インテル® C++ Composer XE 2011 Windows* 版インストール・ガイドおよびリリースノート

資料番号: 321414-003JA

2012 年 4 月 11 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。

インテル® C++ Composer XE 2011 は、以前「インテル® C++ コンパイラー・プロフェッショナル・エディション」と呼ばれていた製品の最新バージョンです。

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。

Update 10 (2011.10)

- インテル® C++ コンパイラー XE 12.1.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- 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 7.0 Update 7
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 4
- 報告された問題の修正

Update 9 (2011.9)

- インテル® C++ コンパイラー XE 12.1.3
- インテル® マス・カーネル・ライブラリー 10.3 Update 9
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 3

- 報告された問題の修正

Update 8 (2011.8)

- インテル® C++ コンパイラー XE 12.1.2
- インテル® マス・カーネル・ライブラリー 10.3 Update 8
- 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 7.0 Update 6
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 2
- 報告された問題の修正

Update 7 (2011.7)

- インテル® C++ コンパイラー XE 12.1.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 7
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 1
- インライン・アセンブリーと組込み関数でのインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX 2) のサポート
- 報告された問題の修正

Update 6 (2011.6)

- 新しいトップレベル・フォルダーへの製品のインストール
- クラスターでのインストールのサポート
- Microsoft* Visual Studio* 2005 のサポート終了予定
- Microsoft* Windows Server* 2003 および Microsoft* Windows Vista* のサポート終了予定
- インテル® Parallel Debugger Extension のサポート終了予定
- IDE 統合のアップデート
- インテル® C++ コンパイラー XE 12.1
    - 追加の C++0x 機能のサポート
    - コンパイラー・オプションの追加
    - インテル® Cilk™ Plus サポートの拡張
    - インテル® Cilk™ Plus の配列表記 (アレイ・ノーテーション) セマンティクスの変更
    - OpenMP* サポートの拡張
    - コンパイラーの主要ドキュメントであるユーザー・リファレンス・ガイドの簡略化と再編成。主な変更点: インテル® コンパイラーの主要機能をまとめた新しいセクション「主な機能」の追加と、コンパイラー・オプション・リファレンスの機能別のグループ化。
- インテル® デバッガー 12.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 6
- 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 7.0 Update 5
- インテル® スレッディング・ビルディング・ブロック 4.0
- 報告された問題の修正

Update 5 (2011.5)

- インテル® マス・カーネル・ライブラリー 10.3 Update 5
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 8
- 報告された問題の修正

Update 4 (2011.4)

- インテル® マス・カーネル・ライブラリー 10.3 Update 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 7.0 Update 4
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 7
- 報告された問題の修正

Update 3 (2011.3)

- インテル® マス・カーネル・ライブラリー 10.3 Update 3
- 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 7.0 Update 3
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 6
- /Qsox オプションの拡張
- 報告された問題の修正

Update 2 (2011.2)

- インテル® マス・カーネル・ライブラリー 10.3 Update 2
- 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 7.0 Update 2
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 5
- immintrin.h の 3 つの組込み関数の変更
- “ginspxe-runsc.exe” ユーティリティーの変更
- 報告された問題の修正

Update 1 (2011.1)

- インテル® マス・カーネル・ライブラリー 10.3 Update 1
- 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 7.0 Update 1
- 報告された問題の修正

製品リリース (2011.0)

- 最初の製品リリース

1.2 製品の内容

インテル® C++ Composer XE 2011 Update 10 Windows* 版には、次のコンポーネントが含まれています。

- インテル® C++ コンパイラー XE 12.1.4。Windows* オペレーティング・システムを実行する IA-32 およびインテル® 64 アーキテクチャー・システムで動作するアプリケーションをビルドします。
- 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 7.0 Update 7
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- インテル® スレッディング・ビルディング・ブロック 4.0 Update 4
- インテル® Parallel Debugger Extension
- 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 1GB (2GB 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)
- Microsoft* Windows* XP SP3、Microsoft* Windows Vista* (サポート終了予定)、Microsoft* Windows* 7、Microsoft* Windows Server* 2003 (サポート終了予定)、Microsoft* Windows

Server* 2008、Microsoft* Windows HPC Server* 2008 (エンベデッド・エディションはサポートされていません)
    - Microsoft* Windows Server* 2008 または Windows HPC Server* 2008 では Microsoft* Visual Studio* 2008 SP1 が必要です。下記にリストされている Visual Studio* のその他のバージョンは Windows Server* 2008 または Windows HPC Server* 2008 ではサポートされていません。
- 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* 2010 Standard 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) [1]
    - Microsoft* Visual Studio* 2008 Standard 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) [1]
    - Microsoft* Visual Studio* 2005 Standard 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) (サポート終了予定) [1]
- 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++* 2010 Express Edition
    - Microsoft* Visual C++* 2008 Express Edition
- インテル® 64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合:
    - Microsoft* Windows* Software Development Kit (SDK) for Windows* 7 and .NET Framework 4
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

説明:

1. Microsoft* Visual Studio* 2005/2008 Standard Edition では、[x64 コンパイラおよびツール] コンポーネントがデフォルトでインストールされます。Professional 以上のエディションでは、[カスタム] インストールが必要です。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® インテグレーテッド・パフォーマンス・プリミティブまたはインテル® スレッディング・ビルディング・ブロック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。

1.3.1 IA-64 アーキテクチャー (インテル® Itanium®) 開発の未サポート

本バージョンでは、IA-64 アーキテクチャー (インテル® Itanium®) システム上、または IA-64 アーキテクチャー・システム向けの開発をサポートしていません。インテル® コンパイラー 11.1 ではまだサポートされています。

1.3.2 Windows Server* 2003 および Windows Vista* のサポート終了予定

インテル® C++ Composer XE の将来のメジャーリリースでは、Windows Server* 2003 および Windows Vista* はサポートされなくなる予定です。これらのオペレーティング・システムを使用している場合は、インテルでは新しいバージョンへの移行を推奨しています。

1.3.3 Visual Studio* 2005 のサポート終了予定

インテル® C++ Composer XE の将来のメジャーリリースでは、Visual Studio* 2005 はサポートされなくなる予定です。Visual Studio* 2005 を使用している場合は、インテルでは新しいバージョンへの移行を推奨しています。

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 Visual Studio* 2008 のインテル® Composer XE ヘルプに最初にアクセスしたときの遅延

Microsoft* Visual Studio* 2008 にインストールされたヘルプ・ドキュメントに最初にアクセスしたとき、表示に時間がかかる場合があります。これは、Visual Studio* でヘルプを表示する前に、新しいヘルプをコレクションに統合して、コレクションの索引を再生成するためです。Visual Studio* にすでに統合されているヘルプとインストールするヘルプのサイズに応じて、新しくヘルプが表示されるまで数分以上かかる場合があります。

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、日本語ユーザー向けのサポートを提供しています。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。

