インテル® ソフトウェア開発ツールで始める

コードの現代化と最適化

インテル® C++ および Fortran コンパイラーの導入

2019 年 4 月
iSUS 編集長
すがわら　きよふみ

内容

- はじめに: インテル® コンパイラーとは
- インテル® Parallel Studio XE 2019
- インテル® コンパイラーの最適化機能とオプション

© 2019 iSUS *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。

インテル® ソフトウェア開発ツール

25 年以上の歴史

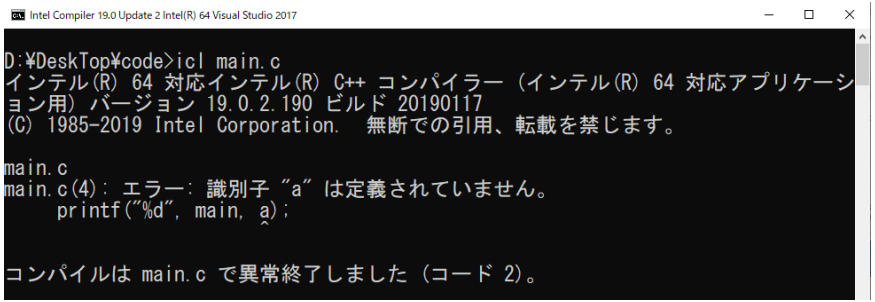
- 日本語版は 1995 年より提供
- インテル社から提供される最新の日本語バージョンは、インテル® コンパイラー 2019 (V19)
- 2011 年から日本語技術ポータルサイト iSUS (www.isus.jp) を運営中

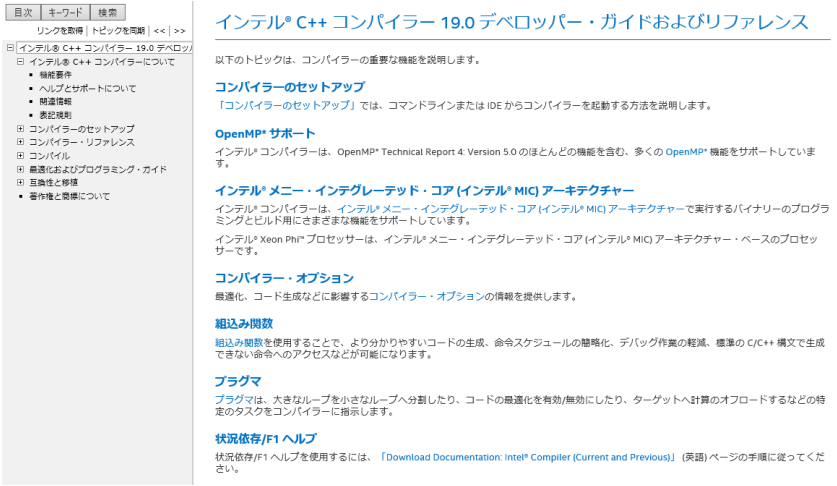
iSUS ではインテル® Advisor 2018/2019、インテル® VTune™ Amplifier 2019 の日本語パッケージを提供中

© 2019 iSUS *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。

日本語サポート: Windows* および Linux* 向け

- コンパイラー・メッセージの日本語化
  - 英語/日本語の切り替えも可能
- インテル® コンパイラー・デベロッパー・ガイドおよびリファレンスの日本語化
  - C/C++
  - Fortran
- 日本語での技術サポート (エクセルソフト社が提供)

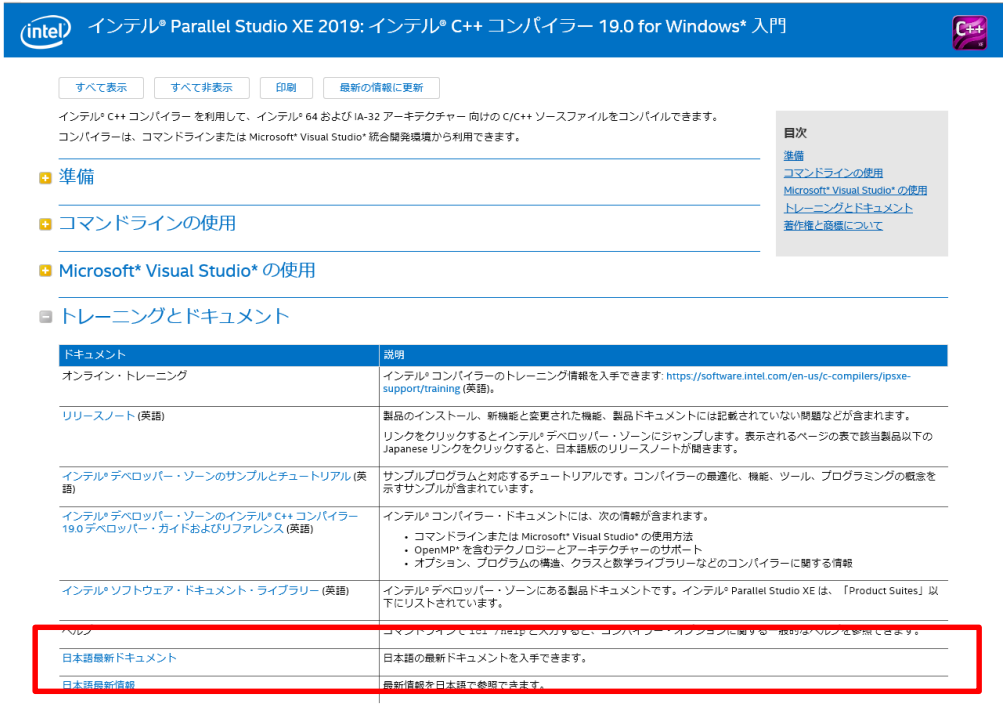




© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラー日本語マニュアル入手

- インテル® ソフトウェア開発製品のほとんどのドキュメントはオンライン化されていますが、オンラインで日本語マニュアルは参照できません
- オフライン版の日本語ドキュメントを入手するには、コンパイラーのインストール後、導入ガイド (Getting Started) を開いて、ドキュメント・セクションを開いて案内に従ってください



インテル® Parallel Studio XE 2019: インテル® C++ コンパイラー 19.0 for Windows* 入門

すべて表示 | すべて非表示 | 印刷 | 最新の情報に更新

インテル® C++ コンパイラー を利用して、インテル® 64 および IA-32 アーキテクチャー 向けの C/C++ ソースファイルをコンパイルできます。
コンパイラーは、コマンドラインまたは Microsoft* Visual Studio* 統合開発環境から利用できます。

目次
- 準備
- コマンドラインの使用
- Microsoft* Visual Studio* の使用
- トレーニングとドキュメント
- 著作権と商標について

準備

コマンドラインの使用

Microsoft* Visual Studio* の使用

トレーニングとドキュメント

ドキュメント	説明
オンライン・トレーニング	インテル® コンパイラーのトレーニング情報を入手できます: https://software.intel.com/en-us/c-compilers/ipsxe-support/training (英語)。
リリースノート (英語)	製品のインストール、新機能と変更された機能、製品ドキュメントには記載されていない問題などが含まれます。 リンクをクリックするとインテル® デベロッパー・ゾーンにジャンプします。表示されるページの表で該当製品以下の Japanese リンクをクリックすると、日本語版のリリースノートが聞きます。
インテル® デベロッパー・ゾーンのサンプルとチュートリアル (英語)	サンプルプログラムと対応するチュートリアルです。コンパイラーの最適化、機能、ツール、プログラミングの概念を示すサンプルが含まれています。
インテル® デベロッパー・ゾーンのインテル® C++ コンパイラー 19.0 デベロッパー・ガイドおよびリファレンス (英語)	インテル® コンパイラー・ドキュメントには、次の情報が含まれます。 • コマンドラインまたは Microsoft* Visual Studio* の使用方法 • OpenMP* を含むテクノロジーとアーキテクチャーのサポート • オプション、プログラムの構造、クラスと数学ライブラリーなどのコンパイラーに関する情報
インテル® ソフトウェア・ドキュメント・ライブラリー (英語)	インテル® デベロッパー・ゾーンにある製品ドキュメントです。インテル® Parallel Studio XE は、「Product Suites」以下にリストされています。
ヘルプ	コマンドラインで icl /help と入力すると、コンパイラー・オプションに関する一般的なヘルプを参照できます。
日本語最新ドキュメント	日本語の最新ドキュメントを入手できます。
日本語最新情報	最新情報を日本語で参照できます。

© 2019 iSUS *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。

既存の開発環境へアドオンとしてインストール

Windows*:

