インテル® Parallel Studio XE 2015 Composer Edition for C++ Linux* 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 4
1.1 変更履歴 4
1.1.1 Update 1 4
1.1.2 インテル® Composer XE 2013 SP1 以降 (インテル® Parallel Studio XE 2015 Composer Edition での変更) 4
1.2 製品の内容 5
1.3 インテル® デバッガー (IDB) を削除 6
1.4 動作環境 6
1.4.1 SuSE Enterprise Linux 10* のサポートを終了 8
1.4.2 Red Hat* Enterprise Linux* 5 および Debian 6* のサポート終了予定 8
1.5 ドキュメント 8
1.6 サンプル 8
1.7 日本語サポート 9
1.8 テクニカルサポート 9
2 インストール 9
2.1 GUI インストーラー 10
2.2 オンライン・インストーラー 10
2.2.1 `http_proxy` が設定されているのに sudo によるインストールで接続に失敗する 10
2.2.2 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 10
2.3 インテル® Software Manager 10
2.4 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 11
2.5 クラスターでのインストール 11
2.6 サイレント・インストール 11
2.6.1 非インタラクティブ・カスタム・インストールのサポート 11

2.7 ライセンスサーバーの使用 12
2.8 Eclipse* 統合のインストール 12
2.9 既知のインストールの問題 12
2.10 インストール先フォルダー 12
2.11 削除/アンインストール 14
3 インテル® C++ コンパイラー 14
3.1 互換性 14
3.2 新機能と変更された機能 14
3.2.1 IA-32 およびインテル® 64 アーキテクチャー向けインテル® 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 (インテル® コンパイラー 15.0.1) 15
3.2.2 #pragma simd で MIN/MAX リダクションをサポート 16
3.2.3 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 16
3.2.4 スタティック解析は非推奨 (廃止予定) 16
3.2.5 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 16
3.2.6 _bittest および _bittestandcomplement 組込み関数をサポート (インテル® C++ コンパイラー 15.0) 16
3.2.7 コンパイルエラーを回避するインテル・バージョンの x86intrin.h を提供 (インテル® C++ コンパイラー 15.0) 17
3.2.8 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) 17
3.2.9 OpenMP* 機能の追加サポート (インテル® C++ コンパイラー 15.0) 17
3.2.10 インテル® C++ コンパイラー 15.0 のインテル® Cilk™ Plus の変更点 17
3.2.11 複数の関数バージョン (GNU* 互換、CPU ディスパッチ用) 18
3.2.12 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 18
3.2.13 GNU* C 標準インクルード・ファイルを提供 18
3.2.14 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 18
3.2.15 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 18
3.3 新規および変更されたコンパイラー・オプション 18
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 18
3.3.2 -o で始まるコンパイラー・オプションは非推奨 (廃止予定) 19
3.3.3 -ansi-alias がデフォルトでオン 20
3.3.4 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 20
3.3.5 データ・アライメントに関係なく同じコードを実行する -no-opt-dynamic-align オプション 20