日本語サポートはすべての製品アップデートで提供されているわけではありません。

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

インストール時にコンパイラー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 インストール前の準備

2.1.1 64 ビット・アプリケーション用の Visual Studio* の設定

Microsoft* Visual Studio* 2005 または 2008 を使用し、64 ビット・アプリケーション (インテル® 64 アーキテクチャー向け) を開発する場合は、Visual Studio* の構成を変更して、64 ビット・サポートを追加します。

Visual Studio* 2005/2008 Standard Edition または Visual Studio* 2010 Professional Edition 以上を使用する場合は、インテル® 64 対応アプリケーションのビルド用に構成を変更する必要はありません。その他のエディションの場合は、次の操作を行ってください。

1. [コントロール パネル] の [プログラムの追加と削除] から [Microsoft Visual Studio 2005 (または 2008)] を選択し、[変更と削除] をクリックします。[Visual Studio メンテナンス モード] ウィンドウが表示されます。[次へ] をクリックします。
2. [機能の追加と削除] をクリックします。
3. [選択した機能をインストールします] で [言語ツール] の [Visual C++] を展開します。
4. [x64 コンパイラおよびツール] ボックスがオンになっていない場合は、オンにし、[更新] をクリックします。ボックスがオンの場合は、[キャンセル] をクリックします。

Visual C++* Express Edition では 64 ビットの開発はサポートされていません。

2.1.2 Microsoft* Windows Vista* および Microsoft* Windows* 7 でのインストール

Microsoft* Visual Studio* 2005 ユーザーは、Visual Studio* 2005 Service Pack 1 (VS 2005 SP1) と Visual Studio* 2005 Service Pack 1 Update for Windows Vista* (VS 2005 SP1 ページからリンクを提供) をインストールしてください。これらのアップデートをインストールした後に、管理者権限で Visual Studio* が実行できることを確認してください。実行できない場合、インテル® コンパイラーを使用できません。詳細は、Microsoft* の「Visual Studio* on Windows Vista*」ページ (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx (英語)) および関連ドキュメントを参照してください。

2.2 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。

DVD で製品を受け取った場合、製品 DVD を DVD ドライブに挿入します。自動でインストールが開始されます。自動で開始されない場合は、Windows* エクスプローラで DVD ドライブのトップレベル・ディレクトリーを開き、setup.exe をダブルクリックします。

製品のダウンロード版を購入した場合は、ダウンロードしたファイル (.EXE) をダブルクリックして、インストールを開始します。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。

2.2.1 インテル® C++ コンパイラー 11.1 プロフェッショナル・エディションのライセンスまたはシリアル番号によるインストール

インテル® C++ コンパイラー 11.1 プロフェッショナル・エディションのライセンスおよびシリアル番号は、インテル® C++ Composer XE 2011 では使用できません。製品のサポートステータスがアクティブな場合、以下の手順に従って、新しいアップグレード・ライセンスおよびシリアル番号を無料で取得できます。

1. インテル® ソフトウェア開発製品レジストレーション・センターの [登録ユーザーのログイン] セクションで、[ログイン ID] と [パスワード] を入力してサイトにログインします。サポートサービスが有効なすべての製品のリストが表示されます。
2. 旧名称の製品とともに、XE 製品の名前も表示されます。[最新版アップデートのダウンロード] 列で最新のアップデートのリンクをクリックすると、製品のアップグレード・ページが表示されます。アップグレードする製品名をクリックしてください。
3. アップデートされたライセンスファイルを登録アドレスへメールで送信するか、表示されるシリアル番号を使用して、インテル® C++ Composer XE 2011 製品をインストールすることができます。

2.2.2 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。

2.2.3 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。

2.2.4 インテルのアクティベーション・ツールを使用した製品のアクティベーション

この製品リリースでは、新しいインテルのアクティベーション・ツール
“ActivationTool.exe” が “[Common Files]\Intel\Parallel Studio XE 2011\Activation\” にインストールされます。

インストール中に評価用ライセンスまたは評価用シリアル番号を使用したり、あるいは [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製品をインストールした場合、製品を購入した後にこのアクティベーション・ツール ([スタート] > [すべてのプログラム] > [インテル(R) Parallel Studio XE 2011] > [Product Activation (製品のアクティベーション)]) を使

用して製品をアクティベートできます。これにより、評価版から製品版へ移行することができます。

2.2.5 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。

2.2.6 Microsoft* Visual Studio* 2005 の管理者権限に関するメッセージダイアログ

Microsoft* Visual Studio* 2005 を使用している場合、Microsoft* Windows Vista* またはそれ以降の Microsoft* Windows* にインストール中、次のようなダイアログが表示されることがあります。

このダイアログが表示された場合は、[常にこのメッセージを使用する] ボックスをオンのままにして、[続行] ボタンをクリックします。[Visual Studio の終了] を選択したり、何もしない場合 (このメッセージは 2 分後にタイムアウトします)、コンパイラー統合のインストールは完了しません。

詳細は、「Microsoft* Windows Vista* でのインストール」を参照してください。

2.3 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。

2.4 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 2011 SP1`
  - `bin`
    - `ia32`
    - `ia32_intel64`

        - intel64
    - compiler
        - include
            - ia32
            - intel64
        - lib
            - ia32
            - intel64
    - debugger
    - Documentation
    - Help
    - ipp
        - bin
        - demo
        - include
        - interfaces
        - lib
        - tools
    - mkl
        - benchmarks
        - bin
        - examples
        - include
        - interfaces
        - lib
        - tests
        - tools
    - tbb
        - bin
        - examples
        - include
        - lib
    - redist
    - Samples
    - VS Integration

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 上での実行用のコンパイラー。インテル® 64 上で動作するアプリケーションをビルドします。

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 2011 SP1.*nnn* (*nnn* はアップデート番号) に変更されます。

2.5 インストールの既知の問題

2.5.1 複数の Visual Studio* のバージョンを使用している場合のドキュメントに関する問題

システムに Microsoft* Visual Studio* 2005 と 2008 の両方をインストールしていてインテル® C++ Composer XE 2011 を両方のバージョンに統合した場合、いずれかのバージョンを削除すると両方のバージョンから統合されていたインテル® C++ Composer XE 2011 ドキュメントが削除されます。

ドキュメントを再インストールするには、以下の操作を行います。

1. コントロール パネルを使用して製品を選択します。
    - Windows* XP の場合: **[コントロール パネル] - [プログラムの追加と削除]** を選択します。
    - Windows* 7 の場合: **[コントロール パネル] - [プログラムと機能]** を選択します。
    - Windows Vista の場合: **[コントロール パネル] - [プログラム]** を選択します。
2. 製品を選択した後、**[変更と削除]** ボタンをクリックします。
3. **[コンポーネントの選択]** ダイアログボックスで、[統合ドキュメント] の選択を解除します。ドキュメントが**削除**されます。
4. ステップ 1 と 2 を繰り返します。
5. **[コンポーネントの選択]** ダイアログボックスで、[統合ドキュメント] を選択します。ドキュメントが**再インストール**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。詳細は、下記を参照してください。

