インテル® Parallel Studio XE 2015 Composer Edition for C++ Windows* インストール・ガイドおよびリリースノート

2014 年 10 月 14 日

目次

1 概要

このドキュメントでは、製品のインストール方法、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。リリースノートの最新アップデートについては、インテル® ソフトウェア開発製品レジストレーション・センターにリストされているリリースノートを参照してください。

インテル® Parallel Studio XE Composer Edition は統合的なソフトウェア開発ツールであり、各コンポーネントは異なるライセンスの下で提供されます。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。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。各コンポーネントの新機能の詳細は、各コンポーネントのリリースノートを参照してください。

1.1.1 Update 1

- [ツール] > [オプション] および [プロジェクト] メニューの項目名の変更

1.1.2 インテル® C++ Composer XE 2013 SP1 以降 (インテル® Parallel Studio XE 2015 Composer Edition での変更)

- インテル® グラフィックス・テクノロジーへのコンパイラー・オフロードをサポート
- インテル® グラフィックス・テクノロジー向けネイティブコード生成をサポート
- 新しい最適化レポートのインターフェイス、構造、オプション (既存の /Qopt-report、/Qvec-report、/Qopenmp-report、/Qpar-report オプションを使用しているユーザーは詳細を確認することを強く推奨)
- レポート情報をソースに統合し関連領域へのハイパーリンクとともに表示する最適化レポートの新しい IDE 統合。詳細は、ユーザー・リファレンス・ガイドを参照してください。
- C++11 言語をフルサポート
- OpenMP* 4.0 の機能を追加サポート
- インテル® Cilk™ Plus の変更
- Microsoft* の vectorcall 呼び出し規約をサポート
- オンライン・インストーラーでのカスタム・インストール・パッケージの作成
- データ・アライメントに関係なく同じコードを実行する /Qopt-dynamic-align- オプション
- PGO によるスレッドセーフなプロファイル生成が可能
- 構造体フィールドへのポインターの問題に対するポインターチェッカーの診断レベルを制御
- aligned_new ヘッダー
- ラムダ関数のデバッグを強化
- 非連続メモリーのコピーを許可する拡張オフロード構文
- SIMD データ型 (例えば、__m128) で算術演算および論理演算の使用を許可
- 関数ごとにインライン動作を制御する新しいプラグマ/宣言子
- PGO .dyn ファイル名にカスタム・プリフィックスを追加する新しい INTEL_PROF_DYN_PREFIX 環境変数
- インテル® C++ コンパイラーで使用する Visual Studio* のツールセットを明示的に指定する Microsoft* Visual Studio* プロパティー、[Base Platform Toolset (ベース・プラットフォーム・ツールセット)] を IDE 統合に追加。
- インテル® MKL クラスター構成で使用する MPI ライブラリーを明示的に指定する Microsoft* Visual Studio* プロパティー、[Use MPI Library (MPI ライブラリーを使用する)] を IDE 統合に追加。
- フラットなパフォーマンス・プロファイルのアプリケーションに対するアドバイスを提供するようにパフォーマンス・ガイドを強化。
- スタティック解析は非推奨 (廃止予定)
- Windows* XP はサポートされていません
- Microsoft* Visual Studio* 2008 はサポートされていません
- インテル® C++ コンパイラー 15.0.0
- インテル® MIC アーキテクチャー向けインテル® Debugger Extension を更新
- インテル® MKL がバージョン 11.2 にアップデート
- インテル® IPP 8.2
- インテル® TBB 4.3

1.2 製品の内容

*インテル® Parallel Studio XE 2015 Composer Edition for C++ Windows** は、次のコンポーネントで構成されています。

- インテル® C++ コンパイラー 15.0.0。Windows* オペレーティング・システムを実行する IA-32、インテル® 64 アーキテクチャー・システム、インテル® Xeon Phi™ コプロセッサー、またはインテル® グラフィックス・テクノロジーで動作するアプリケーションをビルドします。
- 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インテル® Debugger Extension
- インテル® MKL 11.2
- インテル® IPP 8.2
- インテル® TBB 4.3
- Microsoft* 開発環境への統合
- サンプルプログラム
- 各種ドキュメント

1.3 動作環境

アーキテクチャー名についての説明は、http://intel.ly/q9JVjE (英語) を参照してください。

- 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)
    - 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)
- 4GB のディスク空き容量 (すべての機能およびすべてのアーキテクチャー)
- 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)
    - オフロードコードのデバッグには Microsoft* Visual Studio* 2012 または 2013 が必要
- 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開発/テスト
    - 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 および E3-1285L v3 (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。
    - インテル® グラフィックス・テクノロジー対応の最新の 32 ビットまたは 64 ビット・グラフィックス・ドライバー (http://downloadcenter.intel.com から入手できます)
    - Windows* 用 binutils (http://intel.ly/1fHX7xO から入手できます)
        - binutils をインストールした後、`ld.exe` を含むディレクトリーを PATH に追加する必要があります。
- Microsoft* Windows* 7、Microsoft* Windows* 8、Microsoft* Windows* 8.1、Microsoft* Windows* 2008 SP2 (IA-32 のみ)、Microsoft* Windows Server* 2008 (R2 SP1)、Microsoft* Windows* HPC Server 2008、Microsoft* Windows Server* 2012 (エンベデッド・エディションはサポートしていません)
    - Microsoft* Windows* 8、Microsoft* Windows* 8.1 および Microsoft* Windows Server* 2012 では、製品は「デスクトップ」環境にインストールされます。「Windows* 8 UI」アプリケーションの開発はサポートされていません。[4]
- IA-32 対応アプリケーションまたはインテル® 64 対応アプリケーションのビルドに、Microsoft* Visual Studio* 開発環境あるいはコマンドライン・ツールを使用する場合は、次のいずれか:
    - Microsoft* Visual Studio* 2013 Professional Edition 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* 2012 Professional Edition 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* 2010 Professional Edition 以上 (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること)
- IA-32 アーキテクチャー・アプリケーションのビルドに、コマンドライン・ツールのみを使用する場合は、次のいずれか:
    - Microsoft* Visual C++* Express 2013 for Windows Desktop
    - Microsoft* Visual C++* Express 2012 for Windows Desktop
    - Microsoft* Visual C++* 2010 Express Edition
- インテル® 64 対応アプリケーションのビルドに、コマンドライン・ツールのみを使用する場合は、次のいずれか:
    - Microsoft* Visual C++* Express 2013 for Windows Desktop
    - Microsoft* Visual C++* Express 2012 for Windows Desktop
    - Microsoft* Windows* Software Development Kit for Windows* 8
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

注:

1. Microsoft* Visual Studio* 2010 では、このコンポーネントがデフォルトで含まれています。
2. 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。ただし、アプリケーションでインテル® IPP またはインテル® TBB を使用している場合、そのアプリケーションの実行には、インテル® SSE2 命令対応のプロセッサーが必要です。
3. アプリケーションは、上記の開発用と同じ Windows* バージョンで実行できます。また、Windows* XP よりも前の非エンベデッドの Microsoft* Windows* 32 ビット・バージョンでも実行できますが、インテルではこれらの互換性テストは行われてい

ません。開発アプリケーションが、古いバージョンの Windows* にはない Win32* API ルーチンを使用している可能性があります。アプリケーションの互換性テストをご自身の責任で行ってください。アプリケーションを実行するには、特定のランタイム DLL をターゲットシステムにコピーしなければならないことがあります。

4. インテル® C++ コンパイラーは、Windows 8* UI アプリの開発をサポートしていません。インテルでは、ユーザーの皆様のご意見を常に参考にしています。例えば、Windows* 8 UI アプリケーションにインテル® C++ コンパイラーまたはその他のインテル® ソフトウェア開発製品の機能を利用したい方は、インテル® プレミアサポート (https://premier.intel.com/) からご意見をお送りください。
Windows* 8 UI アプリケーションの開発で、このサポートされていないインテル® ソフトウェア開発製品の機能をテストすることにご興味のある方は、http://intel.ly/WLeXRo (英語) をお読みください。

