インテル® C++ コンパイラー 17.0 Update 4 for Linux* リリースノート (インテル® Parallel Studio XE 2017)

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。

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」をお読みください。
本リリースのインテル® C++ コンパイラー 17.0 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 4 (インテル® C++ コンパイラー 17.0.4)

- 日本語版を更新
- 報告された問題を修正

Update 3 (インテル® C++ コンパイラー 17.0.3)

- 報告された問題を修正

Update 2 (インテル® C++ コンパイラー 17.0.2)

- 報告された問題を修正

Update 1 (インテル® C++ コンパイラー 17.0.1)

- インテル® コンパイラーのドキュメントおよび診断メッセージの日本語訳を追加
- 報告された問題を修正

インテル® C++ コンパイラー 16.0 以降 (インテル® C++ コンパイラー 17.0 での変更)

- 第 2 世代インテル® Xeon Phi™ プロセッサー・ファミリー向けに新しい CPU 名 "*mic_avx512*" を追加
- インテル® C++ コンパイラーのヘッダーファイルのリストをサブフォルダーに移動 (17.0 RTM)
- SIMD Data Layout Templates (SDLT) に N 次元配列のサポートを追加 (17.0 RTM)
- Eclipse* 4.6 および CDT 9.0 のサポートを追加 (17.0 RTM)
- OpenMP* 4.0 以降の新機能をサポート
- 新しいインテル® Xeon Phi™ プロセッサー/コプロセッサーへのオフロード機能
- アノテーション付きソースリスト
- コード・アライメント用の新しい属性、プラグマ、コンパイラー・オプション
- C++14 の機能をサポート
- C11 の機能をサポート
- 新規および変更されたコンパイラー・オプション
- オフロード DEFAULTMAP のデフォルト動作の変更
- OpenMP* ヘルパースレッドの削除

先頭へ戻る

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) 対応のインテル® 64 アーキテクチャー・ベースのプロセッサーを搭載したコンピューター (第 2 世代以降のインテル® Core™ i3/i5/i7 プロセッサー、インテル® Xeon® プロセッサー E3/E5 ファミリー、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® メニー・インテグレーテッド・コア (インテル® MIC) 向けの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も、64 ビット・バージョンの OS でのみサポートしています。コンパイラーは、32 ビットの OS にインストールできません。
    - 64 ビット・バージョンの OS で 32 ビット向けの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、対応するオペレーティング・システム、およびプロセッサー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 16.4.4 (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 (64 ビット・アーキテクチャー)
    - Red Hat* Enterprise Linux* (RHEL) 7.1 (64 ビット・アーキテクチャー)

    - 次のプロセッサーをサポートしています。
        - 第 5 世代インテル® Core™ プロセッサー (インテル® Iris™ グラフィックス、インテル® HD グラフィックス 5500/6000/6100)
        - 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) (チップセットの互換性は通常、インテル® Core™ プロセッサーでは問題になりません)
        - インテル® Xeon® プロセッサー E3 v3 ファミリー (インテル® HD グラフィックス P4700)
        - インテル® Xeon® プロセッサー E3 v4 ファミリー (インテル® Iris™ Pro グラフィックス P6300)
        - **注意事項:**
        - チップセットでプロセッサー・グラフィックスを有効にする必要があります。データシートを必ず確認してください。
        - インテル® Xeon® プロセッサーにはインテル® C226 チップセットが必要です。
        - 第 4 世代より前のインテル® 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、インテル® Pentium® プロセッサーおよびインテル® Atom™ プロセッサーはサポートしていません。
- 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)
- 7.50GB 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* 24、25
    - Red Hat* Enterprise Linux* 6、7
    - SUSE* Linux* Enterprise Server 11、12
    - Ubuntu* 14.04 LTS、15.10、16.04 LTS
    - 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 4.3-6 をサポート
    - binutils バージョン 2.20-2.26 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 4.6 および次の両方
    - Eclipse* C/C++ Development Tools (CDT) 8.8 または 9.0
    - Java* ランタイム環境 (JRE) 8.0 (1.8) 以降
- Eclipse* Platform 4.5 および次の両方
    - Eclipse* C/C++ Development Tools (CDT) 8.7
    - 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) 8.4-8.6
    - Java* ランタイム環境 (JRE) 7.0 (1.7) 以降