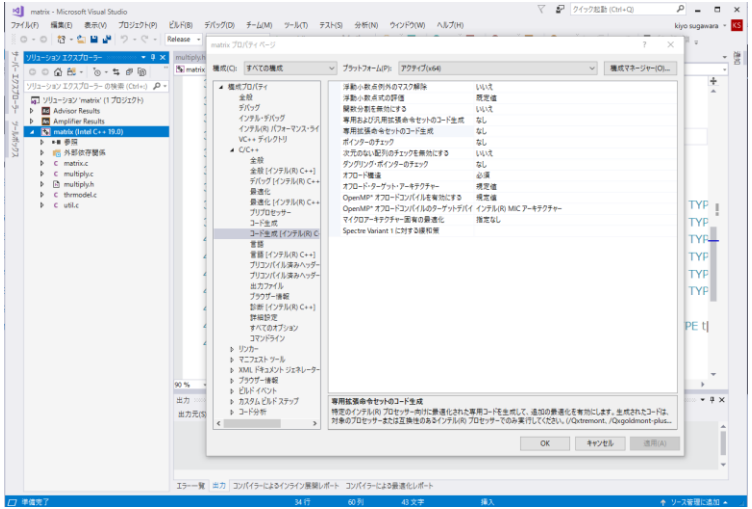
- Visual Studio* 2013/2015/2017

Linux*:

- gcc バージョン 4.3 から 8.x
- binutils バージョン 2.20 から 2.29
- Eclipse Platform 4.7/4.8
- CDT 9.2.x/9.3.x

macOS*:

- Xcode* 9.4 および 10



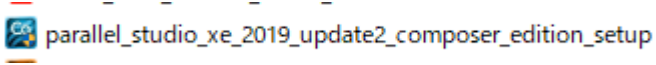
コマンドラインおよび IDE (統合開発環境) をサポートします

詳細は、リリースノート https://software.intel.com/en-us/articles/intel-cpp-compiler-release-notes で確認してください

© 2019 iSUS *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。

パッケージの入手とインストール

- インテル® Parallel Studio XE 2019 評価版または製品版パッケージを入手します
  - Windows* では、パッケージをダブルクリックしてインストール



  - Linux* では、install.sh または install_GUI.sh を起動してインストール

- デフォルトのインストール先
  - Windows*:
    C:\Program Files (x86)\IntelSWTools
  - Linux*:
    /opt/intel

インストール後
compilervars.bat /
compilervars.[sh/csh] を実行
して環境を設定できます

© 2019 iSUS *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。

インテル® Parallel Studio XE のコンポーネント
包括的なソフトウェア開発ツールスイート

Composer Edition

ビルド
コンパイラーとライブラリー

インテル® C/C++、Fortran コンパイラー

インテル® MKL[1]

インテル® DAAL[2]

インテル® TBB[3]
C++ スレッド・ライブラリー

インテル® IPP[4]
画像、信号、データ処理

インテル® Distribution for Python*
ハイパフォーマンスな Python*

Professional Edition

解析
解析ツール

インテル® VTune™ Amplifier
パフォーマンス・プロファイラー

インテル® Inspector
メモリー/スレッドのデバッガー

インテル® Advisor
ベクトル化の最適化、スレッドのプロトタイプ生成、フローグラフ解析

Cluster Edition

スケール
クラスターツール

インテル® MPI ライブラリー
メッセージ・パッシング・インターフェイス・ライブラリー

インテル® Trace Analyzer & Collector
MPI チューニングと解析

インテル® Cluster Checker
クラスター診断エキスパート・システム

オペレーティング・システム: Windows*、Linux*、macOS*[5]

インテル® アーキテクチャー・ベースのプラットフォーム

© 2019 iSUS *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。

[1] インテル® マス・カーネル・ライブラリー
[2] インテル® データ・アナリティクス・アクセラレーション・ライブラリー
[3] インテル® スレッディング・ビルディング・ブロック
[4] 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
[5] インテル® Parallel Studio XE Composer Edition でのみ利用可能

インテル® Parallel Studio XE コンポーネント・ツール

ビルド

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® Distribution for Python*
インテル® MKL
インテル® IPP
インテル® TBB
インテル® DAAL
Composer Edition に含まれる

解析

インテル® VTune™ Amplifier
インテル® Advisor
インテル® Inspector

Professional Edition に含まれる

スケール

インテル® MPI ライブラリー
インテル® Trace Analyzer & Collector
インテル® Cluster Checker

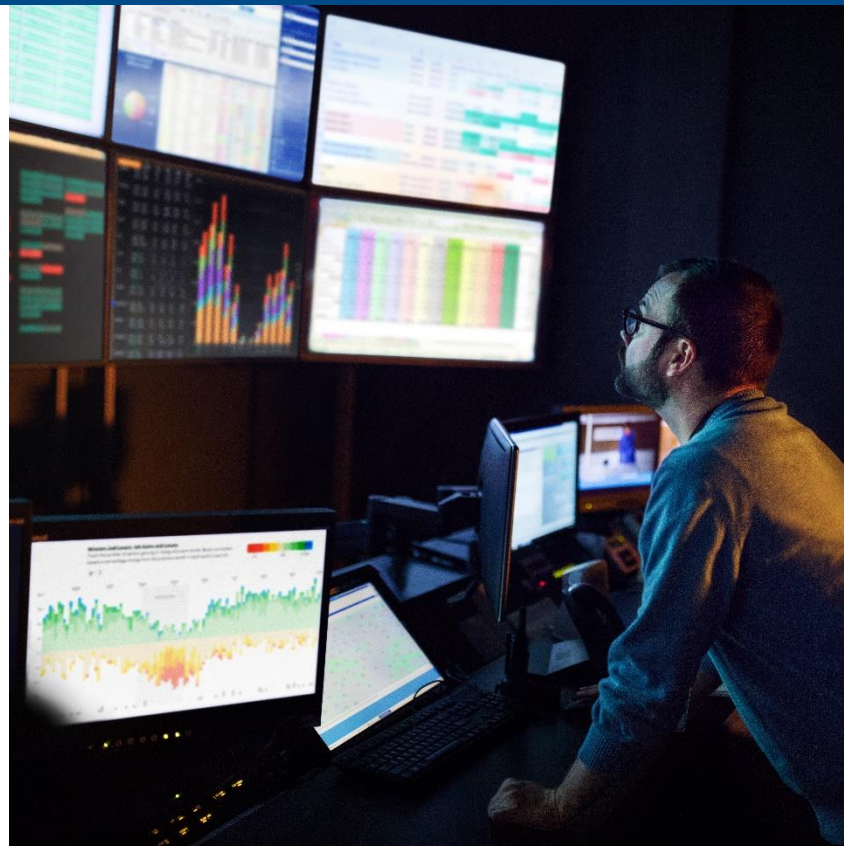
Cluster Edition に含まれる

インテル® コンパイラーによる高速でスケーラブルな並列コード

最先端の C/C++ および Fortran コード・パフォーマンスを実現、最新のインテル® プロセッサーの能力を最大限に活用

- インテル® Xeon® プロセッサーを含むインテル® アーキテクチャー向けに最適化およびベクトル化されたコードを開発
- 最新の言語標準と OpenMP* 標準規格の利用、主要なコンパイラーおよび IDE との互換性

詳細: isus.jp/c-compilers/



© 2019 iSUS *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。

インテル® コンパイラー 2019 (19.0) の新機能

すべてのバージョンに共通の更新内容

- インテル® アーキテクチャーのサポートを強化 — Intel Atom® プロセッサーからインテル® Xeon® スケーラブル・プロセッサーまで、広範なインテル® アーキテクチャー向けに最適化されたコードを生成
- 優れた並列パフォーマンスを実現 — (OpenMP* を使用した) ベクトル化とスレッド化により、インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令を含む最新の SIMD 対応ハードウェアを活用

C++ の新機能

C++17 標準機能の追加サポート

- ラムダ式と定数式のサポートを向上
- GNU* C++ コンパイラーおよび Microsoft* コンパイラーとの互換性を向上

標準規格に基づく並列化により C++ 開発者を支援

- OpenMP* 5.0[1] の一部をサポート
- 最新の並列化仕様によりコードを現代化

Fortran の新機能

Fortran 2018 の大部分をサポート

- Co-Array 機能: EVEMTS および COSHAPE
- IMPORT 文の拡張
- デフォルトのモジュールアクセス

OpenMP* 4.5 の完全サポート (ユーザー定義のリダクションに対応)