1.3.1 Visual Studio* 2008 はサポートされていません

Visual Studio* 2008 のサポートを終了しました。新しいバージョンの Visual Studio* に移行してください。

1.3.2 Windows* XP はサポートされていません

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

1.4 ドキュメント

製品ドキュメントは、「インストール先フォルダー」で示されているように、`Documentation` フォルダーに保存されています。

1.4.1 Microsoft* Visual Studio* のオンラインヘルプ形式の変更

オンラインヘルプ形式がブラウザーベースになりました。Microsoft* Visual Studio* の [ヘルプ] メニューからインテルのドキュメントを参照する場合、または F1 キー、ダイアログボックスにあるヘルプボタン、その他の GUI で状況依存ヘルプを参照する場合、デフォルトのブラウザーに対応するヘルプトピックが表示されます。デフォルトのブラウザーによっては、いくつかの小さな問題が発生することがあります。次のような既知の問題があります。

- [ヘルプ設定の設定] が [ブラウザーで起動] に設定されている場合、[ツール] > [オプション] > [F# ツール] または [ツール] > [オプション] > [Intellitrace] で F1 キーを押すと、ブラウザーが 2 つ開きます。
- **Chrome*:** 検索またはキーワードからトピックを表示すると、目次が同期しません。[トピックを同期] も動作しません。
- **Firefox*:** 目次が表示されなくなることがあります。検索の大文字と小文字は区別されます。
- **Safari*:** Windows* の反応が遅くなります。

1.4.2 Windows Server* 2012 の Microsoft* Internet Explorer* 10 でドキュメントが表示されない問題

Windows Server* 2012 の Internet Explorer* 10 でヘルプまたはドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、信頼済みサイトのリストに "about:internet" を追加します。オプションで、ドキュメントを参照した後に信頼済みサイトのリストから "about:internet" を削除できます。

1.4.3 Windows Server* 2012 で Visual Studio* 2012 のドキュメントを表示できない場合

Windows Server* 2012 で Visual Studio* 2012 のヘルプまたはドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、[インターネット] ゾーンで [MIME スニッフィングを有効にする] および [アクティブ スクリプト] を有効にします。

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

1.5 サンプル

製品コンポーネントのサンプルは、「インストール先フォルダー」の説明にある `Samples` フォルダーに用意されています。

1.6 日本語サポート

インテル® コンパイラーは、日本語と英語の両方を備えたインストーラーで日本語をサポートしています。エラーメッセージ、ビジュアル開発環境ダイアログ、ドキュメントの一部が英語のほかに日本語でも提供されています。エラーメッセージやダイアログの言語は、システムの言語設定に依存します。日本語版ドキュメントは、`Documentation` および `Samples` ディレクトリー以下の `ja_JP` サブディレクトリーにあります。

日本語版は、インテル® Parallel Studio XE 2015 Composer Edition 初期リリースの後の Update で提供されます。

日本語サポート版を英語のオペレーティング・システムで使用する場合や日本語のオペレーティング・システムで英語サポート版を使用する場合は、http://intel.ly/oZjpZs (英語) の説明を参照してください。

1.7 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: 代理店がテクニカルサポートを提供している場合は、インテルではなく代理店にお問い合わせください。

2 インストール

2.1 オンライン・インストーラー

インテル® Parallel Studio XE では、ダウンロード版インストール・パッケージが、サイズの小さいオンライン・インストーラーになりました。オンライン・インストーラーは、選択したパッケージを動的にダウンロードし、インストールします。このパッケージを使用するには、インターネット接続が必要です。インターネット・プロキシーを使用している場合は、プロキシーの設定が必要になることがあります。インターネット接続が利用できない環境でインストールする場合は、このオンライン・インストール・パッケージではなく、フルパッケージを利用してください。オンライン・インストーラーをダウンロードして実行ファイルとして保存し、コマンドラインから起動することもできます。

2.1.1 オンライン・インストーラーによりダウンロードされるコンテンツの格納

オンライン・インストーラーは、ほかのシステムにコピーしてオフラインで使用できるように、ダウンロードしたコンテンツを標準インストール・パッケージ形式で格納します。デフォルトのダウンロード・ディレクトリーは `<Program Files>\Intel\Download` です。この場所は、オンライン・インストーラーの `"--download-dir [FOLDER]"` コマンドライン・オプションで変更できます。オンライン・インストーラーには、インストールしないでパッケージを作成できるダウンロード専用モードも用意されています。このモードは、`"--download-only"` コマンドライン・オプションで有効になります。

2.2 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) のインストール

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® Parallel Studio XE for Windows* 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 Windows* 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。

2.3 インテル® Software Manager

インテル® Software Manager は、製品アップデートの配信方法を簡素化し、現在インストールされているすべてのインテル® ソフトウェア製品のライセンス情報とステータスを表示します。

将来の製品設計の参考のため、製品使用状況に関する匿名情報をインテルに提供する、インテル® ソフトウェア向上プログラムに参加できます。このプログラムは、デフォルトで無効になっていますが、インストール中または後から有効にして参加できます。参加はいつでも

取りやめることができます。詳細は、http://intel.ly/SoftwareImprovementProgram (英語) を参照してください。

2.4 インストール

本製品のインストールには、有効なライセンスファイルまたはシリアル番号が必要です。本製品を評価する場合には、インストール時に [製品を評価する (シリアル番号不要)] オプションを選択してください。

インストールを開始するには、実行ファイル (.EXE) をダブルクリックします。利用可能なダウンロード・ファイルには各種あり、それぞれ異なるコンポーネントの組み合わせを提供していることに注意してください。ダウンロード・ページを注意深くお読みになり、適切なファイルを選択してください。

新しいバージョンをインストールする前に古いバージョンをアンインストールする必要はありません。新しいバージョンは古いバージョンと共存可能です。

2.4.1 PATH 環境変数の変更によるコマンドシェル (cmd.exe) への一時的な影響

Windows* 7 または Windows* 8 では、インストーラーが PATH 環境変数に項目を追加すると、PATH の長さが非常に長くなり (2000-4000 文字)、システムを再起動するまで Windows* コマンドプロンプト (cmd.exe) が動作しなくなることがあります。システムを再起動しても同じ問題が発生する場合は、テクニカルサポートまでお問い合わせください。

2.4.2 サイレント・インストール

自動インストール、「サイレント」インストール機能についての詳細は、http://intel.ly/nKrzhv (英語) を参照してください。

2.4.2.1 非インタラクティブ・カスタム・インストールのサポート

インテル® Parallel Studio XE 2015 は、「インタラクティブ」インストール中のユーザーの選択肢を (サイレント・インストールに使用できる) 設定ファイルに保存する機能をサポートしています。この設定ファイルは、コマンドライン・インストールで次のオプションを使用すると作成されます。

- `--duplicate=config_file_name`: 設定ファイルの名前を指定します。フルパスのファイル名が指定された場合、"`--download-dir`" は無視され、設定ファイルがあるディレクトリーにインストール・パッケージが作成されます。
- `--download-dir=dir_name`: 設定ファイルを作成する場所を指定します (オプション)。このオプションを指定しない場合、インストール・パッケージおよび設定ファイルはデフォルトのダウンロード・ディレクトリーに作成されます。

```
%Program Files%\Intel\Download\<package_id>
```

例: `w_ccompxe_online_2015.0.0XX.exe --duplicate=ic15_install_config.ini --download-dir= "C:\temp\custom_pkg_ic15"`

設定ファイルおよびインストール・パッケージが "`C:\temp\custom_pkg_ic15`" に作成されます。

2.4.3 クラスターでのインストール

インストールするマシンに Microsoft* Compute Cluster Pack のライセンスがあり、クラスターメンバーの場合、「フル・インストール」を選択すると、そのクラスターのアクセス可

能なすべてのノードに製品がインストールされます。「カスタム・インストール」を選択すると、現在のノードのみにインストールするオプションを選択できます。

2.4.4 ライセンスサーバーの使用

「フローティング・ライセンス」を購入された場合は、ライセンスファイルまたはライセンスサーバーを使用したインストール方法について、http://intel.ly/pjGfwC (英語) を参照してください。この記事には、多様なシステムにインストールすることができるインテル・ライセンス・サーバーに関する情報も記述されています。

2.5 製品の変更、更新、削除

Windows* のコントロールパネルの [プログラムの追加と削除] でインストールまたは削除する製品コンポーネントを変更します。

製品のアップデート・バージョンをインストールする際、古いバージョンを最初にアンインストールする必要はありません。複数のバージョンのコンパイラーをインストールし、その中から選択して使用することができます。新しいバージョンのコンパイラーを削除した場合、以前のバージョンの Microsoft* Visual Studio* への統合を再インストールする必要があります。

