インテル® C++ コンパイラー 16.0 Update 1 for Linux* リリースノート (インテル® Parallel Studio XE 2016)

2015 年 12 月 7 日時点

このドキュメントは、インテル® デベロッパー・ゾーンに公開されている「Intel C++ Compiler 16.0 Update 1 for Linux* Release Notes for Intel Parallel Studio XE 2016」の日本語参考訳です。

このドキュメントでは、新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題について説明します。

詳細は、パッケージに含まれるライセンスと本リリースノートの「著作権と商標について」を参照してください。本リリースのインテル® C++ コンパイラー 16.0 Update 1 についての詳細は、次のリンクを参照してください。

変更履歴

Update 1 (インテル® C++ コンパイラー 16.0.1)

- OpenMP* 4.1 ドラフト仕様 TR3 の新機能をサポート
- インテル® SIMD Data Layout Templates (インテル® SDLT)
- 報告された問題を修正
- ドキュメントを更新

インテル® C++ コンパイラー 15.0 以降 (インテル® C++ コンパイラー 16.0.0 での変更)

- -fstack-protector-strong をサポート
- OpenMP* 4.0 の機能を追加サポート

- インテル® Data Analytics Acceleration Library (インテル® DAAL)
- C++14 の機能をサポート
- C11 の機能をサポート
- インテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーをサポート
- ランタイムにインテル® グラフィックス・テクノロジー機能の有無を問い合わせる API を追加
- ランタイムにスレッド空間構成を設定する API を追加
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けの新しい targetptr および preallocated オフロード修飾子
- 複数の同時処理を 1 つの CPU スレッドからインテル® MIC アーキテクチャーへオフロードする新しいオフロードストリームをサポート
- regparm 呼び出し規約 ABI を変更
- Linux* 分割 DWARF デバッグ情報 (DWARF Fission) をサポート
- インテル® グラフィックス・テクノロジーへのオフロードのサポートを検証する gfx_sys_check ユーティリティー
- インテル® グラフィックス・テクノロジーの共有ローカルメモリーをサポート
- #pragma simd で cilk_for ループを利用可能に
- 式の演算子の順序を決定するときに括弧を考慮可能に
- コンパイラーの組込み関数を内部的に定義
- BLOCK_LOOP および NOBLOCK_LOOP プラグマ、unroll_and_jam プラグマの private 節を追加
- 新規および変更されたコンパイラー・オプション

動作環境

- インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) 対応の IA-32 またはインテル® 64 アーキテクチャー・プロセッサーをベースとするコンピューター (インテル® Pentium® 4 プロセッサー以降、または互換性のあるインテル以外のプロセッサー)
    - 64 ビット・アプリケーションおよびインテル® MIC アーキテクチャーを対象とするアプリケーションの開発は、64 ビット・バージョンの OS でのみサポートしています。32 ビット・アプリケーションの開発は、32 ビット・バージョンまたは 64 ビット・バージョンの OS のいずれかでサポートしています。
    - 64 ビット・バージョンの OS で 32 ビット・アプリケーションを開発する場合は、Linux* ディストリビューションからオプションのライブラリー・コンポーネント (ia32-libs、lib32gcc1、lib32stdc++6、libc6-dev-i386、gcc-multilib、g++-multilib) をインストールする必要があります。
- インテル® MIC アーキテクチャー向けの開発/テスト:
    - インテル® Xeon Phi™ コプロセッサー
    - 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)
- インテル® グラフィックス・テクノロジーへのオフロードまたはネイティブサポートの開発/テスト

- オフロードは 64 ビット・アプリケーションでのみサポートされます。
- インテル® グラフィックス・テクノロジー対応 64 ビット・グラフィックス・ドライバー (インテル® ソフトウェア開発製品レジストレーション・センター (http://registrationcenter.intel.com) から入手できます)。インテル® Parallel Studio XE を登録すると Linux* 用インテル® HD グラフィックス・ドライバーのダウンロード情報にアクセスできるようになります。この情報にアクセスできない場合は、テクニカルサポートまでお問い合わせください。次のドライバーバージョンおよび対応するオペレーティング・システムをサポートしています。
  - **Linux* 用インテル® HD グラフィックス・ドライバー 16.3.2 (第 3 世代および第 4 世代インテル® Core™ プロセッサー用)**
    - SUSE LINUX Enterprise Server* 11 SP3
      - カーネル 3.14.5 (事前ビルド済 i915 カーネルモジュールを提供)
    - Ubuntu* 12.04 LTS
      - カーネル 3.2.0-41 (第 3 世代インテル® Core™ プロセッサー用、事前ビルド済 i915 カーネルモジュールを提供)
      - カーネル 3.8.0-23 (第 4 世代インテル® Core™ プロセッサー用、事前ビルド済 i915 カーネルモジュールを提供)
  - **Linux* 用インテル® HD グラフィックス・ドライバー 16.4.2 (第 4 世代および第 5 世代インテル® Core™ プロセッサー用)**
    - CentOS* 7.1 および RHEL 7.1
      - デフォルト LTS カーネル用のカーネルパッチを提供
- 次のプロセッサー・モデルをサポートしています。
  - インテル® Xeon® プロセッサー E3-1285 v3 製品ファミリーおよび E3-1285L v3 製品ファミリー (インテル® C226 チップセット) (インテル® HD グラフィックス P4700)
  - 第 4 世代インテル® Core™ プロセッサー (インテル® Iris™ Pro グラフィックス、インテル® Iris™ グラフィックス、またはインテル® HD グラフィックス 4200+ シリーズ)
  - 第 3 世代インテル® Core™ プロセッサー (インテル® HD グラフィックス 4000/2500)