3.2 新機能と変更された機能

インテル® C++ Composer XE 2011 Update 7 には、インテル® C++ コンパイラー XE 12.1 が含まれています。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、Visual Studio* に統合されたドキュメントを参照してください。

- C++0x からの機能
    - 可変個引数テンプレート
    - char16_t および char32_t データ型
    - _Pragma
    - 関数の削除
    - メンバー関数のデフォルト
    - sizeof、typeid 、decltype によるクラスの非スタティック・データ・メンバーの直接参照
    - ヌルポインター
    - 二重角括弧 ([[ ]]) で囲まれた属性
    - エイリアスとエイリアス・テンプレートの宣言
    - 戻り型を持つ新しい関数宣言構文 (例: `auto f()->int;`)

- OpenMP* 3.1
  - final 節
  - mergeable 節
  - taskyield 宣言子
  - 新しい atomic 節
- スタティック・セキュリティー解析の IDE 統合の向上
- インテル® Cilk™ Plus の拡張:
  - 新しい holder ハイパーオブジェクト

インテル® C++ コンパイラー XE 12.0 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、Visual Studio* に統合されたドキュメントを参照してください。

- インテル® Cilk™ Plus。インテル® C++ コンパイラー向けのこの言語拡張を使用することで、新規および既存のソフトウェアを簡単に並列化できます。
- ガイド付き自動並列化
- C++0x からの機能
  - 右辺値参照
  - 標準的なアトミック演算
  - "gWindows* C++" モードでの C99 の 16 進浮動小数点定数のサポート
  - 右角括弧
  - 拡張 friend 宣言
  - 混在した文字列リテラルの結合
  - `long long` のサポート
  - 可変引数マクロ
  - スタティック・アサーション
  - auto 型変数
  - extern テンプレート
  - `__func__` 事前定義済み指定子
  - 式の型宣言 (decltype)
  - ユニバーサル文字名
  - 強い型付けの列挙型
  - ラムダ
- より高速でやや精度が低い算術ライブラリー関数を使用するためのオプション
- プロセッサーのモデルや製造元に関係なく一貫した結果を返す算術ライブラリー関数を使用するためのオプション

3.2.1 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® AVX 2) 命令のサポート (Update 7)

インテル® C++ Composer XE 2011 Update 7 のコンパイラーは、インライン・アセンブリーと 組込み関数 (immintrin.h) でインテル® AVX 2 命令をサポートします。

3.2.2 インテル® Cilk™ Plus の配列表記 (アレイ・ノーテーション) セマンティクスの変更 (Update 6)

インテル® C++ Composer XE 2011 では、次のようなインテル® Cilk™ Plus の部分配列の代入は、結果の一時コピーが生成されるためパフォーマンスに影響します。

```
a[:]= b[:]+ c[:];
```

インテル® C++ Composer XE 2011 Update 6 から、代入式の右辺の部分配列 (上記の例では b[:] や c[:]) の一部が左辺の部分配列 (上記の例では a[:]) とメモリー上でオーバーラップする場合、そのような代入の結果は不定となります。意図する動作が得られるように、代入式でメモリー上の部分的なオーバーラップが発生しないようにするのはプログラマーの責任です。

ただし、次のように、部分配列が完全にオーバーラップする場合は例外です。

```
a[:]= a[:]+ 3;
```

この場合、配列が完全にオーバーラップするため、意図したとおりに動作し、期待どおりの結果が得られます。

3.2.3 インテル® Cilk™ Plus の “scalar” 節のサポート終了予定

インテル® Cilk™ Plus の要素関数のオプションで使用可能な “scalar” 節は、将来のバージョンのインテル® C++ Composer XE では削除されます。代わりに、機能的に同じ “uniform” 節を使用してください。

3.2.4 /Qsox オプションの追加キーワード、デフォルトの変更 (Update 3)

オブジェクト・ファイルおよび実行ファイルに使用されたコンパイラー・オプションとプロシージャーのプロファイル情報を追加するための /Qsox オプションは、インライン展開された関数のリストを含めたり、プロシージャーのプロファイル情報を除外できるようになりました。

/Qsox の構文は次のように変更されました。

```
/Qsox[-]
/Qsox=keyword[,keyword]
```

keyword には、inline または profile のいずれかを指定できます。キーワードなしで /Qsox を指定すると、以前のリリースとは異なり、コマンドライン・オプションの情報のみが追加されます。以前のリリースと同じ動作にするには、/Qsox=profile を使用してください。/Qsox オプションはコマンドラインで複数回指定することができますが、その場合は左から右の順に解釈されます。

この情報は、オブジェクト・ファイルにコメントとして追加されます。Visual Studio* 2005 以降の Microsoft* のリンカーではこれらのコメントは無視されるため、実行ファイルにはこの情報は含まれません。

3.2.5 3 つの組込み関数の変更 (Update 2)

3 つの組込み関数 (_rdrand16_step()、_rdrand32_step()、_rdrand64_step()) は、Update 2 で変更されました。この変更は、ドキュメントにはまだ反映されていません。これらの組込み関数は、“immintrin.h” ヘッダーファイルで宣言されており、ハードウェアにより生成される乱数値を返します。

3 つとも同じ RDRAND 命令にマップされ、16/32/64 ビットの乱数整数を生成します。

構文

1. extern int _rdrand16_step(unsigned short *random_val);
2. extern int _rdrand32_step(unsigned int *random_val);
3. extern int _rdrand64_step(unsigned __int64 *random_val);

説明

これらの組込み関数は、RDRAND 命令を使用して、ハードウェアにより生成される乱数値の取得を 1 回試みます。生成された乱数値は指定されたメモリー位置に書き込まれ、成功ステータスが返されます。ハードウェアにより有効な乱数値が返された場合は 1 を返し、そうでない場合は 0 を返します。

戻り値

ハードウェアにより生成された 16/32/64 乱数値。

制限事項

_rdrand64_step() 組込み関数は、64 ビット・レジスター対応のシステムでのみ使用できます。

3.2.6 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: `-diag-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。

3.2.6.1 Update 2 からの "inspxe-runsc" コマンドライン・ユーティリティーの変更

インテル® Composer XE 2011 に含まれるこのユーティリティーは、Update 2 から変更されています。この変更は、インテル® Composer XE 2011 を使用してスタティック解析を実行する場合にのみ影響します。スタティック解析を使用しない場合や、このユーティリティーを使用せずにスタティック解析を実行する場合には影響ありません。スタティック解析はインテル® Parallel Studio XE 2011、インテル® Fortran Studio XE 2011 またはインテル® C++ Studio XE 2011 でのみ利用できます。そのため、これらの製品をお使いでない場合は影響ありません。

inspxe-runsc は、アプリケーションのビルド方法を示す**ビルド仕様**を実行します。通常、ビルド仕様ファイルは、ビルドを実行して、実際に行われたコンパイルとリンクを記録することにより生成されます。inspxe-runsc は、インテル® コンパイラーをスタティック解析モードで使用して、再度この処理を行います。スタティック解析結果はリンクステップで生成されるため、inspxe-runsc で複数のリンクステップを持つビルドが含まれるビルド仕様を実行すると、複数のスタティック解析結果が生成されます。