2.6 インストール先フォルダー

インストール・フォルダーの構成を以下に示します。一部含まれていないフォルダーもあります。

- C:\Program Files\Intel\Composer XE 2015
  - bin
    - ia32
    - ia32_gfx
    - ia32_intel64
    - intel64
    - intel64_gfx
    - intel64_mic
    - sourcechecker
  - compiler
    - include
      - cilk
      - gfx
      - ia32
      - intel64
      - mic
    - lib
      - ia32
      - ia32_gfx
      - intel64
      - intel64_gfx
      - mic
    - perf_headers
      - c++
  - debugger
    - gdb

            - LICENSES
            - src
            - target
            - w64_mic
        - debuggerextension
            - mic
- Documentation
    - en_US
        - compiler_c
        - debugger
        - gs_resources
        - ipp
        - mkl
        - ssadiag_docs
        - tbb
        - tutorials
    - ja_JP
        - compiler_c
        - debugger
        - gs_resources
        - ipp
        - mkl
        - ssadiag_docs
        - tbb
        - tutorials
    - msvhelp
        - 1033
        - 1041
- ipp
    - bin
        - ia32
        - intel64
    - examples
    - include
    - interfaces
    - lib
        - ia32
        - intel64
        - mic
    - tools
        - ia32
        - intel64
- mkl
    - benchmarks
        - linpack
        - mp_linpack
    - bin
        - ia32
        - intel64

        - examples
        - include
            - fftw
            - ia32
            - intel64
            - mic
        - interfaces
            - fftw2xc
            - fftw3xc
        - lib
            - ia32
            - intel64
            - mic
        - tests
        - tools
            - builder
    - redist
        - ia32
            - compiler
            - ipp
            - mkl
            - tbb
        - intel64
            - compiler
            - ipp
            - mkl
            - tbb
    - Samples
        - en_US
            - C++
            - ipp
            - mkl
        - ja_JP
            - C++
            - ipp
            - mkl
    - tbb
        - bin
        - examples
            - common
            - concurrent_hash_map
            - concurrent_priority_queue
            - GettingStarted
            - graph
            - parallel_do
            - parallel_for
            - parallel_reduce
            - pipeline
            - task

        - task_group
        - task_priority
        - test_all
    - include
        - serial
        - tbb
    - lib
        - ia32
        - intel64
        - mic

bin、include および lib 配下のフォルダーは次のとおりです。

- ia32: IA-32 上で動作するアプリケーションのビルドに使用するファイル
- intel64: インテル® 64 上で動作するアプリケーションのビルドに使用するファイル
- ia32_intel64: IA-32 上のコンパイラーでインテル®64 上で動作するアプリケーションをビルドする場合に使用するファイル

英語以外の Windows* システムにインストールする場合、Program Files フォルダー名が異なる場合があります。インテル® 64 アーキテクチャー・システムでは、フォルダー名は Program Files (X86) またはそれに相当する名前です。

デフォルトでは、アップデートによって既存のディレクトリーの内容が置換されます。最初のアップデートをインストールするときに、以前のインストールとは別に新しいアップデートをインストールして、システムに両方のファイルを残すオプションを選択できます。両方を残すオプションを選択した場合、古いアップデートのトップレベルのフォルダー名は Composer XE 2015.*nnn* (*nnn* はアップデート番号) に変更されます。

3 インテル® C++ コンパイラー

このセクションでは、インテル® C++ コンパイラーの変更点、新機能、および最新情報をまとめています。

3.1 互換性

バージョン 11 では、IA-32 システムのデフォルトでのコード生成において、アプリケーションを実行するシステムでインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) がサポートされていると仮定するように変更されました。

3.2 新機能と変更された機能

インテル® C++ コンパイラーがバージョン 15.0 になりました。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。

- 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
- インテル® グラフィックス・テクノロジー向けネイティブコード生成をサポート
- 新しい最適化レポートのインターフェイス、構造、オプション
- レポート情報をソースに統合し関連領域へのハイパーリンクとともに表示する最適化レポートの新しい IDE 統合。詳細は、ユーザー・リファレンス・ガイドを参照してください。

- C++11 言語をフルサポート (15.0 での新機能を含む) (/Qstd=c++11)
  - 値カテゴリー (N3055)
  - alignas および alignof (N2341)
  - decltype 拡張 (N3049、N3276)
  - 継承コンストラクター (N2540)
  - ユーザー定義リテラル (N2765)
  - thread_local (N2659)
  - インテル® メニー・インテグレーテッド・コア・アーキテクチャーのインテル® 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) ディストリビューションの一部として提供されるヘッダーファイルおよびライブラリーは、experimental 4.7.0 バージョンです。このバージョンは、4.7.0 gcc ライブラリーの最終リリースで利用可能な一部の gcc 機能をフルサポートしていません。特に、このバージョンの gcc は allocator traits をサポートしていません。
- OpenMP* 4.0 の機能を追加サポート
- インテル® Cilk™ Plus の変更
- Microsoft* の vectorcall 呼び出し規約をサポート
- スタティック解析は非推奨 (廃止予定)
- aligned_new ヘッダー
- ラムダ関数のデバッグを強化
- 非連続メモリーのコピーを許可する拡張オフロード構文
- SIMD データ型 (例えば、__m128) で算術演算および論理演算の使用を許可
- 関数ごとにインライン動作を制御する新しいプラグマ/宣言子
- PGO .dyn ファイル名にカスタム・プリフィックスを追加する新しい INTEL_PROF_DYN_PREFIX 環境変数
- インテル® C++ コンパイラーで使用する Visual Studio* のツールセットを明示的に指定する Microsoft* Visual Studio* プロパティー、[Base Platform Toolset (ベース・プラットフォーム・ツールセット)] を IDE 統合に追加。
- インテル® MKL クラスター構成で使用する MPI ライブラリーを明示的に指定する Microsoft* Visual Studio* プロパティー、[Use MPI Library (MPI ライブラリーを使用する)] を IDE 統合に追加。
- フラットなパフォーマンス・プロファイルのアプリケーションに対するアドバイスを提供するようにパフォーマンス・ガイドを強化。

3.2.1 インテル® グラフィックス・テクノロジー向けネイティブコード生成をサポート

デフォルトでは、コンパイラーはインテル® グラフィックス・テクノロジーにオフロードするカーネル用の仮想 ISA コードを生成します。仮想 ISA はターゲットに依存せず、プラットフォームにインテル® グラフィックス・プロセッサーが統合され、適切なインテル® HD グラフィックス・ドライバーがインストールされているシステムで動作します。インテル® HD グラフィックス・ドライバーには、オフロード・ランタイム・サポートおよび Jitter (JIT コンパイラー) が含まれていて、実行時にアプリケーションを実行するプラットフォーム向けに仮想 ISA をネイティブ ISA に変換してプロセッサー・グラフィックスへのオフロードを行います。Jitter は現在のプロセッサー・グラフィックス情報を実行時に取得します。新しいオプション /Qgpu-arch:<arch> (Windows*) および -mgpu-arch=<arch> (Linux*) を使用すると、リンク時にネイティブ ISA を生成できます。オプションの詳細は、ユーザー・リファレンス・ガイドを参照してください。

3.2.2 スタティック解析は非推奨 (廃止予定)

スタティック解析のサポートは古いオプション (非推奨) で、将来のリリースで削除される予定です。ご意見やお問い合わせは、こちらまでお寄せください。スタティック解析を行うと Microsoft* Visual Studio* IDE のメニューが無効になることに注意してください。メニューを有効にするには、環境変数 __INTEL_STATIC_ANALYSIS を設定して Visual Studio* を再起動してください。

3.2.3 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

サポートは、同期 (`#pragma offload target(gfx)` および `cilk_for` 並列ループ) または非同期 (`#pragma offload target(gfx_kernel)` および `gfx_rt.h` ヘッダーで提供される API) オフロード実装のいずれかで提供されます。詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「主な機能」 > 「インテル® グラフィックス・テクノロジー」を参照してください。既知の制限事項は、リリースノートを参照してください。

3.2.4 新しい最適化レポートのインターフェイス、構造、オプション (インテル® C++ コンパイラー 15.0)