    注: リストされているチップセットのインテル® Xeon® プロセッサーのみサポートしています。ほかのチップセットのインテル® Xeon® プロセッサーはサポートしていません。前世代のインテル® Core™ プロセッサーはサポートしていません。インテル® Celeron® プロセッサーおよびインテル® Atom™ プロセッサーとの互換性はありません。
- 機能を最大限に活用できるよう、マルチコアまたはマルチプロセッサー・システムの使用を推奨します。
- RAM 2GB (4GB 推奨)
- 7.5GB のディスク空き容量 (すべての機能をインストールする場合)

- 次の Linux* ディストリビューションのいずれか (本リストは、インテル社により動作確認が行われたディストリビューションのリストです。その他のディストリビューションでも動作する可能性はありますが、推奨しません。ご質問は、テクニカルサポートまでお問い合わせください。)
    - Fedora* 21、22
    - Red Hat* Enterprise Linux* 5、6、7
    - SUSE Linux Enterprise Server* 11、12
    - Ubuntu* 12.04 LTS (64 ビットのみ)、13.10、14.04 LTS、15.04
    - Debian* 7.0、8.0
    - インテル® Cluster Ready
- Linux* 開発ツール・コンポーネント (gcc、g++ および関連ツールを含む)
    - gcc バージョン 4.1-5.1 をサポート
    - binutils バージョン 2.17-2.25 をサポート
- -traceback オプションを使用するには、libunwind.so が必要です。一部の Linux* ディストリビューションでは、別途入手して、インストールする必要があります。

Eclipse* 開発環境に統合するためのその他の条件

- Eclipse* Platform 4.5 および次の両方
    - Eclipse* C/C++ Development Tools (CDT) 8.7 以降
    - Java* ランタイム環境 (JRE) 7.0 (1.7) 以降
- Eclipse* Platform 4.4 および次の両方
    - Eclipse* C/C++ Development Tools (CDT) 8.4 以降
    - Java* ランタイム環境 (JRE) 7.0 (1.7) 以降
- Eclipse* Platform 4.3 および次の両方
    - Eclipse* C/C++ Development Tools (CDT) 8.2 以降
    - Java* ランタイム環境 (JRE) 6.0 (1.6) 以降

† JRE 6.0 の Update 10 以前には、インテル® 64 アーキテクチャーでクラッシュする既知の問題があります。JRE の最新のアップデートを使用することを推奨します。詳細は、http://www.eclipse.org/eclipse/development/readme_eclipse_3.7.html (英語) のセクション 3.1.3 を参照してください。

注

- インテル® コンパイラーは、さまざまな Linux* ディストリビューションと gcc バージョンで動作確認されています。一部の Linux* ディストリビューションには、動作確認されたヘッダーファイルとは異なるバージョンのものが含まれており、問題を引き起こすことがあります。使用する glibc のバージョンは、gcc のバージョンと同じでなければなりません。最良の結果を得るため、上記のディストリビューションで提供されている gcc バージョンのみを使用してください。
- インテル® コンパイラーは、デフォルトで、インテル® SSE2 命令対応のプロセッサー (例: インテル® Pentium® 4 プロセッサー) が必要な IA-32 アーキテクチャー・アプリケーションをビルドします。コンパイラー・オプションを使

用して任意の IA-32 アーキテクチャー・プロセッサー上で動作するコードを生成できます。
- 非常に大きなソースファイル (数千行以上) を -O3、-ipo および -openmp などの高度な最適化オプションを使用してコンパイルする場合は、多量の RAM が必要になります。
- 上記のリストにはすべてのプロセッサー・モデル名は含まれていません。リストされているプロセッサーと同じ命令セットを正しくサポートしているプロセッサー・モデルでも動作します。特定のプロセッサー・モデルについては、テクニカルサポートにお問い合わせください。
- 一部の最適化オプションには、アプリケーションを実行するプロセッサーの種類に関する制限があります。詳細は、オプションの説明を参照してください。