3.3.6 PGO によるスレッドセーフなプロファイル生成が可能 20
3.3.7 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 20
3.4 その他の変更 20
3.4.1 コンパイラー環境の設定 20
3.4.2 デフォルトの命令セットがインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) を必要とするものに変更 21
3.4.3 -std=c99 を使用すると "asm" キーワードが認識されない (インテル® C++ コンパイラー 15.0 Linux* 版) 21
3.5 既知の問題 21
3.5.1 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 21
3.5.2 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 21
3.5.3 インテル® グラフィックス・テクノロジーへのオフロードの既知の問題 23
3.5.4 インテル® Cilk™ Plus の既知の問題 24
3.5.5 ガイド付き自動並列化の既知の問題 24
3.5.6 スタティック解析の既知の問題 24
4 GNU* GDB デバッガー 25
4.1 機能 25
4.2 GNU* GDB の使用 26
4.3 ドキュメント 26
4.4 既知の問題と変更点 26
4.4.1 GDB* を Eclipse* IDE で使用する場合の問題 26
4.4.2 オフロード・デバッグ・セッションの安全な終了方法 27
4.4.3 ソース・ディレクトリーの設定によるインテル® MIC アーキテクチャー側のデバッガーのアサーション 27
4.4.4 デバッガーから _Cilk_shared 変数へのアクセス 27
5 Eclipse* 統合 27
5.1 提供されている統合 28
5.1.1 統合に関する注意事項 28
5.2 Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法 28
5.2.1 Eclipse* への GNU* プロジェクト・デバッガーの統合 29
5.3 Eclipse*、CDT、および JRE の入手方法とインストール方法 29
5.3.1 JRE、Eclipse*、CDT のインストール 29
5.4 インテル® C++ コンパイラーで開発するための Eclipse* の起動 29
5.5 Fedora* システムでのインストール 30
5.6 コンパイラー・バージョンの選択 30
6 インテル® IPP 30

6.1 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 30
7 インテル® MKL 30
7.1 本バージョンでの変更 31
7.1.1 インテル® MKL 11.2 Update 1 の新機能 31
7.1.2 インテル® MKL 11.2 の新機能 32
7.2 権利の帰属 35
8 インテル® TBB 4.3 35
9 著作権と商標について 36

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 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

- IA-32 およびインテル® 64 アーキテクチャー向けインテル® 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 (インテル® コンパイラー 15.0.1)
- 日本語版を含む最初のアップデート
- SuSE Enterprise Linux Server* 12 をサポート
- インテル® C++ コンパイラー 15.0.1
- インテル® MKL 11.2 Update 1
- インテル® IPP 8.2 Update 1
- インテル® TBB 4.3 Update 1

1.1.2 インテル® 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
- -ansi-alias がデフォルトでオン (警告なしでランタイムの動作が変更される可能性あり)
- 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 -opt-report、-vec-report、-openmp-report、-par-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)
- C++11 言語をフルサポート (可能性のある制限については詳細を参照)
- OpenMP* 4.0 の機能を追加サポート

- #pragma simd で MIN/MAX リダクションをサポート
- インテル® Cilk™ Plus の変更
- オンライン・インストーラーでのカスタム・インストール設定
- データ・アライメントに関係なく同じコードを実行する -no-opt-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
- 複数の関数バージョン (GNU* 互換、CPU ディスパッチ用)
- ラムダ関数のデバッグを強化
- デフォルトのデバッグ情報を DWARF バージョン 3 形式に変更
- 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
- 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
- _bittest および _bittestandcomplement 組込み関数をサポート
- gcc 4.9 をサポート
- binutils 2.24 をサポート (binutils 2.19 のサポートを終了)
- Red Hat* Enterprise Linux* 7 をサポート
- Ubuntu* 14.04 LTS をサポート
- SUSE LINUX Enterprise Server* 10 のサポートを終了
- Debian 6* は非推奨 (サポート終了予定)
- Python* なしで GNU* プロジェクト・デバッガーが利用可能
- インテル® デバッガー (IDB) を削除
- スタティック解析は非推奨 (廃止予定)
- -o で始まるコンパイラー・オプションは非推奨 (廃止予定)
- インテル® C++ コンパイラー 15.0.0
- インテル® MKL 11.2
- インテル® IPP 8.2
- インテル® TBB 4.3
- インテル® Cilk™ Plus のサポートが強化された GNU* プロジェクト・デバッガー (GDB) 7.7

1.2 製品の内容

*インテル® Parallel Studio XE 2015 Update 1 Composer Edition for C++ Linux** 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 15.0.1。Linux* 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、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。
- GNU* プロジェクト・デバッガー (GDB) 7.7
- インテル® MKL 11.2 Update 1
- インテル® IPP 8.2 Update 1
- インテル® TBB 4.3 Update 1
- Eclipse* 開発環境への統合
- 各種ドキュメント