インテル® C++ コンパイラー 15.0 で、4 種類の最適化レポート (/Qopt-report、/Qvec-report、/Qopenmp-report、/Qpar-report) が 1 つの /Qopt-report インターフェイスに統合されました。情報の表示方法、内容、精度が見直され、どの最適化がコンパイラーにより行われたか、最適なパフォーマンスを達成するにはどのようなチューニングを行えばよいか、ユーザーが理解しやすいように変更されました。

並列ビルドの問題により、このレポートはデフォルトで stderr に出力されません。代わりに、各オブジェクト・ファイルごとにレポートを含む出力ファイル (拡張子 .optrpt) が、コンパイルの出力ディレクトリー (オブジェクト・ファイルが生成されるディレクトリー) に生成されます。この動作を変更するには、/Qopt-report-file オプション (例: /Qopt-report-file:stderr) を使用します。

/Qvec-report、/Qopenmp-report、/Qpar-report オプションは廃止予定ですが、現在は /Qopt-report オプションの対応する値にマップされます。レポートの内容および形式、デフォルトの出力先は新しい opt-report と同じになります。

変更の詳細についてドキュメントを参照することを強く推奨します。詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・リファレンス」 > 「コンパイラー・オプションのカテゴリーと説明」 > 「最適化レポートオプション」を参照してください。

3.2.5 OpenMP* 機能の追加サポート (インテル® C++ コンパイラー 15.0)

インテル® C++ コンパイラー 15.0 では、次の OpenMP* 4.0 機能を追加しました。

- `cancel` および `cancellation point` 宣言子
- `depend` 句 (`task` 宣言子)

OpenMP* 4.0 のユーザー定義リダクションはサポートしていません。

3.2.6 インテル® C++ コンパイラー 15.0 のインテル® Cilk™ Plus の変更点

インテル® C++ コンパイラー 15.0 では、インテル® Cilk™ Plus の次の新機能が追加されています。

- #pragma simd 構文の代わりに、キーワードで明示的なベクトル・プログラミングを実装する機能。キーワードは _Simd、_Safelen、_Reduction です。詳細は、『インテル® 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。
- SIMD ベクトル関数 (__declspec(vector)) 内の "レーン ID" を示す __intel_simd_lane() 組込み関数
- Doxygen* を使用してインテル® Cilk™ Plus のドキュメントを生成可能。詳細は、compiler\include\cilk\ReadMe.html を参照してください。
- スカラー関数のベクトル固有のオーバーロードを定義する新しい __declspec(vector_variant(...))

3.2.7 Microsoft* の vectorcall 呼び出し規約をサポート

Microsoft* Visual Studio* 2013 で追加された Microsoft* の vectorcall 呼び出し規約をサポートしました。

3.2.8 アライメント宣言を含むクラス型でデータを正しく動的に割り当てる方法を提供する aligned_new ヘッダー

C++11 ではクラス型のデータ・アライメントを指定できますが、これらの型に基づくアライメントで実際にデータを割り当てる標準メカニズムはありません。aligned_new は、データを適切にアライメントする new および delete のオーバーロードを提供します。

3.2.9 関数ごとにインライン動作を制御する新しいプラグマ/宣言子

インテル® C++ コンパイラー 15.0 では、関数ごとのインライン動作を制御するため、2 つの新しいプラグマ宣言子 inline-max-per-routine および inline-max-total-size が追加されました。詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・リファレンス」 > 「プラグマ」 > 「インテル® コンパイラー固有のプラグマ・リファレンス」を参照してください。

3.2.10 スタティック解析機能 (旧:「スタティック・セキュリティー解析」または「ソースチェッカー」) にはインテル® Inspector XE が必要

バージョン 11.1 の「ソースチェッカー」機能が拡張され、「スタティック解析」に名称が変更されました。スタティック解析を有効にするコンパイラー・オプションはバージョン 11.1 と同じですが (例: `/Qdiag-enable:sc`)、解析結果がコンパイラー診断結果ではなく、インテル® Inspector XE で表示可能なファイルに出力されるようになりました。

3.2.11 インテル® C++ プロジェクト・ファイルの互換性

インテル® C++ プロジェクト・ファイル (`.icproj`) の形式がバージョン 15.0 で変更されました。インテル® C++ の古いバージョンで作成されたプロジェクトを開くと、プロジェクトの変換が必要である旨のメッセージが表示されます。バージョン 15.0 のプロジェクトを古いバージョンのインテル® C++ 統合で使用することはできません (ただし、古いバージョンのコンパイラーは、[ツール] > [オプション] > [Intel Compilers and Tools (インテル(R) コンパイラーおよびツール)] > [Intel C++ (インテル(R) C++)] > [Compilers (コンパイラー)] から使用できます)。

3.3 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・オプションの詳細に関しては、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の

3.3.1 インテル® C++ コンパイラー 15.0 の新規および変更されたコンパイラー・オプション

- /Qmic
- /Qgpu-arch:<arch>[,<arch>]
- /Qopt-report-names:[mangled|unmangled]
- インテル® グラフィックス・テクノロジーへのオフロード用の /Qoffload-option の <tool> として jit を追加
- /Qno-builtin-<func>
- /Gv (vectorcall をデフォルトの呼び出し規約にします)
- /Qopt-dynamic-align[-]
- /Qprof-gen:threadsafe
- /Qopt-report (レベル 4 および 5 を追加)
- /Qopt-report-file:{stdout | stderr | <file>}
- /Qopt-report-per-object
- /Qopt-report-filter:<string>
- /Qopt-report-format:[text|vs]
- /Qopt-report-embed[-]
- /Qcheck-pointers-narrowing[-]
- /Qicl-
- /Zc:trigraphs[-]
- /fast に /fp:fast=2 が含まれます
- /Qeliminate-unused-debug-types[-]
- /I-

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® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・オプション」を参照してください。

3.3.2 山括弧付きのインクルード・ファイルの検索を制御する /I- オプション

/I- オプションを使用すると、指定したコマンドライン・インクルード・パスを効率良く検索できます。/I- オプションの前に /I オプションで指定したディレクトリーは、`#include "file"` 形式のヘッダーでのみ検索されます。`#include <file>` のように山括弧付きのヘッダーでは検索されません。コマンドラインで /I- の後に /I オプションで追加ディレクトリーを指定した場合、追加ディレクトリーはすべての #include で検索されます。また、/I- は、`#include "file"` のように引用符付きのヘッダーの最初の検索ディレクトリーとして現在のファイル・ディレクトリーを使用することを禁止します。

3.3.3 データ・アライメントに関係なく同じコードを実行する /Qopt-dynamic-align- オプション

デフォルトでは、コンパイラーは、浮動小数点演算の一貫性に影響するパフォーマンスを向上するため、データのアライメントに応じて実行するように複数のコードパスを生成します。この動作を無効にするには、/Qopt-dynamic-align- オプションを使用します。

3.3.4 PGO によるスレッドセーフなプロファイル生成が可能

マルチスレッド・アプリケーションでプロファイル情報を安全に生成するには、/Qprof-gen:threadsafe オプションを使用します。

3.3.5 構造体フィールドへのポインターの問題に対するポインターチェッカーの診断レベルを制御

構造体フィールドへのポインターの問題に対するポインターチェッカーの診断を無効にするには、/Qcheck-pointers-narrowing- オプションを使用します。

3.3.6 廃止予定のオプション

次の方法で廃止予定のすべてのコンパイラー・オプションを確認できます。

1) [スタート] メニューからコマンドプロンプトを開きます: [スタート] > [すべてのプログラム] > [Intel Parallel Studio XE 2015] > [Command Prompt (コマンドプロンプト)] > [Parallel Studio XE with Intel Compiler XE v15.0 [Update xx] (インテル(R) コンパイラー XE 15.0 [Update xx])] > [IA-32 Visual Studio xxxx mode (IA-32 Visual Studio xxxx モード)] または [Intel(R) 64 Visual Studio xxxx mode (インテル(R) 64 Visual Studio xxxx モード)] を選択します。
2) 次のコマンドを実行します。

```
>> icl /? deprecate
```