- ランタイムに配列の適合性をチェックする check shape オプション

[1]OpenMP* 5.0 はドラフト

© 2019 iSUS *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。

サポートされる言語仕様

- インテル® C++ コンパイラーでサポートされる C++17 の機能
  https://www.isus.jp/products/c-compilers/c17-features-supported-by-c-compiler/
- インテル® C++ コンパイラーでサポートされる C++14 の機能
  https://www.isus.jp/products/c-compilers/c14-features-supported-by-intel-c-compiler/
- インテル® C++ コンパイラーでサポートされる C++11 の機能
  https://www.isus.jp/products/c-compilers/c0x-features-supported-by-intel-c-compiler/
- インテル® C++ コンパイラーでサポートされる C99 の機能
  https://www.isus.jp/products/c-compilers/c99-support-in-intelr-c-compiler/
- インテル® C++ コンパイラーにおける C11 のサポート
  https://www.isus.jp/products/c-compilers/c11-features-supported-by-intel-c-compiler/
- Fortran 2003 仕様のサポート
  https://www.isus.jp/products/fortran-compilers/fortran-2003-support/
- Fortran 2008 仕様のサポート
  https://www.isus.jp/products/fortran-compilers/fortran-2008-support/

© 2019 iSUS *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。

インテル® コンパイラー: 開発者に大きな価値を提供

ソフトウェア開発者にとって重要なポイント		課題	インテル® コンパイラーの利点
	パフォーマンス 高速なアプリケーションを開発する必要がある	最新のハードウェア・イノベーションを利用しなければならない	最新の x86 互換プロセッサーと命令セットの能力を最大限に引き出すことができる
	生産性 生産性を向上でき、使いやすくなければならない	主要な言語とプログラミング・モデルがサポートされていなければならない	最新の Fortran、C/C++、OpenMP* 標準規格をサポートし、主要なコンパイラーおよび IDE と互換性がある
	スケーラビリティー アプリケーションをローカルで開発・デバッグし、グローバルに配置する必要がある	急速に増え続けるコア数とベクトル幅に合わせてコードを保守しなければならない	新しい世代のプロセッサーでもコードを変更することなくスケーラブルなパフォーマンスを実現できる

© 2019 iSUS *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。

gcc との互換性

- インテル® C++ コンパイラーは、GNU* コンパイラー・コレクション (gcc) のほとんどのバージョンと互換性があります。リリースノートに互換バージョンの一覧があります

- インテル® C++ コンパイラーは GNU* コンパイラーが提供する多くの言語拡張をサポートしています。詳細は、http://www.gnu.org/home.ja.html を参照してください

- インテル® C++ コンパイラーと GNU* gcc コンパイラーは、事前定義済みマクロをサポートします (__GNUC__、__GNUG__、__GNUC_MINOR、__GNUC_PATCHLEVEL など)

© 2019 iSUS *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。

Visual C++ との互換性

- インテル® C++ コンパイラーは、Microsoft* Visual C++* コンパイラーとソース互換およびバイナリー互換 (ネイティブコードのみ) です。Microsoft* Visual Studio* 内で、インテル® C++ コンパイラーでビルドされたバイナリービルドをデバッグすることができます

- インテル® C++ コンパイラー V19 は、Microsoft* Visual Studio* 2013、2015、および 2017 プロジェクトと互換性があります

Microsoft* Visual Studio* との統合

インテル® C++ コンパイラーは、Microsoft* Visual Studio* 2013、2015、および 2017 プロジェクトと互換性があります。

インテル® C++ コンパイラーは、Microsoft* Visual Studio* で提供されるプロジェクトの種類でネイティブ C++ プロジェクトのみをサポートしています。以下のような .NET 属性を備えたプロジェクト・ファイルはインテル® C++ プロジェクトには変換できません。

- 空のプロジェクト (.NET)
- クラス・ライブラリー (.NET)
- コンソール・アプリケーション (.NET)
- Windows* コントロール・ライブラリー (.NET)
- Windows* フォーム・アプリケーション (.NET)
- Windows* サービス (.NET)

サポートされていない主な機能

- COM 属性
- C++ AMP (C++ Accelerated Massive Parallelism)
- C++ マネージド拡張 (新しいプラグマ、キーワード、およびコマンドライン・オプション)
- イベント処理 (新しいキーワード)
- `__abstract` キーワード
- `__box` キーワード
- `__delegate` キーワード
- `__gc` キーワード
- `__identifier` キーワード
- `__nogc` キーワード
- `__pin` キーワード
- `__property` キーワード
- `__sealed` キーワード
- `__try_cast` キーワード
- `__w64` キーワード

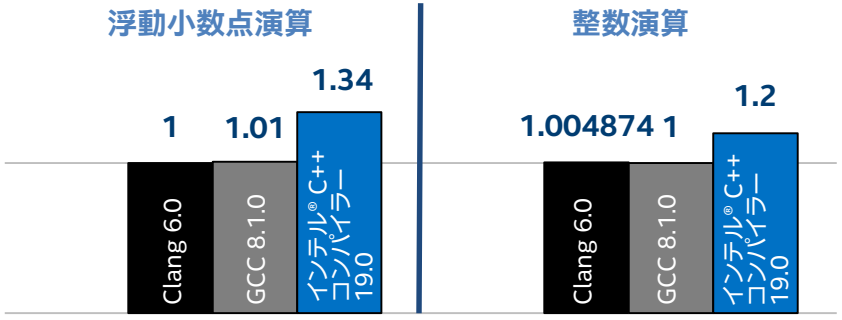
サポートされていないプリプロセッサーの機能

- `#import` ディレクティブの変更 (属性付きコードの場合)
- `#using` ディレクティブ
- `managed` プラグマ、`unmanaged` プラグマ
- `_MANAGED` マクロ
- `runtime_checks` プラグマ

© 2019 iSUS *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。

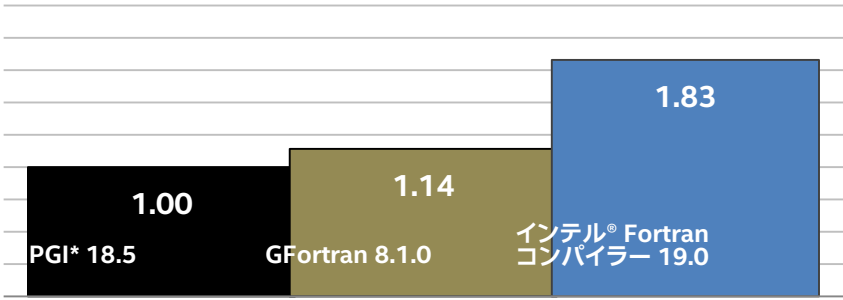
インテル® C++/Fortran コンパイラーによる優れたアプリケーション・パフォーマンス — Linux* (数値が大きいほど高性能)

インテル® C++ コンパイラーによる優れた C++ アプリケーション・パフォーマンス — Linux*



相対 (相乗平均) パフォーマンス SPEC* ベンチマーク (数値が大きいほど高性能)

インテル® Fortran コンパイラーによる優れた Fortran アプリケーション・パフォーマンス — Linux*



相対 (相乗平均) パフォーマンス Polyhedron* ベンチマーク (数値が大きいほど高性能)

パフォーマンス結果は 2018 年 8 月 26 日時点のテスト結果に基づいたものであり、公開されている利用可能なすべてのセキュリティー・アップデートが適用されていない可能性があります。詳細については、構成の開示を参照してください。絶対的なセキュリティーを提供できる製品はありません。性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。詳細については、パフォーマンス・ベンチマーク・テストの開示 (英語) を参照してください。

2018 年 8 月 26 日時点のインテルによるテスト。システム構成: ハードウェア: インテル® Xeon® Platinum 8180 プロセッサー @ 2.50GHz、384GB RAM、ハイパースレッディング有効。ソフトウェア: インテル® コンパイラー 19.0、GCC 8.1.0、PGI* 18.5、Clang/LLVM 6.0。Linux* OS: Red Hat* Enterprise Linux* Server 7.4 (Maipo)、3.10.0-693.el7.x86_64。SPEC* ベンチマーク (www.spec.org (英語))。SPECint* ベンチマーク測定時の CXX テストには SmartHeap* 10 を使用。SPECint*_rate_base_2017 コンパイラー・オプション: C++ テストには SmartHeap* 10 を使用。インテル® C/C++ コンパイラー 19.0: -xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-mem-layout-trans=3。GCC 8.1.0: -march=znver1 -mfpmath=sse -Ofast -funroll-loops -flto。Clang 6.0: -march=core-avx2 -mfpmath=sse -Ofast -funroll-loops –flto。SPECfp*_rate_base_2017 コンパイラー・オプション: インテル® C/C++ コンパイラー 19.0: -xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-prefetch -ffinite-math-only -qopt-mem-layout-trans=3。GCC 8.1.0: -march=skylake-avx512 -mfpmath=sse -Ofast -fno-associative-math -funroll-loops -flto。Clang 6.0: -march=znver1 -mfpmath=sse -Ofast -funroll-loops –flto。SPECint*_speed_base_2017 コンパイラー・オプション: C++ テストには SmartHeap* 10 を使用。インテル® C/C++ コンパイラー 19.0: -xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-mem-layout-trans=3 -qopenmp。GCC 8.1.0: -march=znver1 -mfpmath=sse -Ofast -funroll-loops -flto -fopenmp。Clang 6.0: -march=core-avx2 -mfpmath=sse -Ofast -funroll-loops -flto -fopenmp=libomp。SPECfp*_speed_base_2017 コンパイラー・オプション: インテル® C/C++ コンパイラー 19.0: -xCORE-AVX512 -ipo -O3 -no-prec-div -qopt-prefetch -ffinite-math-only -qopenmp。GCC 8.1.0: -march=skylake-avx512 -mfpmath=sse -Ofast -fno-associative-math -funroll-loops -flto -fopenmp。Clang 6.0: -march=skylake-avx512 -mfpmath=sse -Ofast -funroll-loops -flto -fopenmp=libomp。