インテル® Composer XE 2011 およびインテル® Composer XE 2011 Update 1 の inspxe-runsc は、すべてのスタティック解析結果を同じディレクトリーに生成します。リンクが複数ある場合、これは、1 つのプロジェクトのスタティック解析結果は同じディレクトリーに 1 つだけでなければならないという規則に違反します。新しいバージョンの inspxe-runsc は、リンクステップごとの結果を個別のディレクトリーに生成することで、この規則に従っています。ディレクトリー名は、リンクされるファイルの名前を基に付けられます。2 つの実行ファイル file1.exe と file2.exe をビルドするプロジェクトのビルド仕様の場合、以前のバージョンの inspxe-runsc では、file1 の結果と file2 の結果 (例えば r000sc と r001sc) が同じディレクトリーに作成されます。新しいバージョンの inspxe-runsc でも結果は 2 つ作成されますが、file1 の結果は "My Inspector XE results - file1/r000sc"、file2 の結果は "My Inspector XE results - file2/r000sc" というように別々のディレクトリーに作成されます。2 つの結果のディレクトリーは同じ親ディレクトリーの下に作成されます。

inspxe-runsc には、結果の作成場所を指定するための `-result-dir` (`-r`) コマンドライン・スイッチがあります。このスイッチの動作が変更されました。以前は、r000sc のように結果が作成されるディレクトリーの名前を指定していましたが、現在は、"My Inspector XE Results - name" のように結果が作成されるディレクトリーの親ディレクトリーを指定します。つまり、`-r` スイッチのディレクトリー名は、結果の生成される場所から 2 つ上のディレクトリーのものになります。

inspxe-runsc のこの変更により、結果ディレクトリーが効率良く移動します。この変更に伴い、ユーザーによる対応が必要になります。`-r` スイッチを指定して inspxe-runsc を呼び出

すスクリプトを使用している場合は、新しい動作に合わせて、-r スイッチの引数を変更してください。また、新しいバージョンの inspxe-runsc によって生成されるスタティック解析結果が、以前のバージョンの inspxe-runsc によって生成された結果と同じディレクトリーに保存されることがないように、以前の結果ファイルを新しいディレクトリーに移動する必要があります。以前のバージョンの inspxe-runsc でリンクステップが 1 つのみのビルド仕様を実行した結果は、"My Inspector XE results - name" という形式のディレクトリーに移動します。この操作を行わないと、新しく作成される結果ですべての問題が "新規" として表示されます。以前のバージョンの inspxe-runsc で複数のリンクステップを含むビルド仕様を実行した場合、スタティック解析ではさまざまな問題がありましたが、これらの問題は新しいバージョンを使用することで解決されます。この場合、以前の結果のうち最も新しいものを "My Inspector XE results - name" という形式の新しいディレクトリーに (1 つのディレクトリーに 1 つの結果が含まれるように) コピーします。これにより、新しいバージョンで作成される結果に以前の問題ステート情報が正しく適用される可能性が高くなります。

3.2.7 Microsoft* Visual Studio* 2010 のサポート

本リリースでは、Microsoft* Visual Studio* 2010 をサポートしています。Visual Studio* 2010 のインテル® C++ 統合は、Visual Studio* 2005 または 2008 の統合とは大きく異なります。詳細は次の記事を参照してください: http://intel.ly/o8lHSk (英語)

3.2.8 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 12.1 (インテル® Composer XE 2011 Update 6) で変更されました。インテル® C++ の古いバージョン (インテル® Composer XE 2011 の以前のアップデートも含む) 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 12.1 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 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、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.3.1 インテル® C++ Composer XE 2011 Update 6 の新規および変更されたコンパイラー・オプション

- /arch:CORE-AVX2
- /QxCORE-AVX2
- /QaxCORE-AVX2
- /arch:CORE-AVX-I
- /QxCORE-AVX-I
- /QaxCORE-AVX-I
- /QxSSSE3_ATOM
- /Qopt-mem-layout-trans
- /openmp
- /Qparallel-source-info
- /Qsox
- /Qvla

3.3.2 インテル® C++ Composer XE 2011 の新規および変更されたコンパイラー・オプション

- /Qansi-alias-check
- /Qcilk-serialize
- /Qdiag-sc-dir

- /Qfp-trap
- /Qfp-trap-all:*mode[,mode, ...]*
- /Qguide
- /Qguide-data-trans
- /Qguide-file
- /Qguide-file-append
- /Qguide-opts:*string*
- /Qguide-par
- /Qguide-vec
- /Qimf-absolute-error
- /Qimf-accuracy-bits
- /Qimf-arch-consistency
- /Qimf-max-error
- /Qimf-precision
- /Qintel-extensions
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpatchable-addresses
- /Qprof-value-profiling
- /Qprofile-functions
- /Qprofile-loops
- /Qprofile-loops-report
- /Qpar-runtime-control[n]
- /Qpar-runtime-control-
- /Qregcall
- /Qsimd[-]
- /Qzero-initialized-in-bss

廃止予定のコンパイラー・オプションのリストは、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.3.3 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 2011] > [Command Prompt (コマンドプロンプト)] > [Parallel Studio XE with Intel Compiler XE 2011 v12.1 [Update xx] (インテル(R) コンパイラー XE 12.0 [Update xx])] > [IA-32 Visual Studio xxx mode (IA-32 Visual Studio xxx 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 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`

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。

- `vs2010`
- `vs2008`
- `vs2005`

また、インテル® Visual Fortran Composer XE 2011 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 Composer XE 2011 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。

3.4.2 OpenMP* レガシー・ライブラリーの削除

本リリースでは、OpenMP* のレガシー・ライブラリーが削除されました。"互換性がある" ライブラリーのみ提供されます。

3.4.3 OpenMP* ライブラリーのデフォルトがダイナミック・リンクに変更

バージョン 11.0 より、デフォルトで OpenMP* アプリケーションはダイナミック OpenMP* ライブラリーにリンクされます。OpenMP* ライブラリーのスタティック・リンクを指定するには、`/Qopenmp-link:static` を指定します。スタティック・ライブラリーは廃止され、将来のリリースでは削除される可能性があります。

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/plmnpo (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 日本語ファイル名に関するコマンドライン診断表示の問題

コンパイル診断で日本語が含まれているファイル名は、ネイティブのインテル® 64 対応アプリケーション用コンパイラーを使用して、Windows* コマンドでコンパイルした場合に正しく表示されません。Visual Studio* を使用する場合やインテル® 64 対応アプリケーション用クロスコンパイラーまたは IA-32 対応アプリケーション用コンパイラーを使用する場合は、この問題は発生しません。