3.4 その他の変更

3.4.1 [ツール] > [オプション] および [プロジェクト] メニューの項目名の変更

インテル® Parallel Studio XE 2015 Update 1 から、インテル® コンパイラー関連の一部の項目名が変更されました。

- [ツール] > [オプション] の左ペインにある [Intel Composer XE (インテル(R) Composer XE)] が [Intel Compilers and Tools (インテル(R) コンパイラーおよびツール)] になりました。利用可能な設定 (インクルード・ディレクトリー、コードカバレッジの設定、パフォーマンス・ライブラリーの設定、その他) は変更されていません。
- [プロジェクト] メニューまたはプロジェクトを右クリックして表示されるコンテキスト・メニューの [Intel Composer XE (インテル(R) Composer XE)] が [Intel Compiler (インテル(R) コンパイラー)] になりました。

3.4.2 ビルド環境コマンドスクリプトの変更

ビルド環境を構築するコマンド・ウィンドウ・スクリプトが使用する Microsoft* Visual Studio* バージョンを任意で指定できるよう変更されました。ビルド環境ウィンドウを開くのに、定義済みのスタート・メニュー・ショートカットを使用していない場合は、次のコマンドを使用して適切な環境を構築してください。

```
"<install-dir>\bin\compilervars.bat" arch [vs]
```

arch はビルドする対象アーキテクチャーを指定します。次のいずれかの値を指定できます。

- `ia32`
- `ia32_intel64`
- `intel64`

インテル® メニー・インテグレーテッド・コア・アーキテクチャーをターゲットにしている場合は、開発システム (32 ビットまたは 64 ビット) に応じて、`ia32_intel64` または `intel64` のいずれかを使用する必要があります。

vs は任意で指定します。次のいずれかの値を指定できます。*vs* が指定されていない場合は、コマンドライン統合用にインストール時に指定された Visual Studio* のバージョンがデフォルトで使用されます。

- `vs2013`
- `vs2012`
- `vs2010`

また、インテル® Visual Fortran コンパイラー 15.0 もインストールされている場合、このコマンドによりインテル® Visual Fortran コンパイラーを使用する環境も構築されます。

スクリプトファイル名 iclvars.bat および ifortvars.bat は、以前のリリースとの互換性のために保持されています。

3.4.3 OpenMP* スタティック・ライブラリーの削除

本リリースでは、OpenMP* ランタイム・ライブラリーが削除されました。これらのランタイム・ライブラリーは動的にリンクしてください。

3.4.4 バージョン管理システムでのインテル® C++ プロジェクトの使用

プロジェクトがバージョン管理システム (例: Microsoft* Visual SourceSafe* や Microsoft* Visual Studio* Team Foundation Server など) で管理されている場合、プロジェクトでインテル® C++ プロジェクト・システムを使用するには追加のステップが必要です。このトピックについての詳細な記事は、http://intel.ly/pImnpo (英語) を参照してください。

3.5 既知の問題

3.5.1 コンパイラーの既知の問題

3.5.1.1 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

`/Qcheck-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.dll をリンクする必要があります。`/MT` のようなオプションを `/Qcheck-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。

3.5.1.2 異なるコンパイラーを使用して 256 ベクトル・ビット型引数をコンパイルすると実行時にアクセス違反が発生するコードが生成される

2 つの異なるコンパイラー (Microsoft* Visual C++* 2013 コンパイラーおよびインテル® C++ コンパイラー15.0) を使用してアプリケーションを作成した場合、アライメントされていないデータアクセスによる一般保護違反が発生することがあります。この問題は、256 ベクトル・ビット型引数が呼び出しの際に参照で渡され、呼び出し元を Visual C++* でビルドし、その引数をインテル® C++ コンパイラーでビルドした関数でアクセスすると発生します。

原因は、256 ベクトル・ビット型引数のアライメントが一致しないためです。

<code> に AVX 以降の新しいコード値 (CORE-AVX-I、 CORE-AVX2、その他) を指定して /Qx<code> コンパイラー・オプションを使用した場合は、アプリケーションのソースコードで __mm256_stream_* (非テンポラルデータのロード/ストア組込み関数) が明示的に使用されない限り、アライメントされていないアクセス命令がこれらのインスタンスで実際に使用されるため、この問題は発生しません。

3.5.1.3 Internet Explorer *10 でオンライン・ドキュメントが表示されない問題

Internet Explorer* 10 によってドキュメントで使用されているスクリプトがブロックされることがあります。この場合、空白のページが表示されるか、「スクリプトや ActiveX コントロールを実行しないよう、Internet Explorer で制限されています。」というエラーメッセージが表示されます。ドキュメントを表示するには、[ブロックされているコンテンツを許可] をクリックしてください。エラーメッセージが表示されない場合は、Internet Explorer で [ツール] > [インターネット オプション] > [セキュリティ] > [レベルのカスタマイズ] を選択し、安全だとマークされていないコンテンツをダウンロードする前にダイアログを表示するように設定します。

3.5.2 Visual Studio* の既知の問題

3.5.2.1 Windows Server* 2012 で Visual Studio* 2012 または 2013 のドキュメントを表示できない場合

Windows Server* 2012 で Visual Studio* 2012 または 2013 のヘルプやドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、[インターネット] ゾーンで [MIME スニッフィングを有効にする] および [アクティブ スクリプト] を有効にします。

3.5.2.2 MSVCP90D.dll (またはその他の Microsoft* ランタイム DLL) が見つからない

サンプル・プロジェクト (および Microsoft* Visual C++* プロジェクト) を実行するときに Microsoft* Visual Studio* のランタイム DLL. が見つからない場合、ランタイムエラーが発生します。これは、マニフェスト・ファイルや SXS アセンブリーが見つからないことが原因です。この問題を解決するには、使用しているバージョンの Microsoft* Visual Studio* の redist フォルダー (デフォルトの場所は c:\program files[(x86)]\Microsoft Visual Studio X.X\VC\redist) に移動します。amd64、x86、Debug_NonRedist サブフォルダーで、必要なランタイムが含まれているフォルダーを探します (デバッグ・ライブラリーを探す場合は、ファイル名の最後が D のファイルが含まれているフォルダーを探します)。必要なランタイムが含まれているフォルダーが見つかったら、そのフォルダーの (.manifest ファイルを含む) すべての内容を、実行する .exe ファイルのあるフォルダーにコピーします。

3.5.2.3 Visual Studio* 2010 では /fp:precise がデフォルトでオン

Visual Studio* 2010 で作成または変換されたプロジェクトでは、デフォルトで `/fp:precise` コマンドライン・オプションがオンになります。このオプションは、パフォーマンスを低下させるいくつかの最適化を無効にして、浮動小数点演算の一貫性を向上させる「浮動小数点モデル」を設定します。インテルのデフォルトである `/fp:fast` に戻すには、プロジェクトのプロパティー・ページで [C/C++] > [Code Generation (コード生成)] > [Floating Point Model (浮動小数点モデル)] を Fast に変更します。

3.5.2.4 Visual Studio* 2010 の言語パック

インテル® Parallel Studio XE 2015 をインストールした後に Visual Studio* 2010 の新しい言語パックをインストールすると、[プロジェクト プロパティ] ダイアログにインテル® C++ コンパイラー固有のオプションが表示されなくなることがあります。その場合には、以下の手順を試してみてください。

1) "<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\1033" ディレクトリーが存在する場合は、すべてのファイルを "<program files>\MSBuild\

Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 15.0\<locale-ID>" にコピーします。

2) "<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033\" が存在する場合は、すべてのファイルを "<program files> \MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>" にコピーします。

* <locale-ID> は言語パックを表します。

別の方法として、インテル® Parallel Studio XE 2015 をアンインストールして、再インストールすることもできます。

3.5.3 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

3.5.3.1 _Cilk_shared の制限

- 仮想基本クラスで _Cilk_shared 属性を指定することはできません。
- 複数の基本クラスから _Cilk_shared 属性が指定されたクラスを派生させることはできません (複数の継承は許可されません)。
- _Cilk_shared 属性が指定されたクラスで仮想デストラクターを定義することはできません。
- _Cilk_shared 属性が指定されたクラスを別の _Cilk_shared クラスの基本クラスとして使用する場合、そのサイズが 8 の倍数になるように (必要に応じて、仮のフィールドを追加して) プログラマーが調整する必要があります。
- _Cilk_offload は、共有ライブラリー (DLL) を使うプログラムでは使用できません