2018 年 8 月 26 日時点のインテルによるテスト。システム構成: ハードウェア: インテル® Core™ i7-8700K プロセッサー @ 3.70GHz、64GB RAM、ハイパースレッディング有効。ソフトウェア: インテル® Fortran コンパイラー 19.0、PGI* Fortran 18.5、GFortran 8.1.0。Linux* OS: Red Hat* Enterprise Linux* Server 7.4 (Maipo)、3.10.0-693.el7.x86_64。Polyhedron* Fortran ベンチマーク (www.fortran.uk (英語))。Linux* コンパイラー・オプション: GFortran: -Ofast -mfpmath=sse -flto -march=haswell -funroll-loops -ftree-parallelize-loops=6。インテル® Fortran コンパイラー: -fast -parallel -xCORE-AVX2 -nostandard-realloc-lhs。PGI* Fortran: -fast -Mipa=fast,inline -Msmartalloc -Mfprelaxed -Mstack_arrays -Mconcur=bind -tp haswell。

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。注意事項の改訂 #20110804

© 2019 iSUS *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。

インテル® コンパイラーの最適化機能とオプション

ビルド

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® Distribution for Python*
インテル® MKL
インテル® IPP
インテル® TBB
インテル® DAAL
Composer Edition に含まれる

オプションの基本

インテル® コンパイラーは、gcc/gfortran、visual C++ の大部分のコンパイラー・オプションをそのまま受け付けます

makefile を使用する場合、コンパイラー定義変数 (CC/FC など) を置き換えるだけで使用できます:

- Windows*: icl、ifort
- Linux*/macOS*: icpc、icc、ifort

インテル® コンパイラー固有のほとんどのオプションは、
/Q (Windows*)、-q (Linux*,macOS*) で始まるオプションを指定します

© 2019 iSUS *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。

インテル® コンパイラーを導入する価値

```
Intel Compiler 19.0 Update 2 Intel(R) 64 Visual Studio 2017

Z:¥sample¥vs2017>cl prime.cpp -nologo -WO
prime.cpp

Z:¥sample¥vs2017>prime

     1から10000000までの範囲の素数の数は  664580個。計算時間   3.46秒

Z:¥sample¥vs2017>icl prime.cpp -nologo -WO
prime.cpp

Z:¥sample¥vs2017>prime

     1から10000000までの範囲の素数の数は  664580個。計算時間   1.52秒

Z:¥sample¥vs2017>icl prime.cpp -nologo -WO /QxHOST
prime.cpp

Z:¥sample¥vs2017>prime

     1から10000000までの範囲の素数の数は  664580個。計算時間   1.00秒

Z:¥sample¥vs2017>
```

- 最適なコンパイラー・オプションを見つけるには試行錯誤が必要
- 最もパフォーマンスに影響するオプションは?
- 他のプラットフォームでもスケーラブルか?

© 2019 iSUS *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。

次にすべきことは? – Visual C++

```
Intel Compiler 19.0 Update 2 Intel(R) 64 Visual Studio 2017
Z:¥sample¥vs2017>cl prime.cpp /Qvec-report:2 /O2
Microsoft(R) C/C++ Optimizing Compiler Version 19.16.27027.1 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

prime.cpp

--- 分析関数: bool __cdecl TestForPrime(int)
z:¥sample¥vs2017¥prime.cpp(29) : info C5002: ループはベクター化されません。理由:
 '500'

--- 分析関数: void __cdecl FindPrimes(int,int)
z:¥sample¥vs2017¥prime.cpp(29) : info C5002: ループはベクター化されません。理由:
 '500'
z:¥sample¥vs2017¥prime.cpp(41) : info C5002: ループはベクター化されません。理由:
 '1301'

--- 分析関数: main
z:¥sample¥vs2017¥prime.cpp(29) : info C5002: ループはベクター化されません。理由:
 '500'
z:¥sample¥vs2017¥prime.cpp(41) : info C5002: ループはベクター化されません。理由:
 '1301'
Microsoft (R) Incremental Linker Version 14.16.27027.1
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:prime.exe
```

- 一部のコンパイラーは、最適化レポートを生成します

ベクトル化されていない

© 2019 iSUS *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。

次にすべきことは? – インテル® C++

Intel Compiler 19.0 Update 2 Intel(R) 64 Visual Studio 2017

```
Z:¥sample¥vs2017>icl prime.cpp /Qopt-report:5 /O2 /W0 /nologo
icl: リマーク #10397: 最適化レポートは出力先の *.optrpt ファイルに生成されます。

prime.cpp
ループの開始 Z:¥sample¥vs2017¥prime.cpp(41,5) インライン展開後 Z:¥sample¥vs2017¥prime.cpp(57,5)
   リマーク #25096: ループ交換は完了しませんでした: ループの入れ子が完全ではありません (ソースのいずれかまた
   リマーク #25452: 元の順序はマージンが近いことを除けば適切です。
   リマーク #15542: ループ はベクトル化されませんでした: 内部ループがすでにベクトル化されています。

   ループの開始 Z:¥sample¥vs2017¥prime.cpp(29,5) インライン展開後 Z:¥sample¥vs2017¥prime.cpp(57,5)
      リマーク #15305: ベクトル化のサポート: ベクトル長 4
      リマーク #15309: ベクトル化のサポート: 正規化されたベクトル化のオーバーヘッド 0.025
      リマーク #15300: ループがベクトル化されました。
      リマーク #15475: --- ベクトルのコストサマリー開始 ---
      リマーク #15476: スカラーのコスト: 38
      リマーク #15477: ベクトルのコスト: 29.750
      リマーク #15478: スピードアップの期待値: 1.270
      リマーク #15482: ベクトル化された算術ライブラリーの呼び出し: 1
      リマーク #15488: --- ベクトルのコストサマリー終了 ---
   ループの終了

   ループの開始 Z:¥sample¥vs2017¥prime.cpp(29,5) インライン展開後 Z:¥sample¥vs2017¥prime.cpp(57,5)
   <ベクトル化の剰余ループ>
   ループの終了
ループの終了
```

- インテル® C++ ではベクトル化されている

© 2019 iSUS *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。

スケールしているか?

同一バイナリーを異なるプラットフォームで実行

Intel Compiler 19.0 Update 2 Intel(R) 64 Visual Studio 2017

```
Z:¥sample¥vs2017>icl prime.cpp /QxHOST /W0 /nologo
prime.cpp

Z:¥sample¥vs2017>prime

    1から10000000までの範囲の素数の数は　664580個。計算時間　1.00秒
```

システム

プロセッサ:	Intel(R) Cor
実装メモリ (RAM):	16.0 GB (15.
システムの種類:	64 ビット オペ

パフォーマンス

コアの数	8
スレッド数	8
プロセッサー・ベース動作周波数	3.60 GHz
ターボ・ブースト利用時の最大周波数	4.90 GHz
キャッシュ	12 MB SmartCache
バススピード	8 GT/s DMI3
TDP	95 W

Intel Compiler 19.0 Update 1 Intel(R) 64 Visual Studio 2017

```
Z:¥sample¥vs2017>prime

    1から10000000までの範囲の素数の数は　664

Z:¥sample¥vs2017>prime

    1から10000000までの範囲の素数の数は　664580個。計算時間　1.26秒
```

システム

プロセッサ:	Intel(R) Core
実装メモリ (RAM):	32.0 GB (31.
システムの種類:	64 ビット オペ

パフォーマンス

コアの数	6
スレッド数	12
プロセッサー・ベース動作周波数	3.50 GHz
ターボ・ブースト利用時の最大周波数	4.00 GHz
キャッシュ	8.25 MB L3
バススピード	8 GT/s DMI3
QPI リンク数	0
TDP	140 W

© 2019 iSUS *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。

スケーラブルなバイナリー

同一バイナリーを異なるプラットフォームで実行

システム

プロセッサ:	Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz 3.60 GHz
実装メモリ (RAM):	16.0 GB (15.8 GB 使用可能)
システムの種類:	64 ビット オペレーティング システム、x64 ベース プロセッサ