1.3 インテル® デバッガー (IDB) を削除

インテル® デバッガー (IDB) はこのリリースから削除されました。代わりに GNU* プロジェクト・デバッガー (GDB) ベースのデバッガーが提供されます。

1.4 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)
    - 64 ビット・アプリケーションおよびインテル® MIC アーキテクチャーを対象とするアプリケーションの開発は、64 ビット・バージョンの OS でのみサポートしています。32 ビット・アプリケーションの開発は、32 ビット・バージョンまたは 64 ビット・バージョンの OS のいずれかでサポートしています。64 ビット・バージョンの OS で 32 ビット・アプリケーションを開発する場合は、Linux* ディストリビューションからオプションのライブラリー・コンポーネント (ia32-libs、lib32gcc1、lib32stdc++6、libc6-dev-i386、gcc-multilib、g++-multilib) をインストールする必要があります。
- インテル® MIC アーキテクチャー向けの開発/テスト:
    - インテル® Xeon Phi™ コプロセッサー
    - インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS)
- インテル® グラフィックス・テクノロジーへのオフロードまたはネイティブサポートの開発/テスト
    - オフロードは 64 ビット・アプリケーションでのみサポートされます。
    - SUSE LINUX Enterprise Server* 11 SP3
        - カーネル 3.0.76-11
    - Ubuntu* 12.04 LTS
        - カーネル 3.2.0-41 (第 3 世代インテル® Core™ プロセッサー)
        - カーネル 3.8.0-23 (第 4 世代インテル® Core™ プロセッサー)
    - 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 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。

- 機能を最大限に活用できるよう、マルチコアまたはマルチプロセッサー・システムの使用を推奨します。
- RAM 2GB (4GB 推奨)
- 7.5GB のディスク空き容量 (すべての機能をインストールする場合)
- 次の Linux* ディストリビューションのいずれか (本リストは、インテル社により動作確認が行われたディストリビューションのリストです。その他のディストリビューションでも動作する可能性はありますが、推奨しません。ご質問は、テクニカルサポートまでお問い合わせください。)
  - Fedora* 20
  - Red Hat* Enterprise Linux* 5、6、7
  - SUSE Linux Enterprise Server* 11、12
  - Ubuntu* 12.04 LTS (64 ビットのみ)、13.10、14.04 LTS
  - Debian* 6.0、7.0
  - インテル® Cluster Ready
- Linux* 開発ツール・コンポーネント (gcc、g++ および関連ツールを含む)
  - gcc バージョン 4.1-4.9 をサポート
  - binutils バージョン 2.17-2.24 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 4.3 および次の両方
  - Eclipse* C/C++ Development Tools (CDT) 8.2 以降
  - Java* ランタイム環境 (JRE) 6.0 (1.6) 以降
- Eclipse* Platform 4.2 および次の両方
  - Eclipse* C/C++ Development Tools (CDT) 8.1 以降
  - Java* ランタイム環境 (JRE) 6.0 (1.6) 以降
- Eclipse* Platform 3.8 および次の両方
  - Eclipse* C/C++ Development Tools (CDT) 8.1 以降
  - Java* ランタイム環境 (JRE) 6.0 (1.6) 以降

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

注

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

アーキテクチャー・プロセッサー上で動作するコードを生成できます。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。
- 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。

1.4.1 SuSE Enterprise Linux 10* のサポートを終了

SuSE Enterprise Linux 10* 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。

1.4.2 Red Hat* Enterprise Linux* 5 および Debian 6* のサポート終了予定

将来のリリースでは、Red Hat* Enterprise Linux* 5 および Debian 6* はサポートされなくなる予定です。

1.5 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。

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

1.6 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。

1.7 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・システムで英語版を使用する場合は、http://intel.ly/qhINDv (英語) の説明を参照してください。

1.8 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

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、次のコマンドを使用して、tgz ファイルを書き込み可能な任意のディレクトリーに展開します。

```
tar -xzvf name-of-downloaded-file
```