3.5.2 Visual Studio* の既知の問題

3.5.2.1 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。

3.5.2.2Microsoft Visual Studio* 2005 のカスタムビルド規則*

インテル® C++ コンパイラーの Microsoft* Visual Studio* 2005 への統合は、Visual C++* のカスタムビルド規則機能をサポートしています (一部の機能は制限されています)。カスタムビルド規則を使用すると、ビルド処理にカスタムツールの呼び出しを追加できます。カスタムビルド規則に関する詳細は、Visual C++* のドキュメントを参照してください。

カスタムビルド規則は、Visual C++* プロジェクトをインテル・プロジェクトに変換する前に作成する必要があります。インテル・プロジェクトに変換した後にカスタムビルド規則を変更する場合は、いったんプロジェクトを Visual C++* プロジェクト・システムに変換してからカスタムビルド規則を変更し、再度インテル・プロジェクトに変換して戻します。

カスタムビルド規則に対して定義されている文字列のプロパティーを変更する場合、利用可能なマクロのリストにインテル固有のマクロ (`$(icInstallDir); $(icIDEInstallDir); $(icProjectExt); $(icProjectFileName)` など) は含まれていません。ただし、カスタムビルド規則のプロパティー値でインテル固有のマクロを使用することはできます。指定されたマクロは正しく機能します。

インテル® C++ コンパイラーの Visual Studio* への統合では、現在 Visual C++* のツールのビルド順序機能 (Visual C++* の [Tool Build Order (ツールのビルド順序)] ダイアログボックスから利用可能) をサポートしていません。つまり、ビルドステップの実行順序を変更することはできません。

3.5.2.3Visual Studio 2010 の言語パック*

インテル® C++ Composer XE 2011 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 12.1\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 12.1\<locale-ID>” にコピーします。
2. “<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 12.0\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 12.0\<locale-ID>” にコピーします。
3. "<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® C++ Composer XE 2011 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® 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。

3.5.4 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。

3.5.5 スタティック・セキュリティー解析の既知の問題

3.5.5.1 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・セキュリティー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・セキュリティー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・スイッチ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*)。**このスイッチは、スタティック・セキュリティー解析の結果が作成されるリンクステップで追加する必要があります。**コンパイルステップで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、このスイッチを [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・セキュリティー解析を行う場合は、`-disable-id 12020,12040` スイッチを 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・セキュリティー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方にあるため、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® インテグレーテッド・パフォーマンス・プリミティブ

このセクションでは、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 (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 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/o6nf00 (英語)) およびインテル® IPP リリースノート (http://intel.ly/pSaf2j (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/o6nf00) のドキュメントのリンクを参照してください。

4.1 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。

4.2 インテル® IPP コードサンプル

インテル® IPP コードサンプルは、以下の Web サイトから入手できます。
http://intel.ly/pnsHxc (英語)

サンプルには、オーディオ/ビデオコーデック、画像処理、メディア・プレーヤー・アプリケーション、C++/C#/Java* からの呼び出し関数のソースコードが含まれています。サンプルのビルド方法についての説明は、各サンプルのインストール・パッケージの `readme` ファイルをご覧ください。

5 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® マス・カーネル・ライブラリーの変更点、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、次の Web サイトを参照してください。
http://intel.ly/rlTU0r

5.1 注意事項

- インテル® MKL に含まれる GMP* 数学関数は、将来のリリースでは削除されます。
- タイミング関数 mkl_set_cpu_frequency() は古い関数です。代わりに、『インテル® MKL リファレンス・マニュアル』で説明されている mkl_get_max_cpu_frequency()、mkl_get_clocks_frequency()、mkl_get_cpu_frequency() を使用してください。
- PARDISO ドメインを指定するために定義されている MKL_PARDISO 定数は、mkl_domain_set_num_threads() 関数で使用できなくなりました。代わりに、MKL_DOMAIN_PARDISO を使用してください。
- 畳み込みルーチンと相関ルーチンは、将来のリリースでは 10.2 Update 3 との下位互換性はなくなります。
- インテル® MKL Windows* 版の OpenMP* スタティック・ランタイム・ライブラリーは、将来のリリースで削除されます。

5.2 本バージョンでの変更

5.2.1 最初のリリースでの変更

- BLAS
  - 一度に 2 つの行列-ベクトル積を計算するための新しい関数: [D/S]GEM2VU、[Z/C]GEM2VC
  - 混合精度の一般的な行列-ベクトル積を計算するための新しい関数: [DZ/SC]GEMV
  - 2 つのスケールされたベクトルの和を計算するための新しい関数: *AXPBY

    - 主要関数においてインテル® AVX による最適化: SMP LINPACK、レベル 3 BLAS、DDOT、DAXPY
- LAPACK
    - 行優先順に対応した LAPACK 用の C インターフェイス
    - 1 つの新しい計算ルーチン (*GEQRFP)、2 つの新しい補助ルーチン (*GEQR2P と *LARFGP)、LAPACK 3.2.1 のアップデートを含む Netlib LAPACK 3.2.2 との統合
    - 主要関数においてインテル® AVX による最適化: DGETRF、DPOTRF、DGEQRF
- PARDISO
    - マルチコア環境で問題と解のステップのパフォーマンスが向上
    - スパースの右辺の解算出と部分解ベクトルを出力する部分解算出の追加
    - アウトオブコア (OOC) 因数分解のパフォーマンスが向上
    - ゼロベース (C スタイル) の配列インデックスのサポート
    - 対称行列のスパースデータ構造で行列の対角上のゼロが不要
    - 新しい ILP64 PARDISO インターフェイスにより、LP64 ライブラリーにリンクされている場合に LP64 と ILP64 の両バージョンを使用可能
    - OOC モードでディスクにファイルを格納するのに必要なメモリーを並べ替え直後に予測可能
- スパース BLAS
    - 形式変換関数ですべてのデータ型に対応 (単精度/倍精度の実数/複素数データ)、および関数の戻り値として並べ替えあり/並べ替えなし配列を使用可能
- FFT
    - 新しい MPI FFTW 3.3alpha1 ラッパーによる新しいクラスター機能
    - クラスター FFT のロードバランスの改善によりパフォーマンスが向上
    - すべての 1D/2D/3D FFT においてインテル® AVX による最適化
    - SSE4.2 命令セットをサポートするすべてのシステムにおいて、基数が混在する単精度/倍精度データの 2D/3D FFT のパフォーマンスが向上
    - 2D/3D FFT における 2 つの実数配列として表される分割複素数データのサポート
    - 長さが大きな素数である 1D 複素数-複素数変換のサポート
    - クラスター 1D 複素数変換のハイブリッド並列化 (MPI + OpenMP)、および (MPI プロセス数の倍数である) ベクトル長のパフォーマンスの向上
- VML
    - (ax+b)/(cy+d) の計算を行うための新しい関数。a、b、c、d はスカラー、x、y は実数ベクトル: v[s/d]LinearFrac()
    - 主要関数においてインテル® AVX による最適化
    - デノーマル数をゼロに設定するための新しいモデル、複素ベクトルのオーバーフロー・サポート、各 VML 関数に対して精度を設定するための追加パラメーターを含む新しい関数
- VSL
    - 新しいサマリー統計関数群。基礎統計、共分散/相関関係、プールされたグループ/部分/厳密な共分散/相関関係、分位数/変量分位数、外れ値検出アルゴリズム、欠測値をサポート
        - パフォーマンスが最適化されたアルゴリズム: 欠測値をサポートするための MI アルゴリズム、厳密な共分散を計算するための TBS アルゴリズム、外れ値を検出するための BACON アルゴリズム、(変量データの) 分位数を計算するための ZW アルゴリズム、プールされた共分散を計算するための 1PASS アルゴリズム
    - SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
    - インテル® AVX による最適化: MT19937 と MT2203 BRNG
- ドキュメント: Microsoft* Visual Studio* 2010 に統合される Microsoft* Help Viewer* 1.x 形式の製品ドキュメント
- ランタイムにディスパッチされるダイナミック・ライブラリーの追加により、ランタイムに検出された CPU またはライブラリー関数呼び出しに応じて、依存性のあるライブラリーを動的にロードする単一のインターフェイス・ライブラリーへのリンクが可能