Intel Compiler 19.0 Update 2 Intel(R) 64 Visual Studio 2017

```
Z:¥sample¥vs2017>icl prime.cpp /QaxAVX2,COMMON-AVX512 /WO /nologo
prime.cpp

Z:¥sample¥vs2017>prime

    1から10000000までの範囲の素数の数は  664580個。計算時間   1.00秒
```

システム

プロセッサ:	Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz 3.50 GHz
実装メモリ (RAM):	32.0 GB (31.7 GB 使用可能)
システムの種類:	64 ビット オペレーティング システム、x64 ベース プロセッサ

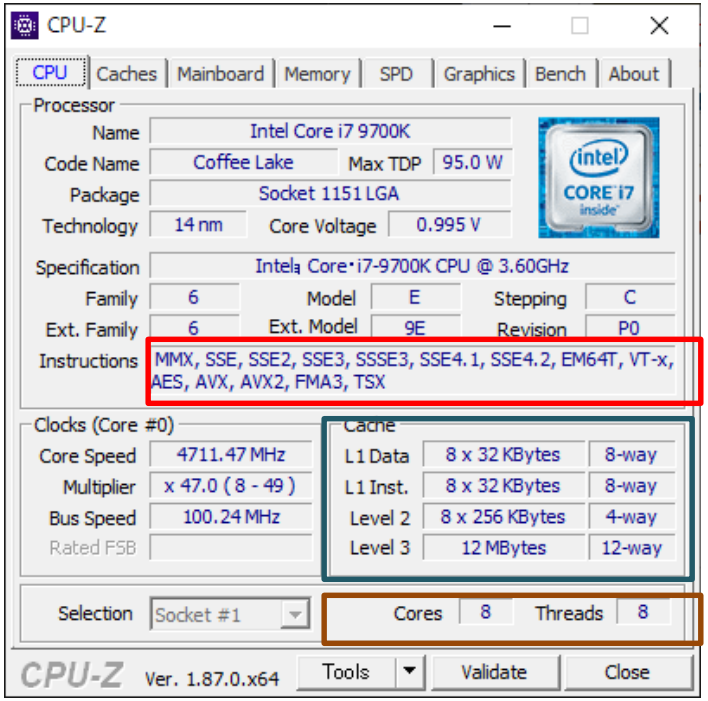
Intel Compiler 19.0 Update 1 Intel(R) 64 Visual Studio 2017

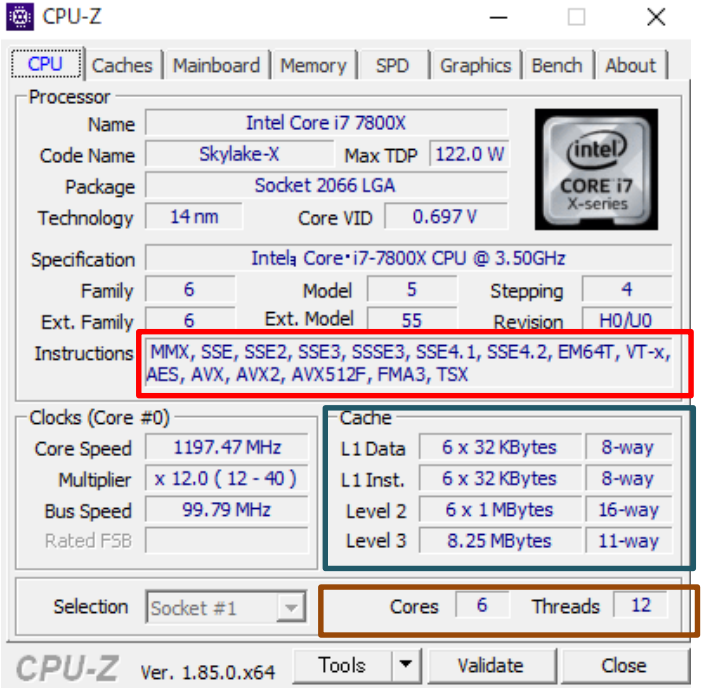
```
Z:¥sample¥vs2017>prime

    1から10000000までの範囲の素数の数は  664580個。計算時間   0.80秒
```


© 2019 iSUS *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。

最適化にはマイクロアーキテクチャーの違いを理解する必要がある

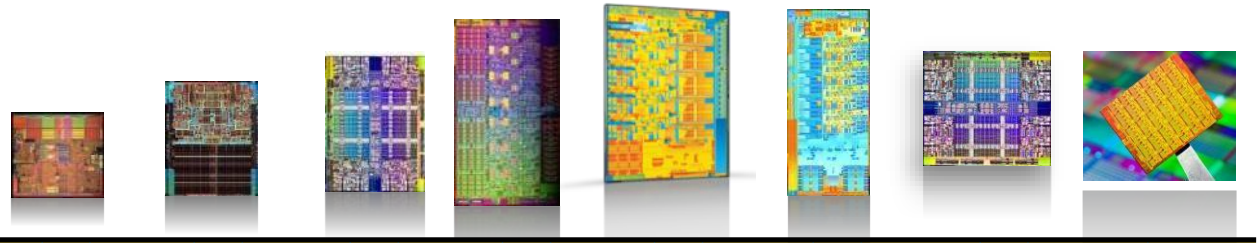




© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Parallel + SIMD は前進への鍵

インテル® Xeon® プロセッサーは、両者とも並列性を高めています



	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー コード名 Woodcrest EP	インテル® Xeon® プロセッサー コード名 Nehalem EP	インテル® Xeon® プロセッサー コード名 Westmere EP	インテル® Xeon® プロセッサー コード名 Sandy Bridge EP	インテル® Xeon® プロセッサー コード名 Ivy Bridge EP	インテル® Xeon® プロセッサー コード名 Haswell EP	インテル® Xeon® プロセッサー コード名 Skylake Server[1]
コア	1	2	4	6	8	12	18	28
スレッド	2	2	8	12	16	24	36	56
SIMD 幅	128	128	128	128	256	256	256	512

さらに多いコア
さらに多くのスレッド
より広いベクトル

OpenMP* は parallel + SIMD を前進させる最も重要な機能の 1 つ

* ark.intel.com で公開されている出荷済の製品仕様

© 2019 iSUS *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。

高レベルの最適化

icc/icl -O による基本的な最適化

-O0　最適化なし、デバッグ用に -g を設定

-O1　スカラーの最適化
コードサイズが増える最適化を除く

-O2　icc/icpc/icl の**デフォルト** (-g, /debug を除く)
自動ベクトル化、一部のループ変換 (アンロール、ループ交換など) を含むソースファイル内のインライン展開
このオプションで開始 (-O0, /Od でデバッグした後)

-O3　より積極的なループの最適化
キャッシュ・ブロッキング、ループ融合、プリフェッチなどを含浮動小数点演算を多用するループや大きなデータセットを処理するループを含むアプリケーションに最適

© 2019 iSUS *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。

数学ライブラリー

icc/icl はインテルの最適化された数学ライブラリーを利用

- libimf (スカラー) および libsvml (スカラー/ベクトル)
- GNU* libm よりも高速
- ドライバーは libm の前に libimf を自動的にリンク
- 追加の関数 (math.h を mathimf.h に置換)

libm を明示的にリンクしない! -lm

- リンクすると遅い libm の関数が使用される
- インテルのドライバーは libm を明示的にリンクしない
- gcc は -lm を使用するため古い makefile によく含まれている

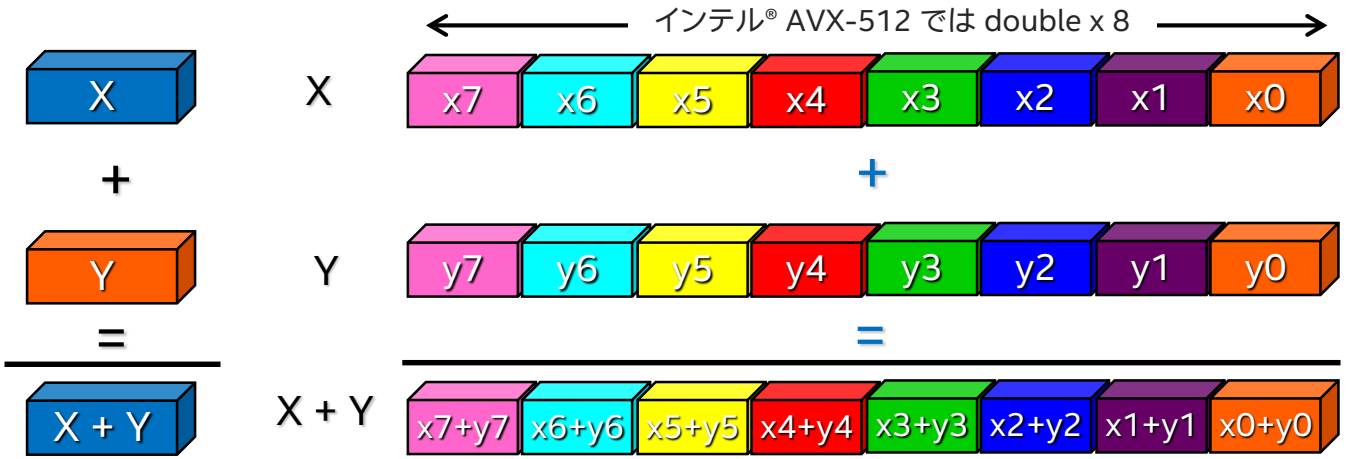
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ベクトル命令とベクトル化は重要