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)

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) は、インテル® C++ コンパイラーのインストール前またはインストール後にインストールできます。

最新バージョンのインテル® MPSS を使用することを推奨します。インテル® Parallel Studio XE for Linux* を登録すると、インテル® ソフトウェア開発製品レジストレーション・センター (http://registrationcenter.intel.com) から入手できます。

ユーザー空間およびカーネルドライバーのインストールに必要な手順については、インテル® MPSS のドキュメントを参照してください。

インテル® C++ コンパイラーの使用方法

コマンドラインおよび Linux* からのインテル® C++ コンパイラーの使用方法についての情報は、『入門ガイド』 (*<install-dir>/documentation_2016/ps2016/getstart_comp_lc.htm*) に含まれています。

インテル® C++ コンパイラー for Linux* は環境モジュール・ソフトウェア・ユーティリティーとともに使用できますが、「モジュールファイル」は含まれていません。詳細は、「インテル® 開発環境での環境モジュールの使用」 (英語) を参照してください。

ドキュメント

製品ドキュメントは、*<install-dir>/documentation_2016/ja/ps2016/getstart_comp_lc.htm* からリンクされています。すべてのツール・コンポーネントのドキュメントは、インテル® Parallel Studio XE サポートページから入手できます。

インテルが提供するデバッグ・ソリューション

- インテルが提供するデバッグ・ソリューションは GNU* GDB ベースです。詳細は、「インテル® Parallel Studio 2016 Composer Edition C++ - デバッグ・ソリューション リリースノート」 (英語) を参照してください。

サンプル

製品サンプルは、*<install-dir>/samples_2016/ja/compiler_c/psxe* ディレクトリーにあります。

テクニカルサポート

インストール時に製品の登録を行わなかった場合は、インテル® ソフトウェア開発製品レジストレーション・センター (http://registrationcenter.intel.com) で登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

テクニカルサポート、製品のアップデート、ユーザーフォーラム、FAQ、ヒント、およびその他のサポート情報は、http://www.intel.com/software/products/support/ (英語) を参照してください。

注: 販売代理店がこの製品のテクニカルサポートを提供している場合、インテルではなく販売代理店にお問い合わせください。

新機能と変更された機能

このバージョンでは、次の機能が新たに追加または大幅に拡張されています。これらの機能に関する詳細は、ドキュメントを参照してください。

OpenMP* 4.1 ドラフト仕様 TR3 の新機能をサポート

インテル® コンパイラー 16.0 における OpenMP* 4.1 ドラフト仕様 TR3 の新しい機能のサポートは、OpenMP* 4.5 仕様 (2015 年 11 月にリリース予定) の規格に合わせて変更される可能性があります。

- *#pragma omp simd simdlen(n)* をサポート
- *#pragma omp ordered simd* をサポート
- *processor* 節の拡張を *#pragma omp declare simd* に追加
- *#pragma omp declare simd* ディレクティブの *linear* 節を新しい修飾子で拡張
    - *linear (linear-list [: linear-step])* (*linear-list* は次のいずれか)
        - *list*
        - *modifier (list)* (*modifier* は *ref*、*val*、*uval* のいずれか)
    - すべての *list* 項目は各 SIMD レーンで呼び出される関数の引数でなければなりません。

    - *modifier* が指定されない場合や *val* または *uval modifier* が指定された場合、各レーンの各 *list* 項目の値は、関数に入るときの *list* 項目の値とレーンの論理番号の倍数 *linear-step* に相当します。
    - *uval modifier* が指定された場合、各呼び出しは各 SIMD レーンと同じメモリー位置を使用します。このメモリー位置は論理的な最終レーンの最後の値で更新されます。
    - *ref modifier* が指定された場合、各レーンの各 *list* 項目のメモリー位置は、レーンの論理番号の倍数 *linear-step* でインデックスされた関数に入るときのメモリー位置の配列に相当します。

インテル® SIMD Data Layout Templates (インテル® SDLT)