- 新しいディレクトリー構造により、インテル® MKL ライブラリーとインテル® Parallel Studio XE 製品ファミリーの統合が単純化され、これまでの "em64t" ディレクトリーが "intel64" ディレクトリーに変更
- 本リリースではインテル® Itanium® アーキテクチャー (IA-64) をサポートしていないため、IA-64 用の最新リリースはインテル® MKL 10.2
- スパースソルバー機能をインテル® MKL のコア・ライブラリーに完全統合。また名前に "solver" を含むライブラリーを製品から削除

5.2.2 Update 1 での変更

- PARDISO/DSS: F90 オーバーロード API の追加 (詳細は、インテル® MKL リファレンス・マニュアルを参照してください)
- PARDISO: 統計情報をより見やすく改良
- スパース BLAS: 最新のインテル® プロセッサーにおいて ?BSRMM 関数のパフォーマンスが向上
- FFT: 負のストライドのサポート
- FFT サンプル: DFTI と FFTW3 の両方のインターフェイスを使用した分割複素 FFT の C および Fortran サンプルの追加
- VML: SSE2 および SSE3 対応システムにおいて、インプレースの Add/Sub/Mul/Sqr 実関数のパフォーマンスが向上
- ポアソン・ライブラリー: ポアソン・ライブラリー関数のデフォルトの動作をシリアルから並列に変更
- 問題の修正: http://intel.ly/rlTU0r (英語)

5.2.3 Update 2 での変更

- BLAS: インテル® Xeon® プロセッサー 5600 番台において転置関数のパフォーマンスが向上
- BLAS: 転置ルーチンのサンプルの追加
- FFT: 必要な精度の関数のみをリンクすることでアプリケーションのフットプリントを小さくする方法を示した Fortran サンプルの追加
- FFT: CCE ストレージを使用するインプレース実数変換にストライドの一貫性チェックを追加
- FFT: 多次元変換のスレッド化の追加
- VSL: クアッドコア インテル® Xeon® プロセッサー 5500 番台において単精度/倍精度の多変量ガウス分布乱数ジェネレーターのパフォーマンスが向上
- VML: インテル® Xeon® プロセッサー 5500 番台において Add、Mul、Sub 関数のインプレース操作のパフォーマンスが向上
- 問題の修正: http://intel.ly/rlTU0r (英語)

5.2.4 Update 3 での変更

- BLAS: インテル® Xeon® プロセッサー 5400 番台を搭載した 32 ビットの Windows* システムにおいて DSYRK、DTRSM、DGEMM のマルチスレッド・パフォーマンスが向上
- LAPACK: 対称/エルミート行列関数および補助関数における連立線形方程式ソルバーの向上、CS (余弦/正弦) 分解を含む Netlib LAPACK 3.3 の実装
- PARDISO: 0 ベースの順列ベクトルの入力をサポート
- PARDISO: pardisoinit() ルーチンのドキュメント化
- PARDISO: 複数の右辺 (RHS) を含む PARDISO のシリアル・パフォーマンスが向上
- PARDISO: 小さな行列のパフォーマンスを向上させる解のステップの並列化の独立制御。詳細は、iparm(25) の説明を参照してください。
- PARDISO: 後方代入の減少による右辺全体の部分解計算。詳細は、iparm(31) の説明を参照してください。
- FFT: 最大 3 ～ 7 次元の実数 FFT 変換の実装
- FFT: 2 つの実数配列として表される分割複素数データを使用した多次元複素数変換の並列化

- クラスター FFT: FORTRAN 90 インターフェイスの拡張による実数-複素数変換への対応、および新しいサンプルの追加
- VML: 新しい Pack/Unpack 複素関数と Gamma/LGamma 実関数の追加
- VML: インテル® Xeon® プロセッサー 5600 番台およびインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーで、すべての関数におけるショートベクトル (< 100) の演算、すべての関数におけるアライメントされていない入力ベクトルの演算、sPow2o3 関数、拡張パフォーマンス (EP) バージョンの Add および Sub 複素関数のパフォーマンスが向上
- VSL: 乱数ジェネレーター (RNG) ストリームからメモリーへの保存、またはメモリーからの復元を行うための関数の追加
- VSL: 新しい UniformBits32 関数および UniformBits64 関数の追加
- VSL: MT2203 BRNG でサポートされる一意のストリーム数を 1024 から 6024 に拡張
- 問題の修正: http://intel.ly/rITU0r (英語)

5.2.5 Update 4 での変更

- BLAS: インテル® Xeon® プロセッサー 5400 番台以降において DTRMM のパフォーマンスが向上
- BLAS: すべての 64 ビット対応プロセッサー、特にインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーにおいて DTRSM のパフォーマンスが向上
- LAPACK: LAPACK 3.3.1 リリースでの問題の修正に対応
- OOC PARDISO: アウトオブコア処理に必要なメモリー量の推定が向上
- FFT: スレッディングの改善による 1D 実数 FFT スケーリングの向上
- FFT: 新しいシングル・ダイナミック・ライブラリー・リンク・モデルを使用するように C および Fortran の FFT サンプルを更新
- VML: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、すべての精度で、Hypot 実関数と Abs 複素関数の単精度のパフォーマンス拡張バージョン、および Arg、Div、Mul、MulByConj 複素関数のパフォーマンスが向上
- サービス関数: インテル® MKL のサービス関数の追加と拡張 (詳細は、http://software.intel.com/en-us/articles/intel-mkl-103-release-notes/ (英語) のオンライン・リリースノートを参照してください)
- 問題の修正: http://intel.ly/rITU0r (英語)