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。

- 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。

先頭へ戻る

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、「インテル® C++ コンパイラー 17.0 for Linux* 入門」 (*<install-dir>*/documentation_2017/ja/compiler_c/ps2017/get_started_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® 開発ツールでの環境モジュールの使用」 (英語) を参照してください。

先頭へ戻る

ドキュメント

製品ドキュメントへのリンクは、*<install-dir>*/documentation_2017/ja/compiler_c/ps2017/get_started_lc.htm にあります。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」 (英語) から入手できます。

日本語のサポート

日本語対応のインテル® コンパイラーをインストールした場合、オプションで日本語のサポートが提供されます。エラーメッセージ、仮想開発環境のダイアログ、一部のドキュメントが (英語に加えて) 日本語で提供されます。デフォルトでは、エラーメッセージとダイアログの言語はオペレーティング・システムの言語で表示されます。日本語ドキュメントは、ドキュメントの **ja** サブディレクトリーに含まれています。

日本語のサポートは、すべてのアップデートではなく、一部のアップデートで提供されます。

日本語オペレーティング・システムで英語のサポートを使用する (または英語オペレーティング・システムで日本語のサポートを使用する) 方法については、こちらの記事 (英語) を参照してください。

先頭へ戻る

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 XE 2017 Composer Edition for 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。

先頭へ戻る

サンプル

製品のサンプルは、「インテル® ソフトウェア製品のサンプルとチュートリアル」 (英語) からダウンロードできます。

先頭へ戻る

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。

先頭へ戻る

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。

第 2 世代インテル® Xeon Phi™ プロセッサー・ファミリー向けに新しい CPU 名 "*mic_avx512*" を追加

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) 基本命令、競合検出命令、指数および逆数命令、プリフェッチ命令、および RDSEED および ADX (Multi-Precision Add-Carry Instruction Extensions) 命令を含むインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) 対応第 2 世代インテル® Xeon Phi™ プロセッサー・ファミリー向けに、新しい CPU 名 "*mic_avx512*" を追加しました。

新しい CPU 名の使用方法は、『インテル® C++ コンパイラー 17.0 デベロッパー・ガイドおよびリファレンス』を参照してください。

インテル® C++ コンパイラーのヘッダーファイルのリストをサブフォルダーに移動

コンパイラーのヘッダーファイルのリストを既存の include フォルダーのサブフォルダーに移動しました。インテル® C++ コンパイラーのヘッダーを使用するソースコードを変更する必要はありません。新しいサブフォルダーは、コンパイラー・ドライバーによりコンパイル中に自動的に検索されます。

SIMD Data Layout Templates (SDLT) に SIMD プログラムの集約 (Gather)/分散 (Scatter) を減らす N 次元配列のサポートを追加

- C++ AOS レイアウトでは、SIMD ループ/関数をベクトル化するときの集約 (Gather)/分散 (Scatter) の生成を最小限に抑えるように AOS -> SOA 変換の注釈を付けるプログラマー用の言語拡張は存在しません。
- SDLT プリミティブ・テンプレート V2 は、n-D コンテナーをサポートしています。このコンテナーは、n-D AOS レイアウトを n-D SOA レイアウトに変換するプリミティブ/メソッドのセットをサポートする C++11 機能を使用して設計および実装されています。

OpenMP* 4.0 以降の新機能をサポート

- `#pragma omp for linear (list [ : linear-step ])` をサポート
  - `list` は `list` または `modifier(list)` のいずれか
- `linear` 節の `ref`、`val`、`uval modifier` をサポート
  - 例: `linear(ref(p)), linear(val(i):1), linear(uval(j):1)`
- `#pragma omp simd simdlen(n)` をサポート
- `#pragma omp ordered [simd]` をサポート
- 配列全体のリダクション: `int x[n]; #pragma omp simd reduction(+:x)`
- *processor* 節の拡張を `#pragma omp declare simd` に追加 (OpenMP* 4.5 の正式な機能ではありません)
- `#pragma omp for schedule` 節の `SIMD` 修飾子と `NONMONOTONIC` 修飾子をサポート
  - インテル® C++ コンパイラー 17.0 には、ループの反復をチームのスレッド間でどのように分割するかについて、ユーザー制御を強化する `SIMD` 修飾子と `NONMONOTONIC` 修飾子の拡張が含まれています。詳細は、『インテル® C++ コンパイラー・デベロッパー・ガイドおよびリファレンス』を参照してください。
- reduction 節で部分配列をリスト項目としてサポート
  - `reduction(reduction-identifier:list)`: リスト項目が部分配列の場合、reduction 節がセクションの個々の要素に適用されると見なして処理されます。プライベートの部分配列の要素は、連続して割り当てられます。