SIMD: (Single Instruction Multiple Data)

```
for (i=0; i<n; i++) z[i] = x[i] + y[i];
```

- ❑ スカラーモード
  - 1 つの命令で 1 つの結果を生成
  - 例: vaddss (vaddsd)
- ❑ ベクトル (SIMD) モード
  - 1 つの命令で複数の結果を生成可能
  - 例: vaddps (vaddpd)



© 2019 iSUS *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。

ベクトル化

ベクトル化はパックド SIMD 命令を使用したループを生成

さまざまな選択肢:

- **自動ベクトル化:** コンパイラーがすべて処理、O2 以上を指定
  - コンパイラーが正当性を保証する必要がある
- **アシスト付きベクトル化:** ユーザーがプラグマの使用、言語構造やソースコードの変更により追加のヒントを提供する
- **明示的なベクトル化:** ユーザーが OpenMP* プラグマを使用してベクトル化を指示する
  - ユーザーが正当性を保証する必要がある
  - コンパイラーが自動ベクトル化できない場合やループのベクトル化を制御する

© 2019 iSUS *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。

命令セットが拡張されるときに何が起こるのか？

SIMD レジスターの幅が同じ場合 (SSE2 -> SSE4.2)

- 新しい命令や機能をすぐに利用できます

SIMD レジスターの幅が異なる場合 (SSE4.2 -> AVX)

- コンパイラーやプロセッサーが利用できても、OS が新しい命令セットをサポートするのを待つ必要があります

… ただし …

SIMD 幅にスケーラブルなコードを作成しておきましょう

© 2019 iSUS *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。

インテル® コンパイラーのベクトル化オプション

自動ベクトル化

/Qax<SIMD 命令セット,…> (-ax<SIMD,…>)
/Qx<SIMD 命令セット> (-x<SIMD>)
/QxHOST (-xHOST)

SIMD **命令セットに指定可能なキーワード:**

SSE2, SSE3, SSSE3, SSE4.1, SSE4.2,	(128 ビット)
ATOM_SSSE3, ATOM_SSE4.2,	(128 ビット)
AVX, CORE-AVX-I, CORE-AVX2,	(256 ビット)
MIC-AVX512, CORE-AVX512, COMMON-AVX512	(512 ビット)

/O2 (-O2) 以上が指定されると自動ベクトル化が有効となる:
これらのオプションが指定されない場合のデフォルト命令セットは **SSE2**

© 2019 iSUS *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。

サポートされるプロセッサー固有のコンパイラー・オプション

インテル® プロセッサーのみ	他社製プロセッサーを含む (-m は gcc でも指定可能)
-xsse2, /Qxsse2	-msse2 (デフォルト), /arch:sse2
-xsse3, /Qxsse3	-msse3, /arch:sse3
-xssse3, /Qxssse3	-mssse3, /arch:ssse3
-xsse4.1, /Qxsse4.1	-msse4.1, /arch:sse4.1
-xsse4.2, /Qxsse4.2	-msse4.2, /arch:sse4.2
-xavx. /Qxavx	-mavx, /arch:avx
-xcore-avx2, /Qxcore-avx2	
-xcore-avx512, /Qxcore-avx512	
-xHost, /QxHOST	-xHost (-march=native)
インテルの CPUID をチェック	CPUID をチェックしない
サポートしていないプロセッサーで実行した場合はメッセージを表示	サポートしていないプロセッサーで実行した場合は不正な命令エラーになる

© 2019 iSUS *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。

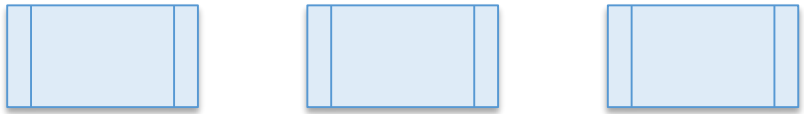
ベクトル化オプションはパフォーマンスに最も影響する

/Qx (-x) と /Qax (-ax)

すべてのソースファイルに同じオプションを指定することを推奨

… しかし困難な場合がある …

オブジェクト・ファイルが提供される、事情によりオプションが制限される



main.c　　func1.c　　func2.obj (または特定のオプションが指示される)

```
> icl main.c func1.c func2.obj /QaxCORE-AVX2 /Fetest_code.exe
  または
> icl func2.c /QxCORE-AVX512 /c
> icl main.c fuinc1.c func2.obj /QaxCORE-AVX2 /Fetest_code.exe
```

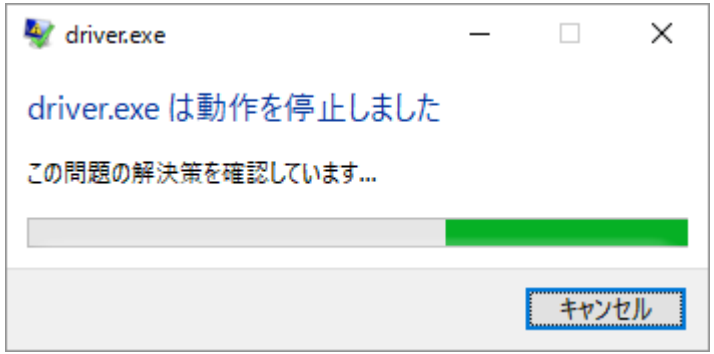
実行すると?

© 2019 iSUS *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。

ベクトル化オプションはパフォーマンスに最も影響する (2)

/Qx (-x) と /Qax (-ax)

問題となる: ターゲットの実行環境が最新のプロセッサーではない、
インテル® AVX/AVX2 からインテル® AVX-512 への移行中のコードが含まれる…



必ずしも例外が発生するとは限らない ……

ベクトル化できなければ
インテル® AVX-512 命令は
生成されない

コンパイラーのレポートやインテル® **Advisor** を使用して、
ベクトル化されているかどうかを確認

© 2019 iSUS *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。

ベクトル化オプションはパフォーマンスに最も影響する (3)

ランタイムチェックの影響

```
C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl multiply.c /c /QaxCORE-AVX512 /QxCO
RE-AVX2 /Qopenmp /nologo
multiply.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl driver.c multiply.obj /QaxCORE-AVX2
 /nologo
driver.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>driver

ROW:101 COL: 101
Execution time is 3.269 seconds
GigaFlops = 6.241052
Sum of result = 195853.999899
```

```
C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl multiply.c /c /QxCORE-AVX2 /Qopenmp
 /nologo
multiply.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>icl driver.c multiply.obj /QaxCORE-AVX2
 /nologo
driver.c

C:¥Users¥kiyo¥Desktop¥hakata¥code¥matmul>driver

ROW:101 COL: 101
Execution time is 2.872 seconds
GigaFlops = 7.103760
Sum of result = 195853.999899
```

> icl multiply.c /c /QaxCORE-AVX512 /QxCORE-AVX2
> icl driver.c multiply.obj /QaxCORE-AVX2 /Fedriver.exe

> icl multiply.c /c /QxCORE-AVX2
> icl driver.c multiply.obj /QaxCORE-AVX2 /Fedriver.exe

処理中に数万回、数十万回呼び出されるような下位の関数では、ランタイムチェックを行うべきではない

© 2019 iSUS *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。

ベクトル化オプションはパフォーマンスに最も影響する (4)

バイナリーに適用されている命令セットは?

プログラムのメトリック

経過時間	0.42秒	合計 INT+FLOAT ギガ OPS	5.89
ベクトル命令セット	AVX2, AVX	合計 GFLOPS	5.16
CPU スレッド数	8	合計 GINTOPS	0.64

パフォーマンス特性

メトリック	合計	
合計 CPU 時間	0.71秒	100.0%
1 でベクトル化されたループ の時間	0.50秒	71.5%
スカラーコード時間	0.20秒	28.5%

ベクトル化のゲイン/効率

ベクトルループのゲイン/効率	5.15x	100%
プログラムのおよそのゲイン	3.97x	

ベクトル命令セットに
インテル® AVX2 が適用されている

サーベイレポートで
詳細を確認

ループライン

関数呼び出しサイトとループ		パフォーマンス...	セルフ時間	合計時間	タイプ	ベクトル化できない理由	ベクトル化されたループ			
							ベクトル ISA	効率	予測ゲイン	VL (ベクトル長)
[ループ、mem.c:41 の mainompparallel_for@	☑		0.504s	0.504s	ベクトル化 (本体)		AVX2	100%	5.15x	4
[ループ、mem.c:41 の mainompparallel_for	☑		0.504s 99.3%	0.504s	ベクトル化 (本体)		AVX2			4
[ループ、mem.c:40 の mainompparallel_for@3	☐		0.003s	0.508s	スカラー	loop was not vectorized: inne...				
[ループ、mem.c:39 の mainompparallel_for@3	☐		0.000s	0.508s	スレッド化 (Open...	loop was not vectorized: inne...				