5.2.6 Update 5 での変更

- BLAS: パフォーマンスの向上: {S,C,Z}TRSM (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) 対応プロセッサー); {S,D}GEM2VU (インテル® AVX 対応プロセッサー、インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー 5500 番台)
- BLAS: スケーリングの向上: ?TRMV (すべてのアーキテクチャーの大規模行列); DGEMM (インテル® Xeon® プロセッサー 5400 番台でスレッド数が奇数の場合)
- LAPACK: LAPACK 3.3.1 の拡張とそれぞれの LAPACKE インターフェイスに対応
- LAPACK: 一般固有値問題に使用される ?SYGST と ?HEGST のパフォーマンスが向上
- LAPACK: LU 分解された行列の逆行列のパフォーマンスが向上 (?GETRI)
- PARDISO: 転置と共役転置の解算出 (ATx=b と AHx=b) の追加; 圧縮されたスパース列 (CSC) 形式のサポート
- PARDISO: MKL_PARDISO_OOC_MAX_SWAP_SIZE 環境変数とインコア PARDISO を使用して、必要なメモリー量が利用可能なメモリー量をやや上回る場合に、アウトオブコア PARDISO のパフォーマンスが向上
- 最適化ソルバー:RCI Trust-Region ソルバーに Inf と NaN のチェックを追加
- FFT: インテル® SSE3 以降に対応したインテル® プロセッサーにおいて、サポートされているすべての精度で 2x2x2 から 10x10x10 の小さな立方体に対する 3D FFT のパフォーマンスが向上

- FFT サンプル: インテル® MKL の DFTI と FFTW の一般的な使用例を示すサンプルコードを変更
- VSL: インテル® Core™ i7-2600 プロセッサーを搭載した 64 ビットのオペレーティング・システムのマシンにおいて、単精度の MT19937 および MT2203 基本乱数ジェネレーターのパフォーマンスが向上
- VSL: SOBOL 準乱数ジェネレーターの整数バージョンのパフォーマンスが向上。

5.2.7 Update 6 での変更

- スパース BLAS: mkl_?csrbsr 変換関数に BSR 形式から CSR 形式への変換時にゼロの要素を検出し削除する新しいオプションを追加
- Windows* での DLL ロード動作の変更: インテル® MKL の DLL を PATH 上の個別のディレクトリーに配置することはできなくなりました。すべての DLL を実行ファイルと同じディレクトリーに配置するか、PATH 環境変数で指定されている別のディレクトリーに配置する必要があります。
- 問題の修正: http://intel.ly/rlTU0r (英語)

5.2.8 Update 7 での変更

- BLAS: 最近のすべてのインテル® Xeon® プロセッサーにおいて、出力行列が小さく外積が大きい (つまり、入力行列が矩形) の場合に DSYRK/SSYRK のマルチスレッド・パフォーマンスが向上
- BLAS: 最近のすべてのインテル® Xeon プロセッサーにおいて、小さな問題 (<10、beta =1) で ?GEMM のパフォーマンスが向上
- BLAS: 32 ビットのプログラムを実行するインテル® Xeon プロセッサー 5500/5600/7500 番台 において、INCX=1 の小さな問題で DSCAL のパフォーマンスが向上
- BLAS のような拡張: 正方行列のインプレース転置のキャッシュ効率とスレッディングの向上
- PARDISO: PARDISO 用の独立したスレッド化コントロールの追加; mkl_domain_set_num_threads() 関数での MKL_DOMAIN_PARDISO の使用
- ポアソン・ライブラリー: 2D/3D 周期的境界条件のサポートの追加
- ドキュメント・ディレクトリーへのリンク・ライン・アドバイザーの追加
- libtool などのスクリプトツールとともに使用するコマンドライン・リンク・ツールの追加
- 次のコンポーネントの関数の stdcall プロトタイプを含む C ヘッダーファイルの追加: BLAS、スパース BLAS、LAPACK、PARDISO/DSS、RCI 反復ソルバー、ベクトル数学関数、ベクトル・スタティスティカル関数、およびサポート関数
- mkl_domain_set_num_threads() 関数でドメインの指定に使用する定数の名前を変更 (例: MKL_BLAS は MKL_DOMAIN_BLAS に変更); MKL_PARDISO を除き、古い名前も継続して使用可能
- 問題の修正: http://intel.ly/rlTU0r (英語)

5.2.9 Update 8 での変更

- データ適合コンポーネント: ベクトルスプラインの構築、セル探索や二分探索、スプライン補間の評価、微分、積分の 1 次元アルゴリズムをカバーする新しいデータ適合関数のセットを追加。以下のサポートを含む。
    - 1 次スプライン、2 次スプライン、3 次スプライン、ステップワイズ法、ユーザー定義スプライン
    - 最適なパフォーマンスのために構成パラメーターを用いたセル探索
    - ユーザー定義補間 (内挿) および補外 (外挿)
    - ベクトル値関数
    - 列優先および行優先格納形式
- スパース BLAS: 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) 対応のコア数の多いシステムにおいて、非常に疎な行列の CSR (Compressed Sparse Row) 形式の行列-ベクトル乗算 (?CSRMV) のパフォーマンスが向上

- FFT: インテル® AVX 対応システムにおいて、1D 倍精度 FFT のパフォーマンスが向上
- 統計関数: インテル® Core™ プロセッサーにおいて、分散共分散行列および相関行列の計算 (FAST 法) のパフォーマンスとスケーラビリティーが向上
- スタティック・ライブラリー・ファイルからカスタム DLL を構築するための Microsoft* Visual Studio* プロジェクト・ツールを追加
- 問題の修正: http://intel.ly/rITU0r

5.2.10 Update 9 での変更

- LAPACK: 非常に小さなサイズ (~10 x 10) で [C/Z]GEEV のパフォーマンスが向上
- FFT: 大幅なパフォーマンス向上のためにインプレースの 1D FFT 実関数をスレッド化
- FFT: 32 ビットのオペレーティング・システムを実行するインテル® Xeon® プロセッサー E5 ファミリーのシステムにおける 2 の累乗の倍精度複素 1D FFT のスケーラビリティー向上のために新しいアルゴリズムを追加
- 乱数ジェネレーター: 新しいインテル® マイクロアーキテクチャー (開発コード名: Ivy Bridge) ベースのプロセッサーで利用可能なハードウェアをサポートする、RdRand 命令に基づく非決定性乱数ジェネレーターをサポート
- ベクトル算術関数: インテル® Core™ プロセッサーにおいて、Erf() 関数および Pow3o2() 関数のパフォーマンスが向上
- データ適合: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、スプラインベースの評価、微分、積分ルーチンのパフォーマンスが向上
- 問題の修正: http://intel.ly/rITU0r

5.2.11 Update 10 での変更

- BLAS: 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) 対応の 32 ビット・プログラムで dznrm2 および dnrm2 のパフォーマンスが向上
- LAPACK: LAPACK バージョン 3.4.0 のサポートを追加
- データ適合: インテル® Xeon® プロセッサー E7-4870/E5-2690 において、以下の領域における SearchCells1D() 関数のパフォーマンスが向上。
    - 補間点の数が 32 を超える任意の非一様および擬一様領域
    - 補間点の数が 32 未満のすべての種類の領域
- 問題の修正: http://intel.ly/rITU0r

