合量子 / 分子力学 (QM/MM) の並列計算および GPU 高速計算の効率的なアルゴリズムの開発に取り組んでいます。また、タンパク質ベース系の物理的・化学的性質の決定における MM および QM/MM 分子動力学アルゴリズムの使用にも注目しています。

インテル® ソフトウェア開発製品

intel Software

HPCwire 誌 2014 年リーダーズ / エディターズ・チョイス・アワードを受賞



インテル® Parallel Studio XE
Cluster Edition
ベスト HPC ソフトウェア製品 / テクノロジー – エディターズ・チョイス

インテル® MPI ライブラリー
ベスト HPC クラスター・ソリューション / テクノロジー – リーダーズチョイス

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。

ウォーカー研究室は、米国国立科学財団 (NSF)、米国国立衛生研究所 (NIH)、カリフォルニア大学、インテル、NVIDIA、 Microsoft、 英国外務・英連邦省、 ビジネス・イノベーション・職業技能省から研究助成金を受けています。研究所で開発され実用化された技術には、 次世代のウィルス阻害剤、 バイオエタノール生産の改良、 創薬、 スーパーコンピューター上のハイパフォーマンス科学アプリケーション用の高度なアルゴリズムなどがあります。

詳細は、www.wmd-lab.org (英語) を参照してください。

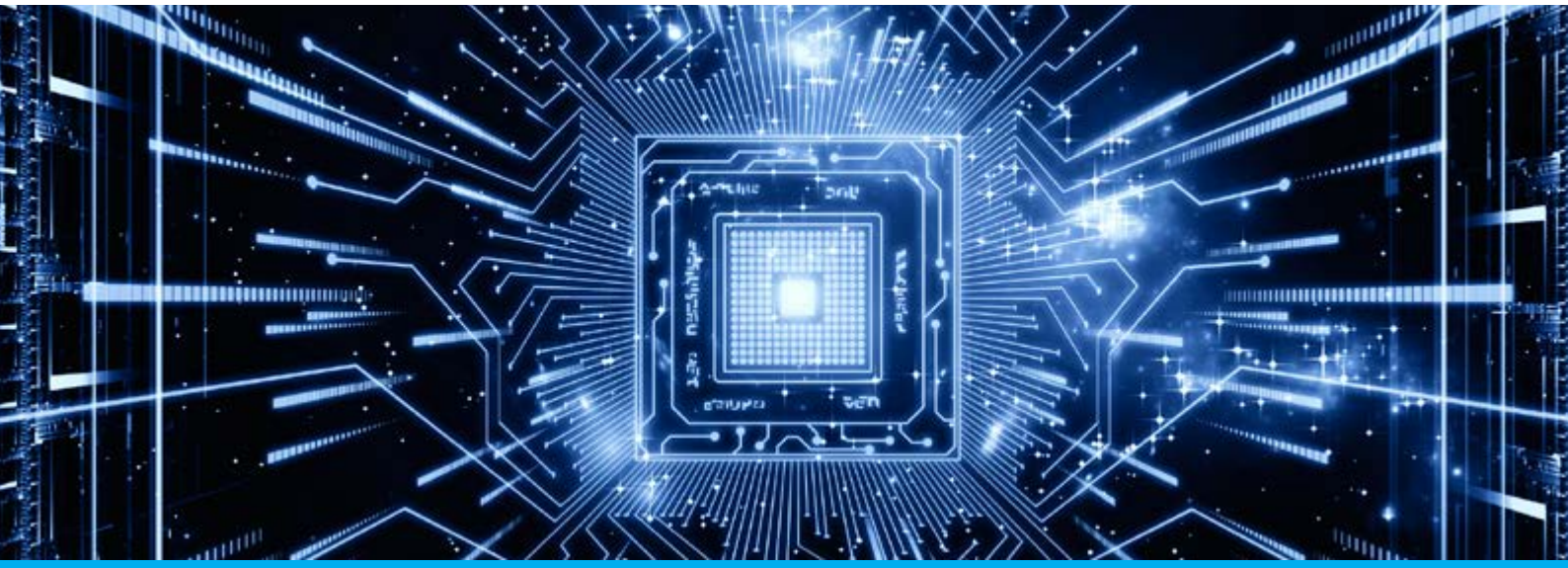
インテル® ソフトウェア開発製品について

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。業界最先端のソフトウェア・ツールには、 インテル® Parallel Studio XE Composer Edition/Professional Edition/Cluster Edition が含まれます。

インテル® ソフトウェア開発ツールの詳細は、intel.co.jp/software を参照してください。

インテル® ソフトウェア開発製品を試そう >

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。



インテル® ソフトウェア開発製品が HPCwire アワードを獲得

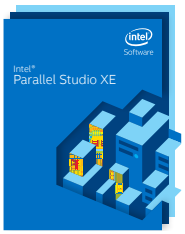
2 つのインテル® ソフトウェア開発製品、インテル® MPI ライブラリーおよびインテル® Parallel Studio XE Cluster Edition – ハイパフォーマンス MPI ハイブリッド・クラスター開発ツールスイートが、HPCwire 誌のアワードを受賞しました。

「HPCwire は、非常に厳格なプロセスを経てコミュニティーからのフィードバックを得ることにより、コミュニティーが考えるアワードの候補者を毎年選んでいます。コミュニティーのフィードバックに基づいて候補者を決定し、候補者を発表して、さまざまなアワード分野でどの候補者が最も影響力を与えているか、読者からの意見を集計します。」と HPCwire シニアエディターの Tiffany Trader 氏は述べています。