新しいインテル® Xeon Phi™ プロセッサー/コプロセッサーへのオフロード機能

- OpenMP* 4.5 節の変更
  - 結合構造または複合構造の場合、if 節でディレクティブ名修飾子をサポート
    - `if([directive-name-modifier :] scalar-expression)` 構造が directive-name-modifier で指定された場合、if 節はその構造のセマンティクスにのみ適用されます。その他の場合、if 節を適用できるすべての構造に適用されます。
      例: `#pragma omp target parallel for if(target : do_offload_compute)`
  - `use_device_ptr(list)` 節を `#pragma omp target data` に実装
  - `is_device_ptr(list)` 節を `#pragma omp target` に実装
- 結合 `target` 構造のサポート
  - `#pragma omp target parallel`

    - `#pragma omp target parallel for`
    - `#pragma omp target simd`
    - `#pragma omp target parallel for simd`
- 新しいデバイスメモリー API のサポート
    - `void* omp_target_alloc()`
    - `void omp_target_free()`
    - `int omp_target_is_present()`

アノテーション付きソースリスト

- この機能は、コンパイラーによる最適化レポートをソースファイルに追加します。リスト形式はテキストまたは html のいずれかで指定します。リストを表示する場所は、呼び出し元、呼び出し先、または両方で指定できます。

ループのコード・アライメント用の新しい属性、プラグマ、コンパイラー・オプション

- 新しい属性 `__attribute__((code_align(n)))` は、関数を 2 の累乗のバイト境界 *n* でアライメントします。
- 新しいプラグマ `#pragma code_align [(n)]` は、後続のループの先頭を 2 の累乗のバイト境界 *n* でアライメントします。
- 新しいコンパイラー・オプション `-falign-loops[=n]` は、すべてのループを 2 の累乗のバイト境界 *n* でアライメントします。`-fno-align-loops` (デフォルト) は、特別なループのアライメントを行いません。

C++14 の機能をサポート

インテル® C++ コンパイラー 17.0 は、*/Qstd:c++14* (Windows®) または *-std=c++14* (Linux*/OS X*) コンパイラー・オプションで以下の C++14 の機能をサポートします。

- 可変テンプレート (N3651)
- constexpr の緩和 (拡張) (N3652)
- サイズ指定された解放 (N3663)
- 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 17.0 は、*/Qstd:c11* (Windows®) または *-std=c11* (Linux*/OS X*) コンパイラー・オプションで以下の C11 の機能をサポートします。

- `_Atomic` および `__attribute((atomic))` キーワードを除くすべての C11 の機能
- 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。

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 17.0 デベロッパー・ガイドおよびリファレンス』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。

- *-f[no-]align-loops*: 2 の累乗のバイト境界でループをアライメントします。
- *-fp-model consistent*: 異なる最適化レベルや同じアーキテクチャーの異なるプロセッサーで、一貫した再現性のある結果を有効にします。

- *-qopt-report-annotate*: アノテーション付きソースリスト機能を有効にし、その形式を指定します。
- *-qopt-report-annotate-position*: アノテーション付きソースリスト機能を有効にし、ループの最適化によりインライン展開が行われた場合に最適化メッセージを表示するアノテーション付きソースの位置を指定します。

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 17.0 デベロッパー・ガイドおよびリファレンス』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。

オフロード DEFAULTMAP のデフォルト動作の変更

`DEFAULTMAP (TOFROM:SCALAR)` ローカルスカラーは、デフォルトではオフロードされません。
“default map: scalar” 節を指定する必要があります。
スカラー変数は、マップタイプが `TOFROM` の `MAP` 節で指定されたかのように処理されます。この節を指定しない場合、スカラー変数はマップされず、暗黙的に `FIRSTPRIVATE` 属性が指定されます。ディテクティブでは、1 つの `DEFAULTMAP` 節を指定できます。