- インテル® SDLT は、SIMD ベクトル化の熟練者に手を借りることなく、SIMD ハードウェアとコンパイラーを最大限に活用できるよう支援するライブラリーです。
- インテル® SDLT は、ISO C++11、インテル® Cilk™ Plus SIMD 拡張、および *#pragma ivdep* をサポートしているすべてのコンパイラーで使用できます。
- インテル® SIMD Data Layout Templates の特長
    - ベクトル化のデータレイアウトが SOA の場合でも AOS 形式でプログラム可能
    - パフォーマンス効率の良い SIMD ベクトル化をターゲット
    - ほかの明示的な SIMD プログラミング・モデルと互換
- インテル® SDLT で効率的な SIMD コードを生成するには、次のコンパイラー・オプションが必要です (または推奨します)。
    - C++11 のサポート
        - */Qstd:c11* (Windows*) または *-std=c11* (Linux*)
    - レベル 2 以上のコード最適化
        - */O2* および */O3* (Windows*) または *-O2* および *-O3* (Linux*)
    - 最適化における ANSI エイリアシング規則
        - /Qansi-alias (Windows*) または -ansi-alias (Linux*)
    - ターゲットにするプロセッサー機能をコンパイラーにオプションで指示します。
        - /Qx*code* (Windows*) または -x*code* (Linux*)
    - パフォーマンス上の利点がある場合、複数の機能固有の自動ディスパッチ・コードを生成するようにコンパイラーにオプションで指示します。
- C++11 依存関係により、インテル® SDLT を Linux* で使用する場合は GCC 4.7 以降が必要です。インテル® SDLT を STL アルゴリズムで使用する場合は GCC 4.8 以降を推奨します。
- C++11 依存関係により、インテル® SDLT を Windows* で使用する場合は Microsoft* Visual Studio* 2012 以降が必要です。

OpenMP* 4.0 の機能を追加サポート

- ユーザー定義リダクションを定義する *#pragma omp declare reduction* をサポート
- *#pragma omp simd collapse(n)* をサポート

インテル® Data Analytics Acceleration Library (インテル® DAAL)

- インテル® DAAL は、使いやすいライブラリーによりビッグデータの解析と機械学習のパフォーマンスを大幅に向上します。

C++14 の機能をサポート

インテル® C++ コンパイラー 16.0 は、*/Qstd:c++14* (Windows*) または *-std=c++14* (Linux*/OS X*) コンパイラー・オプションで C++14 の機能をサポートします。

- 以前の主要バージョンのコンパイラーとの比較を含む、サポートしている機能の最新リストは、「インテル® C++ コンパイラーでサポートされる C++14 の機能」を参照してください。

C11 の機能をサポート

インテル® C++ コンパイラー 16.0 は、*/Qstd:c11* (Windows*) または *-std=c11* (Linux*/OS X*) コンパイラー・オプションで C11 の機能をサポートします。

- 以前の主要バージョンのコンパイラーとの比較を含む、サポートしている機能の最新リストは、「インテル® C++ コンパイラーにおける C11 サポート」を参照してください。

***-fstack-protector-strong* をサポート**

インテル® C++ コンパイラーで gcc* 4.9 オプション *-fstack-protector-strong* をサポートしました。

インテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーをサポート

インテル® C++ コンパイラーでインテル® TBB のスレッドレイヤーをサポートするインテル® MKL ライブラリーのサポートを追加しました。これらのライブラリーは、/Qmkl および /Qtbb (Windows*) または -mkl および -tbb (Linux* / OS X*) の両方のオプションを指定し、/Qopenmp 、-qopenmp、-fopenmp オプションを指定しない場合に使用されます。OpenMP* が必要なときにインテル® TBB を使用するインテル® MKL ライブラリーをリンクする場合は、リンク時にライブラリーを明示的に指定する必要があります。

ランタイムにインテル® グラフィックス・テクノロジー機能の有無を問い合わせる API を追加

ソースレベルでランタイムにインテル® グラフィックス・テクノロジーに関するハードウェア情報を取得する API *_GFX_get_device_platform(void)*、*_GFX_get_device_sku(void)*、*_GFX_get_device_hardware_thread_count(void)*、*_GFX_get_device_min_frequency(void)*、*_GFX_get_device_mas_frequency(void)*、および *_GFX_get_device_current_frequency(void)* が追加されました。

ランタイムにスレッド空間構成を設定する API を追加

アプリケーションでランタイムにスレッド空間とスレッドグループの高さと幅を制御できる API *_GFX_set_thread_space_config(int, int, int, int)* が追加されました。アプリケーションでランタイムにスレッド空間の高さと幅を制御できる GFX_THREAD_SPACE_WIDTH および GFX_THREAD_SPACE_HEIGHT 環境変数が追加されました。スレッドグループの高さと幅を制御できる GFX_THREAD_GROUP_WIDTH および GFX_THREAD_GROUP_HEIGHT 環境変数もあります。

インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けの新しい targetptr および preallocated オフロード修飾子