「ハイパフォーマンスなデータインテンシブ・コンピューティングに興味を持つ科学、テクノロジー、ビジネス・プロフェッショナルのポータルである HPCwire は、候補者と受賞者を客観的に選ぶように努めています。我々は、編集グループとして特定の 1 つのソリューションを支持しないように注意しています。

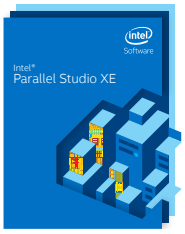
インテルは、コミュニティーのニーズに対応するさまざまなツールを取り揃えた、HPC エコシステムにおける主要企業です。インテルが並列コンピューティングと HPC の分野で非常に重要な役割を果たしており、業界から注目をされていることは明らかです。」

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。



ベスト HPC ソフトウェア製品 / テクノロジー – エディターズ・チョイス : **インテル® Parallel Studio XE Cluster Edition – ハイパフォーマンス MPI ハイブリッド・クラスター開発ツールスイート**

インテル® Parallel Studio XE Cluster Edition は、HPC およびクラスターアプリケーションの開発、デバッグ、チューニングを簡素化して高速化します。この高度な C++/Fortran 統合ツールスイートは、標準規格に準拠したコンパイラー、プログラミング・モデル、ツールにより、効率的な開発を支援します。これらのツールは、ビルトインの直感的な並列モデル、ベクトル化サポート、マルチファブリック MPI ライブラリー、高度な MPI エラーチェックとプロファイリングによりパフォーマンスを向上します。



ベスト HPC クラスター・ソリューション / テクノロジー – リーダーズチョイス : **インテル® MPI ライブラリー**

インテル® MPI ライブラリー 5.0 は、インテル® アーキテクチャー・ベースのクラスターでアプリケーションをより最適に実行できるように、複数のファブリックにハイパフォーマンス MPI (メッセージ・パッシング・インターフェース) バージョン 3.0 仕様を実装しています。ソフトウェアや操作環境の変更を要求することなく、新しいインターコネクトに変更またはアップグレードするだけで、エンドユーザーは最高のパフォーマンスを得ることができます。

インテル® MPI ライブラリー 5.0 は、インテル® Parallel Studio XE Cluster Edition に含まれています。

インテル®
Parallel Studio XE Cluster Edition の無料体験版を試す >

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

The Parallel Universe Issue 20 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

本冊子の ISSUE 1 から編集作業に携わり、早いもので今回で ISSUE 20 となります。

英文冊子をそのまま日本語版にしてくださいと依頼され作り出してから早 6 年、単純に英語を日本語にするだけだと思われがちですが、英語フォントを日本語フォントに変更するだけで大分見栄えが変わります。さらに英語から日本語へ翻訳した際には、文字数が変わるため、英文冊子に合わせるには一工夫が必要です。その他にも日本語版ではさまざまな調整が必要で、一見単純な作業のように見えますが、簡単ではありません。

毎回このプロジェクトが来るたび、私の物作り魂に火が付き、もっと読みやすく、美しい冊子を皆様にお届けできるよう、楽しみながら作成させていただいております。

私だけでなく我々編集部員は皆、この冊子が皆様にとって価値のあるものになるよう思いを込めて作成しておりますので、この Parallel Universe が一人でも多くの方の目に触れ、今後の研究や開発業務に役立てていただければと願うばかりです。

iSUS 事務局 Parallel Universe 編集部員 森

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。

intel Software

並列計算を高速化

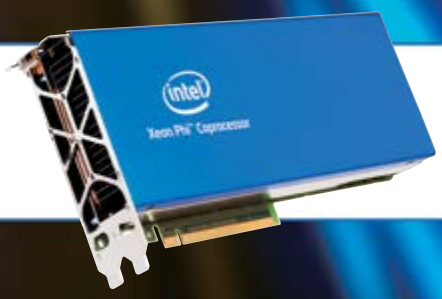
インテル® Parallel Studio XE 2015

Cluster Edition

MPI/ マルチスレッドによる並列計算のパフォーマンスや、信頼性、開発効率の向上を支援するツールスイートです。インテル® Parallel Studio XE を使用することで、より簡単により速く、ハードウェアの性能を最大限に発揮させることができるようになります。

以下をバンドル

- インテル® C/C++、Fortran コンパイラー : 最新プロセッサーの性能を最大限に引き出せるように、最適化されたバイナリーを自動生成
- インテル® MKL: 数値演算　・インテル® IPP: 画像処理　・インテル® TBB: マルチスレッド C++ テンプレート・ライブラリー
- インテル® VTune™ Amplifier XE: パフォーマンスの問題となる原因を素早く特定
- インテル® Inspector XE: マルチスレッド・エラー、メモリーリークを動的に検出
- インテル® Advisor XE: マルチスレッド化すべき個所を特定し、効果を予測
- インテル® MPI ライブラリー : 最適化済みの MPI ライブラリー
- インテル® Trace Analyzer & Collector: パフォーマンスに影響する MPI 通信を解析しソースコードを特定



インテル® Parallel Studio XE は、
インテル® Xeon Phi™ コプロセッサーに対応しています！

※ 既存のコードを利用することで、比較的容易に、インテル® Xeon Phi™ コプロセッサーにプログラムを移行できます。
※ 一部の製品を除きます。

インテル® ソフトウェア開発製品のお問い合わせはこちら

http://www.intel.co.jp/software
または総販売代理店のエクセルソフト株式会社 (Tel: 03-5440-7875 / E-mail: intel@xlsoft.com) までお問い合わせください。

無料評価版はこちら　**www.xlsoft.com/intel**

インテル® ソフトウェア製品のパフォーマンス / 最適化に関する詳細は、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 Xeon Phi、VTune は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。
* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

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 Xeon Phi、VTune、Xeon 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。