OpenMP* ヘルパースレッドの削除

プログラム実行中の記録に使用されていた OpenMP* 監視スレッドが削除されました。ただし、ユーザーは、インテル® VTune™ Amplifier XE やその他のツールでこのスレッドを確認することができます。

先頭へ戻る

終了予定のサポート

終了したサポート

Red Hat* Enterprise Linux* 5 のサポートを終了

こ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。

第 3 世代インテル® Core™ プロセッサーの GFX オフロードのサポートを終了

第 3 世代インテル® Core™ プロセッサーのプロセッサー・グラフィックスへの GFX オフロードのサポートは、インテル® C++ コンパイラー 17.0 で終了しました。

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
16.3.2 ドライバーのサポートを終了しました。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。

32 ビット・ホストへのインストールのサポートを終了

32 ビット・ホストへのインストールのサポートは、このリリースで終了しました。32 ビット・ターゲット用コードの生成は 64 ビット・ホストでサポートされます (*-m32* コンパイラー・オプションを使用)。

_GFX_enqueue を削除

_GFX_enqueue は削除されました。*_GFX_offload* に変更してください。

先頭へ戻る

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静的にリンクされる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 -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: *MIC* 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 アーキテクチャーの 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で並列化されますが、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。そのため、cilk_for を使用する場合、オフロード実行が行われないと、バックアップ・コードの実行パフォーマンスに大きく影響する可能性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。

  回避策は、次のいずれかです。
  - 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。
```

別の方法として、スタティック・ランタイムを使用してインテル® Cilk™ Plus のオープンソース・バージョンをビルドできます。インテル® Cilk™ Plus の実装についての詳細は、https://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・バージョンで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。

IA-32 アプリケーション用インテル® 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、英語) を IA-32 アプリケーション用インテル® 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。

GCC 6.0 は未サポート

- GCC 6.0 とインテル® C++ コンパイラー 16.0 Update 2 の使用はサポートしていません。インテル® C++ コンパイラー 16.0 Update 2 を GCC 6.0 とともに使用して C++ ファイルをコンパイルすると、次のエラーが出力されます。

  "エラー #3802: gcc6 ヘッダーで使用されている SFINAE の実装が不完全であるため、バージョン 16.0 Update 2 は GNU* バージョン 6.0 以降と互換性がありません。続行する場合は、オプション -wd3802 を使用してください。"

後置する戻り型の外部定義の decltype 式でテンプレート依存関数の呼び出しが行われた場合のスプリアスエラー

- これはインテル® C++ コンパイラー 16.0 Update 2 における既知のリグレッションです。次に例を示します。

```
template <class T>
struct C {
  int then(int*,int*);
  template <class T2>
  auto then(T2 arg) -> decltype(this->then(&arg, &arg));
};
template <class T>
template <class T2>
auto C<T>::then(T2 arg) -> decltype(this->then(&arg, &arg)) { return 0; }
void foo() {
```

```
    C<int> f;
    f.then(99);
}
```

この問題を回避するには、定義を内部に移動するか、(戻り型を明示的に宣言して) 後置する戻り型を使用しないようにします。

c++14 の constexpr の緩和と Boost 問題

- -std=c++14 モードで Boost を使用していて constexpr 機能に関連すると思われるコンパイルエラーが表示された場合、boost_1_59_0/boost/config/compiler/gcc.hpp で BOOST_NO_CXX14_CONSTEXPR を定義してください。変更する行 (行 256 の前後) を次に示します。

  変更前:

  ```
  #if !defined(__cpp_constexpr) || (__cpp_constexpr < 201304)
  #  define BOOST_NO_CXX14_CONSTEXPR
   #endif
  ```

  変更後:

  ```
  //#if !defined(__cpp_constexpr) || (__cpp_constexpr < 201304)
  #  define BOOST_NO_CXX14_CONSTEXPR
   //#endif
  ```

先頭へ戻る

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

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、インテル製品の欠陥や故障によって人身事故が発生するような用途向けに使用することを前提としたものではありません。

インテル製品は、予告なく仕様や説明が変更されることがあります。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、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。

先頭へ戻る

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。