オフロード構文に 2 つの新しい修飾子 *targetptr* および *preallocated* が追加されました。これらの修飾子を使用すると、*#pragma offload (targetptr)* から、またはオフロードコード (*preallocated targetptr*) により、インテル® MIC アーキテクチャー専用のメモリーを割り当てることができます。

複数の同時処理を 1 つの CPU スレッドからインテル® MIC アーキテクチャーへオフロードする新しいオフロードストリームをサポート

ターゲットデバイスで並列に実行される、インテル® MIC アーキテクチャーへの複数処理のオフロードを実装する、新しい API (*_Offload_stream_handle*、*_Offload_stream_completed*、*_Offload_device_streams_completed*) および追加のオフロードプラグマの節 (*stream*、*signal*) が利用できるようになりました。オフロードストリーム API の詳細は、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』を参照してください。

オフロードストリームを使用するには、次の環境変数を設定する必要があります。

- MIC_ENV_PREFIX=MIC
- MIC_KMP_AFFINITY=norespect,none
- OFFLOAD_STREAM_AFFINITY={compact | scatter}

regparm 呼び出し規約 ABI を変更

regparm 呼び出し規約を gcc 4.0 で実装されている ABI に合わせて変更しました。この変更により、インテル® C++ コンパイラーの以前のバージョンとの互換性はなくなります。以前の動作に戻すには、-mregparm-version=1 オプションを指定してください。

整数ベクトル型で基本演算と論理演算子をサポート

vector_size 属性を使用する int データ型の変数で、gcc ドキュメント (http://gcc.gnu.org/onlinedocs/gcc/Vector-Extensions.html (英語)) に記述されている基本演算を利用できるようになりました。

Linux* 分割 DWARF デバッグ情報 (DWARF Fission) をサポート

インテル® C++ コンパイラー 16.0 では、主要デバッグ情報を含む追加のオブジェクト・ファイル (拡張子 *.dwo*) を生成する *-gsplit-dwarf* コンパイラー・オプションが追加されました。現在は、32 ビット・ターゲットと 64 ビット・ターゲットのみサポートしています。このオプションを指定してコンパイルするには binutils 2.23 以降が必要です。また、デバッグするには gdb 7.5 以降が必要です。

インテル® グラフィックス・テクノロジーへのオフロードのサポートを検証する gfx_sys_check ユーティリティー

プラットフォームがインテル® グラフィックス・テクノロジーへのオフロードをサポートしているか確認するユーティリティー *gfx_sys_check* が提供されました。このユーティリティーは、インテル® グラフィックス・テクノロジーに関連する詳細も提供します。

インテル® グラフィックス・テクノロジーの共有ローカルメモリーをサポート

インテル® グラフィックス・テクノロジーのスレッド間で共有ローカルメモリーを有効にして、RAM トラフィックを減らしパフォーマンスを向上する、新しい *cilk_for* ループ *_Thread_group* 節、新しい *__thread_group_local* 型とストレージ修飾子、新しい API *_gfx_gpgpu_thread_barrier()* が提供されました。

#pragma simd で cilk_for ループを利用可能に

インテル® Cilk™ Plus の *cilk_for* キーワードを使用して並列化された C/C++ for ループを、*#pragma simd* または *_Simd* キーワードを使用して、明示的なベクトル化のターゲットにできるようになりました。

式の演算子の順序を決定するときに括弧を考慮可能に

/Qprotect-parens (Windows*) および *-fprotect-parens* (Linux*/OS X*) オプションを指定すると、コンパイラーは式を評価するときに括弧を考慮し、演算の順序を変更しません。例えば、これらのオプションを使用すると、コンパイラーは次の変換を行いません。

double y = (a + b) + c;
から
double y = a + (b + c);

これらのオプションは、デフォルトでは無効です。これらのオプションを使用すると、パフォーマンスに悪影響を及ぼすことがあります。

コンパイラーの組込み関数を内部的に定義

コンパイル時間を短縮するため、インテル® C++ コンパイラー 16.0 では組込み関数宣言のヘッダーファイルをチェックしないようになりました。この変更が型チェックに影響する可能性があるため、関数プロトタイプを追加する、

-D__INTEL_COMPILER_USE_INTRINSIC_PROTOTYPES コンパイラー・オプションが追加されました。

BLOCK_LOOP および NOBLOCK_LOOP プラグマ、unroll_and_jam プラグマの private 節を追加

ループ・ブロッキング情報をコンパイラーに伝える新しいプラグマと更新されたプラグマが追加されました。これには、新しいプラグマ *#pragma BLOCK_LOOP [clause[[,] clause...]]* および *#pragma NOBLOCK_LOOP* も含まれます。さらに、*#pragma unroll_and_jam* への *private(var list)* 節の追加も含まれます。詳細は、『インテル® C++ コンパイラー・ユーザー・リファレンス・ガイド』を参照してください。