3.5.3.2 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。

3.5.3.3 リンク時に検出されない見つからないシンボル

オフロード・コンパイル・モデルでは、インテル® MIC アーキテクチャーを対象とするバイナリーはダイナミック・ライブラリー (.so) として生成されます。ダイナミック・ライブラリーは、参照されている変数やルーチンをロード時に解決できるため、リンク時にこれらをすべて解決する必要はありません。この動作により、ロード時に解決できない一部の見つからない変数やルーチンを見逃してしまうことがあります。リンク時にすべての見つからないシンボルを識別して解決するには、次のコマンドライン・オプションを使用して未解決の変数をリストします。

```
/Qoffload-option,mic,compiler,"-z defs"
```

3.5.3.4 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。

```
$ icl -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;
```

3.5.3.5 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。

3.5.3.6 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) は含まれません。

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>\compiler\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。

3.5.3.7 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。

3.5.4 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

3.5.4.1 gfx_linker: : error : command 'ld.exe' exited with non-zero exit code -107374170

x64 プロジェクトでインテル® グラフィックス・テクノロジーへコードをオフロードすると、binutils に含まれている ld.exe でリンカーエラーが表示されることがあります。この問題を解決するには、64 ビット用の binutils bin ディレクトリーを PATH 環境変数に追加して、Microsoft* Visual Studio* を再起動してください。

3.5.4.2 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。

3.5.4.3 その他の問題

- Windows* 7 では、オフロードが行われたときにディスプレイをロックできません。アクティブ・ディスプレイが必要です。
- 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

3.5.5 インテル® Cilk™ Plus の既知の問題

- スチールが行われた後、対応する _Cilk_sync の前に SEH 例外がスローされると、Microsoft* C++ 構造化例外処理 (SEH) は失敗します。

3.5.6 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 (/Qipo) 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。この問題を回避するには、/Qipo を無効にします。Visual Studio* では、[プロジェクト] > [プロパティ ページ] > [C/C++] > [Optimization (最適化)] > [Interprocedural Optimization (プロシージャー間の最適化)] を「No (いいえ)」に設定します。

3.5.7 スタティック解析の既知の問題

3.5.7.1 仮想関数を含む C++ クラスに対する正しくないメッセージ

スタティック解析機能を使用するためには、インテル® Inspector XE も必要です。

プログラムで仮想関数を含む C++ クラスが使用されている場合に、スタティック解析は正しくない診断を多数出力します。場合によっては、診断結果の数が多すぎて結果ファイルが使用できないこともあります。

このような C++ ソース構造を使用しているアプリケーションでは、次のコマンドライン・オプションを追加することで不要なメッセージを表示しないようにできます: /Qdiag-disable:12020,12040 (Windows*) または –diag–disable 12020,12040 (Linux*)。**このオプションは、スタティック解析の結果が作成されるリンク時に追加する必要があります。**コンパイル時に追加しただけでは十分な効果が得られません。Microsoft* Visual Studio* では、このオプションを [プロパティ ページ] > [Linker (リンカー)] > [Command Line (コマンドライン)] に追加します。

ビルド仕様ファイルを使用してスタティック解析を行う場合は、-disable-id 12020,12040 オプションを inspxe-runsc の呼び出しに追加します。

例:
```
inspxe-runsc -spec-file mybuildspec.spec -disable-id 12020,12040
```

この問題を含む作成済みのスタティック解析結果がある場合は、インテル® Inspector XE の GUI でそのファイルを開いて、次の手順に従って不要なメッセージを非表示にすることができます。

- 不要なメッセージは "Arg count mismatch (引数の数の不一致)" と "Arg type mismatch (引数の型の不一致)" です。それぞれの問題に対して、次の手順を実行します。
- 問題フィルターで不要な問題の種類をクリックします。これにより、それ以外の問題が非表示になります。
- 問題セットの表で任意の問題をクリックします。
- Ctrl+A キーを押すとすべての問題を選択できます。
- 右クリックしてポップアップ・メニューから [Change State (ステートの変更)] > [Not a problem (問題なし)] を選択し、不要なすべての問題のステートを設定します。
- 問題の種類のフィルターを [All (すべて)] に戻します。
- 他の不要な問題の種類に対して、上記の手順を行います。

- [Investigated/Not investigated (調査済み/未調査)] フィルターを [Not investigated (未調査)] に設定します。このフィルターは最後のほうにあるため、フィルターペインを下にスクロールしないと見えないことがあります。[Not a problem (問題なし)] ステートは [Not investigated (未調査)] と見なされるため、これで不要なメッセージが非表示になります。

4 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インテル® Debugger Extension

このセクションでは、インテル® Debugger Extension の変更点、新機能、カスタマイズ、および既知の問題をまとめています。インテル® Debugger Extension は、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コードのみサポートします。

4.1.1 機能

- オフロード拡張を使用して、コプロセッサーのネイティブ・アプリケーションとホスト・アプリケーションの両方をサポート
- 同時に複数のコプロセッサー・カードをデバッグ (オフロード拡張を使用)

4.1.2 インテル® Debugger Extension の使用

インテル® Debugger Extension は Microsoft* Visual Studio* IDE のプラグインです。Microsoft* Visual Studio* IDE で定義されたプロジェクトのデバッグを可能にします。インテル® Xeon Phi™ コプロセッサー向けアプリケーションは、ロードして実行することも、アタッチすることもできます。

インテル® Debugger Extension の使用方法は、「ドキュメント」を参照してください。

4.1.3 ドキュメント

インテル® デバッガーのドキュメントは、以下の場所にあります。

```
<install-dir>\Documentation\[en_US|ja_JP]\debugger\ ↵
                                    mic\gdb_quickstart_win.pdf
```

4.1.4 既知の問題と制限事項

- オフロードセクションで条件付きブレークポイントを使用すると、デバッガーがストールすることがあります。条件付きブレークポイントをオフロードセクション内に作成した場合、条件を評価するときにデバッガーがハングアップすることがあります。この問題は現在調査中で、将来のリリースで修正される予定です。
- オフロードデバッグは Microsoft* Visual Studio* 2012 および Microsoft* Visual Studio* 2013 でのみサポートされています。
- オフロードセクションではデータ・ブレークポイントはサポートされていません。
- オフロードセクションでは [逆アセンブル] ウィンドウで開始アドレスから 1024 バイトを超える範囲にスクロールすることはできません。
- インテル® MIC アーキテクチャー・アプリケーションの例外処理はサポートされていません。
- アプリケーション実行中のブレークポイントの変更は正しく動作しません。変更されたかのように見えますが、変更が適用されません。
- インテル® MIC アーキテクチャーのネイティブ・アプリケーションの開始はサポートされていません。現在実行中のアプリケーションにアタッチすることはできます。

- Microsoft* Visual Studio* の [スレッド] ウィンドウには、スレッドの凍結、凍結解除、名前変更を行うコンテキスト・メニューがあります。これらのコンテキスト・メニューは、インテル® Xeon Phi™ コプロセッサー上のスレッドでは正しく動作しません。
- オフロードセクションの直前にブレークポイントを設定すると、オフロードセクションの最初の文にブレークポイントが設定されます。この動作は、設定したブレークポイントとオフロードセクションの間にホスト用の文がない場合のみ起こります。これは Microsoft* Visual Studio* ブレークポイントの通常の動作ですが、ホストとコプロセッサーのコードが混在表示されることがあります。オフロードセクションの不要なブレークポイントは、必要に応じて、手動で無効に (または削除) することができます。
- オフロードセクションを含むインテル® 64 対応アプリケーションのみ、インテル® メニー・インテグレーテッド・コア・アーキテクチャー向けインテル® Debugger Extension を使用してデバッグすることができます。
- オフロードセクションをステップアウトすると、ホストコードにステップバックせず、(別のイベントが発生しない限り) 停止することなく実行が継続されます。これは意図された動作です。
- 「次のステートメントの設定」機能は、オフロードセクション内では動作しません。
- ブレークポイントがプロジェクトのオフロードセクションにすでに設定されている場合、デバッガーを開始するとアドレスのない境界ブレークポイントが表示されることがありますが、動作には影響しません。
- オフロードセクションでは、アドレスまたは [逆アセンブル] ウィンドウ内でブレークポイントを設定しても動作しません。
- オフロードセクションでは、次のヒットカウンター条件を含むブレークポイントは動作しません:「ヒット カウント数が次の数と等しいときに中断」および「ヒット カウントが次の数の倍数になったときに中断」
- [逆アセンブル] ウィンドウの次のオプションはオフロードセクション内では動作しません:「行番号を表示」、「シンボル名の表示」、「ソースコードの表示」
- オフロードセクションの外部で宣言された変数を評価すると誤った値が表示されます。
- 詳細は情報は、出力 (デバッグ) ウィンドウを参照してください。実装されていない機能がリストされるか (上記を参照)、デバッグセッションの設定問題に必要な追加情報が提供されます。このウィンドウを開くには、Microsoft* Visual Studio* で [デバッグ] > [ウィンドウ] > [出力] を選択します。
- オフロードが有効なアプリケーションをデバッグする場合、[イミディエイト] ウィンドウにおいて書き込む前にそのメモリー位置を読み取る変数代入 (例えば、x=x+1) を入力すると、デバッガーはハングアップします。オフロードが有効なアプリケーションの変数値を変更するときは、[イミディエイト] ウィンドウを使用しないでください。
- インテルから提供されているデバッガー拡張の動作 (例えば、実行制御) や出力 (例えば、逆アセンブリー) は Microsoft* Visual Studio* のデバッガーと異なる場合があります。これは、それぞれ実装しているデバッグ手法が異なるためです。デバッグに大きな影響はありません。