© 2019 iSUS *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。

コード中で CPU の命令セットをチェックする

```
#include <immintrin.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
  const unsigned long knl_features =
      (_FEATURE_AVX512F | _FEATURE_AVX512ER |
       _FEATURE_AVX512PF | _FEATURE_AVX512CD );
  if ( _may_i_use_cpu_feature( knl_features ) )
    printf("This CPU supports AVX-512F+CD+ER+PF as introduced in Knights Landing¥n");
  else
    printf("This CPU does not support all Knights Landing AVX-512 features¥n");
  return 1;
}
```

インテル® コンパイラーは、プロセッサーの機能をチェックする
_may_i_use_cpu_feature() 組込み関数をサポート

https://www.isus.jp/specials/target-intel-core-processors/
https://www.isus.jp/products/c-compilers/how-to-detect-knl-instruction-support/

© 2019 iSUS *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。

ベクトル化に影響する 6 つの要因

ループ伝搬依存

```
DO I = 1, N
   A(I+1) = A(I) + B(I)
ENDDO
```

関数呼び出し

```
for (i = 1; i < nx; i++) {
  x = x0 + i * h;

     sumx = sumx + func(x, y, xp);
}
```

ポインター・エイリアシング

```
void scale(int *a, int *b){
   for (int i = 0; i < 1000; i++)
         b[i] = z * a[i];
}
```

不明なループカウント

```
struct _x { int d; int bound; };

void doit(int *a, struct _x *x)
{
  for(int i = 0; i < x->bound; i++)
    a[i] = 0;
}
```

間接メモリーアクセス

```
 for (i=0; i<N; i++)
            A[B[i]] = C[i]*D[i]
```

外部ループ

```
for(i = 0; i <= MAX; i++) {
  for(j = 0; j <= MAX; j++) {
    D[i][j] += 1;
  }
}
```

さらに ....

© 2019 iSUS *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。

依存性 – ベクトル化を妨げる要因

ベクトル化は SIMD レーンが別の (前の) レーンの結果に依存しない場合にのみ安全

```
for (int i = 0; i < N; i++)
  a[i + 2] = a[i] + c;
```

この例では、i=2,a[4] の結果は i=0,a[2] の結果に依存している
これは「ループ伝播」、「ベクトル」または「フロー」の依存性
a[4] と a[2] の両方が同じ SIMD 命令で使用された場合、
a[4] を計算するときに a[2] の結果がまだ利用できないため、
a[2] のオリジナルの値が使用され、a[4] の結果は不正な値になる

コンパイラーはこのループを自動ベクトル化しないため、
手動でベクトル化する必要がある

© 2019 iSUS *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。

コンパイラーは保守的

すべての潜在的な依存性が実際に影響するとは限らない

```
void scale(int *a, int *b)
{
    for (int i = 0; i < 10000; i++) b[i] = a[i] + 4;
}
```

- コンパイラーはポインター a と b が「エイリアス」されると仮定する
- データ依存性にはループのマルチバージョンが役立つ
- コンパイラーがベクトル化できるように支援する
  - -fargument-noalias、/Qalias-args- を指定してコンパイルする
  - restrict キーワードを使用する: void scale(int *a, int *restrict b)
  - for ループの前に #pragma ivdep を挿入

© 2019 iSUS *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。

コンパイラーのレポート – 最適化レポート

最適化レポートの詳細レベルの指定

- -qopt-report[=n], /Qopt-report[:n]

最適化レポートの出力の指定

- -qopt-report=<filename>, /Qopt-report:<filename>
- オプションを指定しない場合、<filename>.optrpt ファイルが生成される

特定のフェーズのみの最適化レポート

- -qopt-report-phase[=list], /Qopt-report-phase[:list]
  指定できるフェーズ:
  - all – すべてのフェーズの最適化レポート (デフォルト)
  - loop – 入れ子のループおよびメモリーの最適化
  - vec – 自動ベクトル化および明示的なベクトル・プログラミング
  - par – 自動並列化
  - openmp – OpenMP* を使用したスレッド化
  - ipo – プロシージャー間の最適化 (インライン展開を含む)
  - pgo – プロファイルに基づく最適化
  - cg – コード生成

© 2019 iSUS *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。

プロシージャー間の最適化 (IPO)

マルチパスの最適化

icc –ipo, icl /Qipo

関数/ソースファイル境界を解析および最適化

- 関数のインライン展開、定数の伝播、依存性解析、データ/コードレイアウトなど

2 ステップのプロセス

- コンパイルステップ – オブジェクトは中間表現を含む
- リンクステップ – すべてのオブジェクトをコンパイルして最適化
- シームレス: リンカーは ipo およびコンパイラー・オプションを指定してビルドされたオブジェクトを自動検出
- 場合によっては、ビルド時間とバイナリーサイズが増えることがある
- ipo=n を指定してビルドを並列化できる
- ホットなモジュールのみビルド (プログラム全体をビルドする必要はない)

多くの小さな関数を含むアプリケーションで特に効果的

opt-report-phase=ipo を指定してインライン展開された関数のレポートを取得

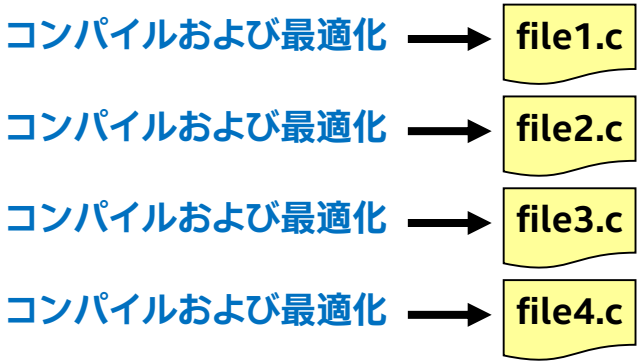
© 2019 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

プロシージャー間の最適化 (IPO)

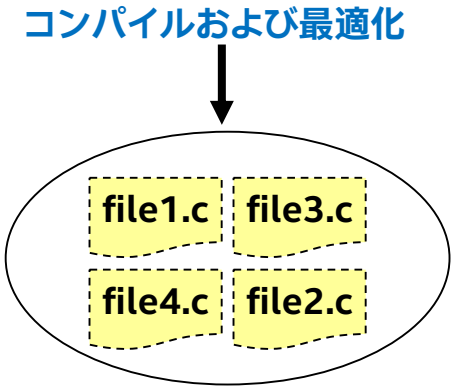
ファイル境界の最適化を拡張

`-ip, /Qip`	1 つのソースファイルのモジュール間のみ
`-ipo, /Qipo`	複数のファイル/アプリケーション全体のモジュール

IPO なし



IPO あり



© 2019 iSUS *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。

ベクトル化を評価するコンパイラーの機能

/Qvec- (-no-vec) オプション

- ベクトル化によりアプリケーションがどれくらい恩恵を得られているか簡単に調査
- /Qvec- (-no-vec) オプションが指定されるとコンパイラーは、SIMD 命令を使用するが、ベクトル化せずにスカラー操作を行うコードを生成
- /Qvec- (-no-vec) ありと、なしのバイナリーを作成してパフォーマンスを比較

© 2019 iSUS *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。

インテル® SSE2

Intel Compiler 17.0 Update 2 Intel(R) 64 Visual Studio 2015

```
> icl driver.c multiply.c /Qipo /nologo
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 2.436 seconds
GigaFlops = 8.375205
Sum of result = 195853.999899

> icl driver.c multiply.c /Qipo /nologo /Qvec-
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 6.534 seconds
GigaFlops = 3.122436
Sum of result = 195853.999899
```

インテル® AVX2

Intel Compiler 17.0 Update 2 Intel(R) 64 Visual Studio 2015

```
> icl driver.c multiply.c /Qipo /nologo /QxCORE-AVX2
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 1.935 seconds
GigaFlops = 10.543669
Sum of result = 195853.999899

> icl driver.c multiply.c /Qipo /nologo /QxCORE-AVX2 /Qvec-
driver.c
multiply.c

> driver

ROW:101 COL: 101
Execution time is 5.052 seconds
GigaFlops = 4.038401
Sum of result = 195853.999899
```