新規および変更されたコンパイラー・オプション

コンパイラー・オプションの詳細は、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』の「コンパイラー・オプション」セクションを参照してください。

- -daal[= lib] インテル® Data Analytics Acceleration Library (インテル® DAAL) の特定のライブラリーにリンクするようにコンパイラーに指示します。
- -mgpu-asm-dump[=filename] オフロードするプロセッサー・グラフィックス用コードのネイティブ・アセンブリー・リストを生成するようにコンパイラーに指示します。このオプションは、インテル® グラフィックス・テクノロジーにのみ適用されます。
- -mregparm-version=n regparm 引数の引き渡しに使用するアプリケーション・バイナリー・インターフェイス (ABI) のバージョンを指定します。
- -print-sysroot コンパイル時に使用するターゲットの sysroot ディレクトリーを出力します。
- -qoffload-arch=arch [: visa] コードのオフロードに使用するターゲット・アーキテクチャーを指定します。このオプションは、インテル® MIC アーキテクチャーおよびインテル® グラフィックス・テクノロジーにのみ適用されます。インテル® グラフィックス・テクノロジーの場合、仮想 ISA (vISA) を指定することもできます。
- -qoffload-svm 共有仮想メモリー (SVM) モードを使用するかどうかを指定します。このオプションは、インテル® グラフィックス・テクノロジーにのみ適用されます。
- -qopt-prefetch-issue-excl-hint インテル® マイクロアーキテクチャー Broadwell (開発コード名) 以降で prefetchW 命令をサポートします。

廃止予定のコンパイラー・オプションのリストは、『インテル® C++ コンパイラー 16.0 ユーザーズガイド』の「コンパイラー・オプション」セクションを参照してください。

-o で始まるコンパイラー・オプションは廃止予定

-o で始まるコンパイラー・オプションは廃止予定です。これらのオプションは、-q で始まる新しいオプションに変更されます。例えば、-opt-report は -qopt-report

に変更されます。この変更は、-o<text> オプションを出力ファイル名とするサードパーティーのツールとの互換性を向上するために行われました。

終了予定のサポート

Red Hat* Enterprise Linux 5* のサポートを終了予定

Red Hat* Enterprise Linux* 5 のサポートは、将来のリリースで終了する予定です。

32 ビット・ホストへのインストールのサポートを終了予定

32 ビット・ホストへのインストールは、将来のリリースで終了する予定です。ただし、32 ビット・ターゲット用コード生成のサポートは引き続き行われます。

Linux* 用インテル® HD グラフィックス・ドライバー 16.3.2 (第 3 世代および第 4 世代インテル® Core™ プロセッサー用) のサポートを終了予定

16.3.2 ドライバーのサポートは、将来のリリースで終了する予定です。第 4 世代インテル® Core™ プロセッサー・ベースのシステムは、16.4.2 ドライバーおよびこのドライバーをサポートしているオペレーティング・システム (CentOS* 7.1 または RHEL 7.1) に移行することを推奨します。

Linux* 用インテル® HD グラフィックス・ドライバー 16.3.2 をサポートするオペレーティング・システムのサポートを終了予定

SLES 11 SP3 および Ubuntu* 12.04 LTS のサポートは、将来のリリースで終了する予定です。

終了したサポート

Debian* 6 のサポートを終了

これらのオペレーティング・システム・バージョンのサポートを終了しました。新しいバージョンのオペレーティング・システムに移行してください。

Fedora* 20 のサポートを終了

これらのオペレーティング・システム・バージョンのサポートを終了しました。新しいバージョンのオペレーティング・システムに移行してください。

Eclipse* 3.8 のサポートを終了

これらの IDE バージョンのサポートを終了しました。新しいバージョンの IDE に移行してください。

スタティック解析のサポートを終了

スタティック解析のサポートを終了しました。ご意見やお問い合わせは、こちら (英語)までお寄せください。

Mudflap ライブラリーのサポートを終了

-fmudflap および Mudflap ライブラリーのサポートを終了しました。

既知の制限事項

ポインターチェッカーにダイナミック・ランタイム・ライブラリーが必要

-check-pointers オプションを使用する場合は、ランタイム・ライブラリー *libchkp.so* をリンクする必要があります。*-static* または *-static-intel* のようなオプションを *-check-pointers* とともに使用すると、設定に関係なくこのダイナミック・ライブラリーがリンクされることに注意してください。詳細は、http://intel.ly/1jV0eWD (英語) を参照してください。

インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーの既知の問題

- オフロードコードを含む共有ライブラリーをロードする前に *MIC_LD_LIBRARY_PATH* を設定

  dlopen を使用してプログラム内で共有オブジェクトをロードするとき、.so にオフロードが含まれる場合は *.so が読み込まれるように *MIC_LD_LIBRARY_PATH* 変数を設定する必要があります。これは、.so を相対パスまたはフルパスで指定する場合にも必要です (例: *dlopen("../../libmylib.so", <flag>)*)。

- 共有ライブラリーに含まれるコードをオフロードする際に *-qoffload=mandatory* オプションまたは *-qoffload=optional* オプションを指定してメインプログラムのリンクが必要

  オフロードには初期化処理が必要ですが、これはメインプログラムでのみ行うことができます。つまり、共有ライブラリーに含まれるコードをオフロードする場合、初期化処理が行われるように、メインプログラムもリンクしなければなりません。メインコードやメインプログラムへスタティック・リンクされたコードにオフロード構造が含まれる場合、これは自動で行われます。そうでない場合、*-qoffload=mandatory* コンパイラー・オプションまたは *-qoffload=optional* コンパイラー・オプションを指定して、メインプログラムをリンクする必要があります。

- オフロード・コンパイル・モデルでリンク時に見つからないシンボルの検出

リンク時に見つからないシンボルを検出するために *-offload-option,mic,compiler,"-z defs"* を指定する必要はなくなりました。コンパイラー・ドライバーが、この検出を自動的に行います。

- コンパイル時の診断の **MIC** タグ

  ターゲット (インテル® MIC アーキテクチャー) とホスト CPU のコンパイルを区別できるようにコンパイラーの診断インフラストラクチャーが変更され、出力メッセージに **MIC** タグが追加されました。このタグは、インテル® MIC アーキテクチャー用のオフロード拡張を使用してコンパイルしたときに、ターゲットのコンパイル診断にのみ追加されます。

  下記の例で、サンプル・ソース・プログラムは、ホスト CPU とターゲット (インテル® MIC アーキテクチャー) のコンパイルの両方で同じ診断を行っています。ただし、プログラムによっては、2 つのコンパイルで異なる診断メッセージが出力されます。新しいタグが追加されたことで、CPU とターゲットのコンパイルを容易に区別できることが分かります。

  $ icc -c sample.c

  *sample.c(1): 警告 #1079: *MIC* 関数 "main" の戻り型は "int" でなければなりません。*
  void main()
  ^
  *sample.c(5): 警告 #120: *MIC* 戻り値の型が関数の型と一致しません。*
  return 0;
  ^
  sample.c(1): 警告 #1079: 関数 "main" の戻り型は "int" でなければなりません。
  void main()
  ^
  sample.c(5): 警告 #120: 戻り値の型が関数の型と一致しません。
  return 0;

- ランタイム型情報 (RTTI) は未サポート

  仮想共有メモリー・プログラミングでは、ランタイム型情報 (RTTI) はサポートされていません。特に、*dynamic_cast<>* と *typeid()* の使用はサポートされていません。

- 直接 (ネイティブ) モードにおけるランタイム・ライブラリーのコプロセッサーへの転送

  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) に、*/lib* 以下のインテル® コンパイラーのランタイム・ライブラリー (例えば、OpenMP* ライブラリー *libiomp5.so*) が含まれなくなりました。

このため、直接モード (例えば、コプロセッサー・カード上) で OpenMP* アプリケーションを実行する場合は、アプリケーションを実行する前にインテル® MIC アーキテクチャー OpenMP* ライブラリー (*<install_dir>/compilers_and_libraries_2016/linux/lib/mic/libiomp5.so*) のコピーをカード (デバイス名の形式は micN。最初のカードは *mic0*、2 番目のカードは *mic1*、...) に (*scp* 経由で) アップロードする必要があります。

このライブラリーが利用できない場合、次のようなランタイムエラーが発生します。

/libexec/ld-elf.so.1: "sample" で要求された共有オブジェクト "libiomp5.so" が見つかりません。

libimf.so のような別のコンパイラー・ランタイムでも同様です。必要なライブラリーは、アプリケーションおよびビルド構成により異なります。

- オフロード領域からの *exit()* の呼び出し

  オフロード領域から *exit()* を呼び出すと、"オフロードエラー: デバイス 0 のプロセスがコード 0 で予想外に終了しました" のような診断メッセージが出力され、アプリケーションが終了します。