5 インテル® IPP

このセクションでは、インテル® IPP のこのバージョンでの変更点、新機能、および最新情報をまとめています。

インテル® IPP 8.2 の最新情報は、<install dir>\Documentation\<locale>\ipp\ReleaseNotes.htm にある製品のリリースノート (英語) を参照してください。

インテル® IPP についての詳細は、次のリンクを参照してください。

- **新機能:** インテル® IPP 製品ページ (http://intel.ly/OG5IF7 (英語)) およびインテル® IPP リリースノート (http://intel.ly/1uj984p (英語)) を参照してください。
- **ドキュメント、ヘルプ、サンプル:** インテル® IPP 製品ページ (http://intel.ly/OG5IF7) のドキュメントのリンクを参照してください。

5.1 別途ダウンロード可能なインテル® IPP 暗号化ライブラリー

インテル® IPP 暗号化ライブラリーは別途ダウンロード可能です。ダウンロードとインストールの手順については、http://intel.ly/ndrGnR (英語) を参照してください。

6 インテル® MKL

このセクションでは、インテル® MKL の変更点、新機能、および最新情報をまとめています。問題の修正については、http://intel.ly/RGiGV9 (英語) を参照してください。

6.1 本バージョンでの変更

6.1.1 インテル® MKL 11.2 の新機能

- インテル® ストリーミング SIMD 拡張命令 4.1 (インテル® SSE4.1) およびインテル® ストリーミング SIMD 拡張命令 4.2 (インテル® SSE4.2) 命令セット対応のすべてのインテル® Atom™ プロセッサー向けの最適化を提供
- 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) 命令セットをサポート (BLAS、DFT、VML の最適化は制限あり)
- BLAS および LAPACK ドメインで verbose モードをサポート (インテル® MKL 関数呼び出しの入力引数をキャプチャー可能)
- インテル® MPI ライブラリー 5.0 をサポート
- インテル® MKL を使用して特定の複雑な問題を解く方法を説明する新しいドキュメント、インテル® MKL クックブックを提供
- すべてのプロセッサーにおいて小行列の ?GEMM パフォーマンスを向上する MKL_DIRECT_CALL または MKL_DIRECT_CALL_SEQ コンパイル機能を追加 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照)
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーにおいて、シングル・ダイナミック・ライブラリー (mkl_rt) をリンクする機能を追加
- カスタマイズ可能なエラーハンドラーを追加。詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) リファレンス・マニュアル』の「mkl_set_exit_handler() 」の説明を参照
- リソース共有メカニズムによりインテル® Xeon Phi™ コプロセッサーの自動オフロード機能を拡張 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) リファレンス・マニュアル』の mkl_mic_set_resource_limit() 関数および MKL_MIC_RESOURCE_LIMIT 環境変数の説明を参照)
- クラスター用並列直接法スパースソルバー:
    - インテル® MKL PARDISO 直接法スパースソルバーの分散メモリーバージョンである、クラスター用並列直接法スパースソルバーを追加
    - 分散行列の行列集約ステップのパフォーマンスが向上

    - 複数の因数分解ステップにおける並べ替え情報の再利用が可能に
    - 分散 CSR 形式、分散行列、RHS、分散ソリューションのサポートを追加
    - 複数の右辺が含まれる式の解の算出をサポート
    - 因数分解および解の算出ステップのクラスターサポートを追加
    - ピュア MPI モードのサポートおよびハイブリッド構成での単一 OpenMP* スレッドのサポートを追加
- BLAS:
    - 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) 対応の 64 ビット・プロセッサーにおいて ?GEMM のスレッド・パフォーマンスが向上
    - インテル® AVX-512 命令セット用の ?GEMM、?TRSM、DTRMM を最適化
    - インテル® MIC アーキテクチャーにおいて、外積 [large m, large n, small k] および Tall Skinny 型行列 [large m, medium n, small k] の ?GEMM のパフォーマンスが向上
    - インテル® MIC アーキテクチャーにおいて自動オフロードモードの ?TRSM および ?SYMM のパフォーマンスが向上
    - インテル® AVX2 対応の 64 ビット・プロセッサーにおいてレベル 3 BLAS 関数のパフォーマンスが向上
    - コンパイル中に MKL_DIRECT_CALL または MKL_DIRECT_CALL_SEQ が定義されている場合、すべてのプロセッサーにおいて小行列の ?GEMM パフォーマンスが向上 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照)
    - インテル® SSE4.2、インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX)、およびインテル® AVX2 命令セット対応の 64 ビット・プロセッサーにおいて、beta=1、k=1 の場合の DGER および DGEMM のパフォーマンスが向上
    - インテル® AVX-512 命令セット用の (D/Z)AXPY を最適化
    - インテル® AVX2 およびインテル® AVX-512 命令セット用の ?COPY を最適化
    - インテル® AVX-512 命令セット用の DGEMV を最適化
    - インテル® AVX およびインテル® AVX2 対応の 64 ビット・プロセッサーにおいて SSYR2K のパフォーマンスが向上
    - すべてのインテル® プロセッサー用の ?AXPBY のスレッド・パフォーマンスが向上
    - インテル® AVX-512 において side=R、uplo={U,L}、transa=N、diag={N,U} の場合の DTRMM のパフォーマンスが向上
- LINPACK:
    - ヘテロジニアス Intel® Optimized MP LINPACK Benchmark for Clusters において行列生成のパフォーマンスが向上
    - Intel® Optimized MP LINPACK Benchmark パッケージのインテル® MIC アーキテクチャー用オフロード・オプションでインテル® AVX2 ホストをサポート
    - インテル® AVX2 対応の 64 ビット・プロセッサーにおいて Intel® Optimized MP LINPACK Benchmark for Clusters パッケージのパフォーマンスが向上
- LAPACK:
    - ?(SY/HE)RDB のパフォーマンスが向上
    - 固有ベクトルが必要な場合の ?(SY/HE)EV のパフォーマンスが向上
    - 固有ベクトルが不要な場合の ?(SY/HE)(EV/EVR/EVD) のパフォーマンスが向上
    - 劣決定 (M が N 未満) の場合の ?GELQF、?GELS および ?GELSS のパフォーマンスが向上