5.3 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 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。

6 インテル® スレッディング・ビルディング・ブロック

本バージョンのインテル® スレッディング・ビルディング・ブロックの変更に関する詳細は、TBB ドキュメント・ディレクトリーの CHANGES というファイルを参照してください。

6.1 既知の問題

インテル® スレッディング・ビルディング・ブロック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。

6.1.1 ライブラリーの問題

- インテル® スレッド・チェッカーまたはインテル® スレッド・プロファイラーを使用した際により正確な結果を得るには、インテル® TBB とともに使用する前にそれらの製品の最新のアップデート・リリースをダウンロードしてください。
- 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。

7 インテル® Parallel Debugger Extension

このセクションでは、インテル® Parallel Debugger Extension の変更点、新機能、および最新情報をまとめています。

7.1 インテル® Parallel Debugger Extension のサポート終了予定

将来のメジャーリリースでは、インテル® Parallel Debugger Extension は削除される可能性があります。削除された場合、次の機能が使用できなくなります。

- OpenMP* ウィンドウ
- データ共有検出 (代わりに、インテル® Inspector XE のデータ共有検出機能を利用できます)
- ベクトルレジスター・ウィンドウ (Visual Studio* の標準のレジスターウィンドウでもベクトルレジスターを確認できます)

7.2 新機能

- インテル® Cilk™ Plus サポート
  - o インテル® Cilk™ Plus プログラムをデバッグ時に再コンパイルなしでシリアル実行することができます。
  - o インテル® Cilk™ Plus のワーカースレッドは、Visual Studio* デバッガーで明確にマークされます。
  - o Cilk Plus スレッド・スタック・ウィンドウにスチールポイントと現在の Cilk Plus ワーカーが表示されます。
- [Threads (スレッド)] ウィンドウ
  - o データ共有検出の向上
  - o OpenMP* 3.0 のサポート
  - o Windows* OS の同期関数のサポート
  - o データ共有検出の解析パフォーマンスの向上

7.3 既知の問題

- インテル® Parallel Debugger Extension は、次の正しくない警告を出力することがあります。

  ```
  OMP:Warning #90: ittnotify:Lookup of “g__itt_<function>“ function
  in <installdir><redistlibpath>\pdbx.dll library failed.
  ```

  これは、`/debug:parallel` コンパイラー・オプションを指定してビルドされた OpenMP* 実行ファイルをデバッグした場合に出力されます。この警告は無視しても問題ありません。
- Microsoft* Visual Studio* 2005 を使用している場合、6 つのインテル固有の例外を手動で有効に設定する必要があります。[デバッグ] > [例外] を選択し、[Win32 Exceptions] ツリーを展開して、以下の項目を有効にします。

  ```
  a1a01db0 Intel Parallel Debugger Extension Exception 0
  a1a01db1 Intel Parallel Debugger Extension Exception 1
  a1a01db2 Intel Parallel Debugger Extension Exception 2
  a1a01db3 Intel Parallel Debugger Extension Exception 3
  a1a01db4 Intel Parallel Debugger Extension Exception 4
  a1a01db5 Intel Parallel Debugger Extension Exception 5
  ```

  これは、プロジェクトごとに 1 回設定します。
- デバッグセッション中にインテルのデバッグ例外を無効にすると、Visual Studio* (Visual Studio* 2008 SP1 まで) がハングアップすることがあります。
- インテル® Parallel Debugger Extension を使用するには、OpenMP* ライブラリーが動的にリンクされている必要があります (デフォルト)。インテル® Parallel Debugger Extension を使用する場合、OpenMP* ライブラリーのスタティック・リンクを指定する `/Qopenmp-link:static` を使用しないでください。
- 並列デバッグを行う前に並列デバッグ・インストルメンテーションを有効にしてください (`/debug:parallel` スイッチ)。[プロジェクト] > [プロパティ ページ] > [構成プロパティ] > [C/C++] > [Debugging (デバッグ)] > [Enable Parallel Debug Checks (並列デバッグチェックを有効にする)] で [Yes (/debug:parallel) (はい (/debug:parallel))] を選択します。この設定を行わない場合、デバッガーはデータ共有イベントや再入可能な呼び出しでの中断を検出できません。
- Microsoft* Visual Studio* 2010 の場合:最初に Microsoft* Visual C++* でプロジェクトをビルドするように指定し、後でインテル® C++ コンパイラーを使用するように変更した場合、並列デバッグ環境のプロパティー設定 [Auto (自動)] が認識されるように、いったんソリューションを閉じてから再度開く必要があります。

- Microsoft* Visual Studio* 2008 を使用し、64 ビット・アプリケーションのデバッグを行う場合、Visual Studio* 2008 Service Pack 1 がインストールされている必要があります。
    - サービスパックがインストールされていない場合、Visual Studio* 2005 および 2008 での 64 ビット・アプリケーションのデバッグは、低メモリー領域にリンクされる場合のみ行うことができます。低メモリー領域にリンクされない場合、デバッグ対象が終了するまでイベントは表示されません。終了後、すべてのイベントがイベントウィンドウに表示されます。64 ビット・アプリケーションを適切にデバッグするには、[プロジェクト] > [プロパティ] > [Linker (リンカー)] > [Advanced (詳細)] でベースアドレスを 0x10000 に設定します。
- [Data Sharing Events (データ共有イベント)] ウィンドウで関数のローカル変数やヒープ変数が「???」と表示されます。
- SSE レジスターウィンドウが 64 ビット・アプリケーションで動作しません。ウィンドウに「???」と表示されます。
- スタティック・ローカル変数のフィルターがコンテキスト・メニューから正しく設定されません。
- 逆アセンブルビューで再入可能な呼び出しの検出が停止します。スタティック関数の場合、正しく動作しません。デザインモードでは、`{,,myapp.exe}my_extern_function` など、適切なコンテキスト演算子の後で関数を使用してください。
- デバッガー拡張ウィンドウの配置が "docked" から "floating" に変更されるとウィンドウは空のままです。この問題を回避するには、"docked" のままにしておくか、または配置の変更後にデバッグセッションを再起動します。
- デバッガー拡張では、Visual Studio* からアプリケーションを開始する必要があります。既存のプロセスへアタッチしている場合は動作しません。
- ウィンドウが非表示、あるいは閉じられた後に再度開かれた場合は、デフォルト (16 進) 設定に戻ります。

7.4 ドキュメント

インテル® Parallel Debugger Extension のドキュメントは、Microsoft* Visual Studio* の [ヘルプ] メニューから、または [Parallel Debugger Extension] ウィンドウがアクティブな状態で F1 キーを押して表示することができます。`debugger-documentation.htm` にある "HTML バージョン" へのリンクをクリックして表示することもできます。

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を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。

本書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。

本書で紹介されている注文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。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、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。