インテル® グラフィックス・テクノロジーへのオフロードの既知の問題

- オフロードコードのホストバージョンが並列化されない

  コンパイラーは、#pragma offload 以下に並列ループのターゲットバージョンとホストバージョンの両方を生成します。ホストバージョンは、オフロードが実行できない場合 (通常は、ターゲットシステムにインテル® グラフィックス・テクノロジーが有効なユニットがない場合) に実行されます。並列ループは、オフロードの並列セマンティクスを含む cilk_for の並列構文または配列表記文を使用して指定する必要があります。ターゲットバージョンはターゲット実行の際に並列化されますが、現在、ホスト側のバックアップ・バージョンが並列化されない制限があります。cilk_for を使用したときにオフロード実行が行われないと、バックアップ・コード実行のパフォーマンスに大きく影響する場合があることに注意してください。配列表記文は現在ホスト側で並列コードを生成しないため、パフォーマンスに影響はありません。これは既知の問題で、将来のリリースで修正される予定です。

- 非 root 権限で複数のプロセスを実行するとオフロードに失敗することがある

  (非 root 権限で) 複数のプロセスをオフロードしようとすると失敗することがあります。/dev/dri/card0 を開く最初のプロセスのみ DRM 認証をパスすることができ、master 権限があります。DRM 認証をパスするには、"root" または "master" 権限が必要です。このため、root 権限で実行するとすべてのプロセスがパスしますが、非 root 権限では 1 つのプロセスしかパスしません。これは Linux* の既知の制限です。

回避策を次に示します。

  - 各プロセスをシリアルに実行します。
  - root として実行します。

- インテル® グラフィックス・テクノロジーへのオフロードの既知の制限事項
  - オフロードコードでは、次の機能を使用できません。
    - 例外処理
    - RTTI
    - *longjmp/setjmp*
    - VLA
    - 変数引数リスト
    - 仮想関数、関数ポインター、その他の間接呼び出しまたはジャンプ
    - 共有仮想メモリー
    - 配列や構造体のようなポインターを含むデータ構造
    - ポインターまたは参照型のグローバル変数
    - OpenMP*
    - *cilk_spawn* または *cilk_sync*
    - インテル® Cilk™ Plus のレデューサー
    - ANSI C ランタイム・ライブラリー呼び出し (SVML、*math.h*、*mathimf.h* 呼び出し、およびその他いくつかの例外あり)
  - 64 ビット浮動小数点演算および整数演算は非効率

インテル® Cilk™ Plus の既知の問題

- ランタイムのスタティック・リンクはサポートされていません。

  インテル® Cilk™ Plus ライブラリーのスタティック・バージョンは、意図的に提供されていません。スタティック・ライブラリーをリンクする *-static-intel* を使用すると、警告が表示され、インテル® Cilk™ Plus ライブラリーのダイナミック・バージョン *libcilkrts.so* がリンクされます。

  $ icc -static-intel sample.c

  icc: 警告 #10237: -lcilkrts はダイナミックにリンクされました。スタティック・ライブラリーは利用できません。

  代わりに、インテル® Cilk™ Plus のオープンソース・バージョンとスタティック・ランタイムをビルドできます。インテル® Cilk™ Plus の実装についての詳細は、http://www.isus.jp/article/intel-cilk-plus/ を参照してください。
  インテル® Cilk™ Plus ライブラリーのダイナミック・バージョンを使用したときに問題が発生した場合は、テクニカルサポートまでご連絡ください。

ガイド付き自動並列化の既知の問題

- プログラム全体のプロシージャー間の最適化 (*-ipo*) が有効な場合、単一ファイル、関数名、ソースコードの指定範囲に対してガイド付き自動並列化 (GAP) 解析は行われません。

インテル® IA-32 C++ コンパイラーを Red Hat* Enterprise Linux* 6 で使用すると SPEC CPUv6 ランタイム・バス・エラーが発生する

- SPEC CPUv6 ベンチマーク (現在開発中) をインテル® IA-32 C++ コンパイラーでコンパイルした後、*ulimit* コマンド (例えば、*ulimit -s 2067152 -v 15000000*) を使用してスタック/仮想メモリーを制限して実行すると、バスエラーが発生します。現在、この問題はこのベンチマークでのみ発生していますが、ほかのアプリケーションでも発生する可能性があります。このエラーを回避するには、これらのパラメーターを *unlimited* に設定してください。

著作権と商標について

最適化に関する注意事項
インテル® コンパイラーでは、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

インテル製品は、予告なく仕様や説明が変更される場合があります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本資料で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本資料で紹介されている資料番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、インテルの Web サイト (http://www.intel.com/design/literature.htm (英語)) を参照してください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ を参照してください。

インテル® C++ コンパイラーは、インテルのソフトウェア使用許諾契約書 (EULA) の下で提供されます。

詳細は、製品に含まれるライセンスを確認してください。

Intel、インテル、Intel ロゴ、Celeron、Cilk、Intel Atom、Intel Core、Intel Xeon Phi、Iris、Pentium、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。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。