    - ?GEHRD、?GEEV および ?GEES のパフォーマンスが向上
    - LAPACKE インターフェイスにおいて NaN チェッカーのパフォーマンスが向上
    - ?GELSX、?GGSVP のパフォーマンスが向上
    - 固有ベクトルが不要な場合の ?(SY/HE)(EV/EVR/EVD) のパフォーマンスが向上
    - ?GETRF のパフォーマンスが向上
    - M>=N で特異ベクトルが必要ないときの (S/D)GE(SVD/SDD) のパフォーマンスが向上
    - インテル® MIC アーキテクチャーにおいて自動オフロードモードの ?POTRF UPLO=U のパフォーマンスが向上
    - インテル® MIC アーキテクチャーにおいて ?SYRDB の自動オフロードを追加、固有ベクトルが不要な場合に ?SY(EV/EVD/EVR) がスピードアップ
- PBLAS および ScaLAPACK:
    - 大規模な分散ブロッキング係数の P?GEMM ルーチンで自動オフロードが可能に
- スパース BLAS:
    - インテル® AVX-512 命令セット用の SpMV カーネルを最適化
    - スパース BLAS で対角形式を使用する場合のリリースサンプルを追加
    - インテル® SSE4.2、インテル® AVX、およびインテル® AVX2 命令セット対応システムにおいてスパース BLAS レベル 2 およびレベル 3 のパフォーマンスが向上
- インテル® MKL PARDISO:
    - 任意のソルバーステージで後から使用できるようにインテル® MKL PARDISO ハンドルをディスクに格納する機能を追加
    - 非対称行列およびアウトオブコア・モードにピボット制御のサポートを追加
    - 非対称行列およびアウトオブコア・モードに対角抽出のサポートを追加
    - 非線型方程式の反復ソルバーとしてインテル® MKL PARDISO を使用するサンプルを追加
    - 反復改善が無効な場合、因数分解ステージ後にオリジナル行列で割り当てたメモリーを解放する機能を追加
    - 並べ替えアルゴリズムのアウトオブコア (OOC) 部分サイズのメモリー推定向上により、OOC モードの因数分解ステップのパフォーマンスが向上
    - インテル® MKL PARDISO の出力メッセージを変更
    - 構造対称の因数分解中のゼロピボットをサポート
- ポアソン・ライブラリー:
    - 線形方程式を解く前提条件としてインテル® MKL ポアソン・ライブラリーを使用するサンプルを追加
- 拡張固有値ソルバー:
    - 出力メッセージを変更
    - サンプルを変更
    - スパース問題を解くための事前定義インターフェイスに入力および出力 iparm パラメーターを追加
- FFT:
    - インテル® AVX-512 命令セット用の FFT を最適化
    - インテル® MIC アーキテクチャーにおいて 2 のべき乗でない長さのパフォーマンスが向上
- VML: 各ベクトル要素の小数部を計算する v[d|s]Frac 関数を追加

- VSL RNG:
  - 二項乱数ジェネレーターで ntrial=0 をサポート
  - インテル® MIC アーキテクチャーにおいて MRG32K3A および MT2203 BRNG のパフォーマンスが向上
  - インテル® AVX およびインテル® AVX2 命令セット対応のプロセッサーにおいて MT2203 BRNG のパフォーマンスが向上
- VSL サマリー統計:
  - グループ化された/プールされた平均推定 (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) をサポート
- データ・フィッティング: ブレークポイント数が 2 または 3 の場合の自然 3 次スプライン構築関数の不正な動作を修正
- インテル® MKL 環境変数で指定したすべての設定を無視するインテル® MKL モードを追加
  - mkl_set_env_mode() ルーチン (インテル® MKL 固有のすべての環境設定を無視するようにインテル® MKL に指示) を呼び出してモードをセットアップすると、MKL_NUM_THREADS、MKL_DYNAMIC、MKL_MIC_ENABLE その他のすべてのインテル® MKL 関連の環境変数が無視される; 必要な引数は mkl_set_num_threads() や mkl_mic_enable() などのインテル® MKL サービスルーチンから設定可能

既知の制限事項:

- Windows* で大規模行列を自動オフロードするとデータ破損が発生したりクラッシュすることがあります。COI に問題があります: HSD4868293 (クリティカル)。COI は Windows* で 4GB を超えるバッファーおよび 2M ページを割り当てることができません。

  回避方法: MKL_MIC_MAX_MEMORY=3G に設定します。

  注: この問題はインテル® MPSS 3.3 で解決されます。

注: API シンボル、引数の順序、リンク行はインテル® MKL 11.2 Beta Update 2 で変更されました。詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照してください。

注: 廃止予定の項目は、インテル® MKL 11.2 で廃止予定の項目 (英語) を参照してください。

6.2 権利の帰属

エンド・ユーザー・ソフトウェア使用許諾契約書 (End User License Agreement) で言及されているように、製品のドキュメントおよび Web サイトの両方で完全なインテル製品名の表示 (例えば、"インテル® マス・カーネル・ライブラリー") とインテル® MKL ホームページ (http://www.intel.com/software/products/mkl (英語)) へのリンク/URL の提供を正確に行うことが最低限必要です。

インテル® MKL の一部の基となった BLAS の原版は http://www.netlib.org/blas/index.html (英語) から、LAPACK の原版は http://www.netlib.org/lapack/index.html (英語) から入手できます。LAPACK の開発は、E. Anderson、Z. Bai、C. Bischof、S. Blackford、J. Demmel、J. Dongarra、J. Du Croz、A. Greenbaum、S. Hammarling、A. McKenney、D. Sorensen らによって行われました。LAPACK 用 FORTRAN 90/95 インターフェイスは、http://www.netlib.org/lapack95/index.html (英語) にある LAPACK95 パッケージと類似しています。すべてのインターフェイスは、純粋なプロシージャー用に提供されています。

インテル® MKL クラスター・エディションの一部の基となった ScaLAPACK の原版は http://www.netlib.org/scalapack/index.html (英語) から入手できます。ScaLAPACK の開発は、L. S. Blackford、J. Choi、A. Cleary、E. D'Azevedo、J. Demmel、I. Dhillon、J. Dongarra、S. Hammarling、G. Henry、A. Petitet、K. Stanley、D. Walker、R. C. Whaley らによって行われました。

インテル® MKL Extended Eigensolver の機能は、Feast Eigenvalue Solver 2.0 (http://www.ecs.umass.edu/~polizzi/feast/) をベースにしています。

インテル® MKL の PARDISO は、バーゼル大学 (University of Basel) から無償で提供されている PARDISO 3.2(http://www.pardiso-project.org (英語)) と互換性があります。

本リリースのインテル® MKL の一部の FFT 関数は、カーネギーメロン大学からライセンスを受けて、SPIRAL ソフトウェア生成システム (http://www.spiral.net/ (英語)) によって生成されました。SPIRAL の開発は、Markus Püschel、José Moura、Jeremy Johnson、David Padua、Manuela Veloso、Bryan Singer、Jianxin Xiong、Franz Franchetti、Aca Gacic、Yevgen Voronenko、Kang Chen、Robert W. Johnson、Nick Rizzolo らによって行われました。

7 インテル® TBB 4.3

インテル® TBB の変更に関する詳細は、インテル® TBB ドキュメント・ディレクトリー (`<installdir>\Composer XE 2015\Documentation\<locale>\tbb`) の CHANGES というファイルを参照してください。

7.1 既知の問題

インテル® TBB の本リリースに関する次の注意事項に留意してください。

7.1.1 ライブラリーの問題

- 同じプログラムで連続してインテル® TBB と OpenMP* コンストラクトをともに使用していて、OpenMP* コードにインテル® コンパイラーを使用している場合、KMP_BLOCKTIME に小さな値 (例えば、20 ミリ秒) を設定するとパフォーマンスが向上します。この設定は、kmp_set_blocktime() ライブラリー呼び出しを使用して OpenMP* コード内で行うこともできます。KMP_BLOCKTIME および kmp_set_blocktime() の詳細は、コンパイラーの OpenMP* に関するドキュメントを参照してください。
- 一般に、アプリケーションやサンプルの非デバッグ ("リリース") ビルドは、インテル® TBB ライブラリーの非デバッグバージョンとリンクし、デバッグビルドはインテル® TBB ライブラリーのデバッグバージョンとリンクします。Windows* システムでは、/MD オプションを使用してコンパイルした場合はインテル® TBB ライブラリーのリリース・ライブラリー、/MDd オプションを使用してコンパイルした場合はデバッグ・ライブラリーを使ってビルドしてください。他の組み合わせでは、ランタイムエラーが発生します。デバッグ・ライブラリーとリリース・ライブラリーの詳細については、製品の "Documentation" サブディレクトリーに含まれているチュートリアルを参照してください。

8　著作権と商標について

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、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

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) 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、インテル® IPP、インテル® MKL、およびインテル® TBB は、インテルのエンド・ユーザー・ソフトウェア使用許諾契約書 (EULA) の下で提供されます。

GNU* プロジェクト・デバッガー (GDB) は、General GNU Public License GPL V3 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、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。