デフォルトの状態を検証

© 2019 iSUS *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。

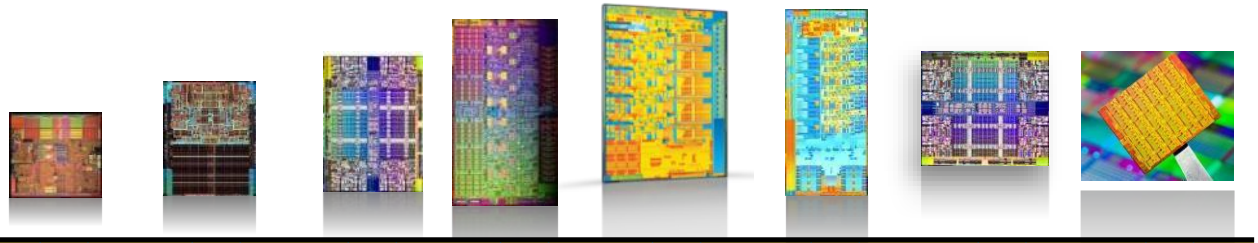
ベクトル化可能なコードを記述するためのガイドライン

- **単純な for ループを使用する** (トリップカウントがループの入り口で判明するようにする)
- **前のループ反復に依存しないようにする**
- **分かりやすいコードを記述する** (次の表記はできるだけ避ける)
  - 多くの関数呼び出し (インライン展開された/単純な数学関数を除く)
  - マスク付きの代入として処理できない分岐
- **ポインターの代わりに配列を使用する**
  - ヒントがないと、コンパイラーはポインターを含むコードを安全にベクトル化できるかどうか判断できない
  - カウンターをインクリメントして配列アドレスに使用する代わりに、ループ・インデックスを配列インデックスで直接使用する
- **効率的なメモリーアクセスを使用する**
  - 内部ループとユニットストライドを使用する (連続メモリーアクセス)
  - 間接アドレス指定は最小限に抑える (b[i] = a[index[i]]; など)
  - できるだけ一貫した方法でデータをアライメントする (インテル® AVX-512 では 64 バイト境界)

© 2019 iSUS *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。

Parallel + SIMD は前進への鍵

インテル® Xeon® プロセッサーは、両者とも並列性を高めています



	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー コード名 Woodcrest EP	インテル® Xeon® プロセッサー コード名 Nehalem EP	インテル® Xeon® プロセッサー コード名 Westmere EP	インテル® Xeon® プロセッサー コード名 Sandy Bridge EP	インテル® Xeon® プロセッサー コード名 Ivy Bridge EP	インテル® Xeon® プロセッサー コード名 Haswell EP	インテル® Xeon® プロセッサー コード名 Skylake Server[1]
コア	1	2	4	6	8	12	18	28
スレッド	2	2	8	12	16	24	36	56
SIMD 幅	128	128	128	128	256	256	256	512

さらに多いコア
さらに多くのスレッド
より広いベクトル

OpenMP* は parallel + SIMD を前進させる最も重要な機能の 1 つ

* ark.intel.com で公開されている出荷済の製品仕様

© 2019 iSUS *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。

自動並列化

- OpenMP* ランタイムベース
- オプションを指定すると、コンパイラーはループを等価なマルチスレッド・コードに自動変換

  -parallel

- 自動パラレライザーは並列で安全に実行できる単純な構造のループを検出して、それらのループのマルチスレッド・コードを自動的に生成
- 次のオプションを指定すると、自動パラレライザーのレポートにコンパイラーが並列化したプログラムのセクションに関する情報が含まれる

  -qopt-report-phase=par

© 2019 iSUS *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。

OpenMP* API

ほぼ 20 年来の技術計算/HPC における共有メモリー並列プログラミング (C/C++ と Fortran) 向けの業界標準:

- 最新の仕様: OpenMP* 4.5 (2015 年 11 月)
- ディレクティブ (C/C++ ではプラグマ) ベース
- ベンダーとプラットフォームにわたる移植性 - インテルは OpenMP* ARB (アーキテクチャー・レビュー委員会) のメンバーであり、OpenMP* のサポートを完全にコミット
- 各種並列性をサポート: スレッド、タスク、SIMD、オフロード
- インクリメンタルな並列処理をサポート

仕様ドキュメント、サンプル、および各種情報は www.openmp.org を参照してください

© 2019 iSUS *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。

コンパイラーが OpenMP* をサポートするか?

- OpenMP* は最も簡単なマルチスレッド・プログラミング・モデルであるが、コンパイラーが OpenMP* をサポートしている必要がある
- OpenMP* をサポートするコンパイラーは、_OPENMP マクロに仕様が公開された年月を数値で返す

仕様のバージョン	値	代表的なコンパイラー
OpenMP* 5.0 TR4	201611	インテル® コンパイラー V18.0
OpenMP* 4.5	201511	インテル® コンパイラー V17.0 、gcc 6.1
OpenMP* 4.0	201307	インテル® コンパイラー V14.0 、gcc 4.9 (offload 5.1)
OpenMP* 3.1	201107	インテル® コンパイラー V12.1 、gcc 4.7
OpenMP* 3.0	200805	インテル® コンパイラー V11.0 、gcc 4.4
OpenMP* 2.5	200505	インテル® コンパイラー V9.0 、gcc 4.2

© 2019 iSUS *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。

ソフトウェアのベクトル化およびスレッド化なしでは、パフォーマンスは向上しない: ハードウェアの世代ごとにその差は大きくなる

ベクトル化での経験:

- インテル® AVX2 やインテル® AVX-512 向けの再コンパイルでは、ゲインはわずか
- どこをベクトル化するか?
- 新しいアーキテクチャー向けに組込み関数を使用すべきか?
- コンパイラー・レポートの内容が分からない?

スレッド化での経験:

- アプリケーションをスレッド化してもパフォーマンスがそれほど向上しない?
- "スケーラビリティーの限界" に達したのか?
- 同期による解放の遅延、エラーか?

「**インテル® Advisor** のベクトル化アドバイザーは、コードのパフォーマンス解析のギャップを埋めてくれます。これは、最新のプロセッサーとコプロセッサーのベクトル化機能を上手く利用できるよう開発者を誘導してくれます。」

Leibniz Supercomputing Centre
科学計算エキスパート
Luigi Iapichino 博士

「**インテル® Advisor** により、並列化候補のプロトタイプを素早く生成し、開発者の時間と労力を節約することができました。」

Sandia National Laboratories
シニア・テクニカル・スタッフ
Simon Hammond 氏

© 2019 iSUS *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。

HPC ソフトウェアの最適化の成功事例

インテル® Parallel Studio XE

科学/研究

アプリケーション・パフォーマンスが最大 **35 倍**向上

NERSC (National Energy Research Scientific Computing Center)

ケーススタディー

人工知能

インテルにより最適化された scikit-learn は stock scikit-learn の最大 **23 倍**高速

Google Cloud Platform*

blog (英語)

ライフサイエンス

シミュレーションが最大 **7.6 倍**高速化し電力効率が **9 倍**向上**

LAMMPS コード - Sandia National Laboratories

テクノロジー概要 (英語)

その他の成功事例は、インテル® Parallel Studio XE のケーススタディーを参照

**インテル® Xeon Phi™ プロセッサー・ソフトウェア・エコシステム・モーメンタム・ガイド (英語)
パフォーマンス結果は 2016-2017 年のテスト結果に基づいたものであり、公開されている利用可能なすべてのセキュリティー・アップデートが適用されていない可能性があります。詳細については、構成の開示を参照してください。絶対的なセキュリティーを提供できる製品はありません。
性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。詳細については、http://www.intel.com/performance/ (英語) を参照してください。システム構成は、個々のケーススタディーのリンクを参照してください。
インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。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® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、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。注意事項の改訂 #20110804

© 2019 iSUS *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。

プライオリティー・サポートの利用

インテル® ソフトウェア開発ツールの有償ライセンスには購入日から 1 年間のプライオリティー・サポートが含まれており、満了時に割引価格で更新可能

利点

- **パフォーマンスと生産性を向上** — インテル® ハードウェアでコードのパフォーマンスを最大限に引き出す方法、パフォーマンス・ボトルネックや開発の課題を解消する方法などを問い合わせ可能
- インテルのエンジニアに**直接問い合わせることができ**、機密の問い合わせやコードサンプルを送信可能
- 技術的な質問やその他の製品ニーズに対する**ヘルプ**
- 製品の新しいアップデートおよび以前のバージョンへの**無料アクセス**

関連情報

- ほかのエキスパートと知識を共有できるコミュニティー製品フォーラム
- 過去数十年のハイパフォーマンス・コード作成の経験を基に構築されたドキュメント・ライブラリー

© 2019 iSUS *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。

iSUS