その後、展開したファイルを含むディレクトリーに移動 (cd) し、次のコマンドでインストールを開始します。

```
./install.sh
```

手順に従ってインストールを完了します。

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。

インストール・スクリプトは、バックグラウンド・プロセス (つまり、"./install.sh &") として実行しないでください。これはサポートされていません。

2.1 GUI インストーラー

GUI をサポートする Linux* システムで、GUI ベースのインストーラーを利用できるようになりました。GUI をサポートしていない場合 (ssh ターミナルから実行する場合など) は、コマンドラインによるインストールが可能です。

2.2 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。

2.2.1 http_proxy が設定されているのに sudo によるインストールで接続に失敗する

ほとんどの sudo プロファイルは、オリジナルユーザーから http_proxy などの特定の設定を継承しないように設定されています。/etc/sudoers ファイルに、次のようなプロキシー設定のプロパゲーションを許可する行が含まれていることを確認してください。

```
Defaults    env_keep += "http_proxy"
```

2.2.2 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 /tmp/<UID> 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。

2.3 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 Linux* 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。

2.5 クラスターでのインストール

Linux* クラスターの複数のノードに製品をインストールするには、次の手順に従う必要があります。

1) インテル® Parallel Studio XE Cluster Edition がインストールされているシステムでインストールを実行します。クラスターのマシン間をパスワードなしで ssh 接続できるように設定する必要があります。
2) ステップ "4 オプション" で、"クラスターのインストール" オプションが表示されます。デフォルトは "現在のノード" です。
3) クラスターにインストールするには、このオプションを選択して、クラスターノードの IP アドレス、ホスト名、FQDN、その他の情報が記述された `machines.LINUX` ファイル (1 行に 1 ノード) を指定します。最初の行には、現在の (マスター) ノードの情報を記述します。
4) `machines.LINUX` ファイルを指定すると、"並行インストールの数" および "共有インストール・ディレクトリーのチェック" オプションが表示されます。
5) すべてのオプションを設定してインストールを開始すると、すべてのノードの接続 (必要条件) が確認され、接続されているノードに製品がインストールされます。

2.6 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/ngVHY8 (英語) を参照してください。

2.6.1 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 Composer Edition 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` (または `-d 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。
    - `/tmp/<UID>/<package_id>`

例: `l_ccompxe_online_2015.0.0XX.sh -duplicate icc15_install_config.ini --download-dir "/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 "/temp/custom_pkg_ic15" に作成されます。

2.7 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。

2.8 Eclipse* 統合のインストール

「Eclipse* 統合」セクションを参照してください。

2.9 既知のインストールの問題

- インテル® Parallel Studio XE Composer Edition またはインテル® Composer XE のより新しいメジャーバージョンをアンインストールすると、インテル® Parallel Studio XE Composer Edition またはインテル® Composer XE の以前のバージョンがシステムに存在している場合でもシンボリック・リンクが削除されます。例えば、インテル® Parallel Studio XE 2015 Composer Edition をアンインストールすると、インテル® Composer XE 2013 SP1 がシステムに存在している場合でもシンボリック・リンクが削除されます。この問題を回避するには、以前のバージョンのディレクトリー (例えば、`composer_xe_2013_sp1.<n>.<pkg>`) を明示的に参照してください
- 「システム要件」に記述されているように、本バージョンでは、IA-32 およびインテル® 64 アーキテクチャー・ベースのシステムで Debian* または Ubuntu* をサポートしています。ただし、ライセンス・ソフトウェアの制約上、Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システム上では、インストール時に [製品を評価する (シリアル番号不要)] オプションで IA-32 コンポーネントをインストールできません。これは、[製品を評価する (シリアル番号不要)] オプションを使用する場合のみの問題です。シリアル番号、ライセンスファイル、フローティング・ライセンス、その他のライセンス・マネージャー操作、およびオフラインでのアクティベーション操作 (シリアル番号を使用) には影響はありません。Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システムで、本バージョンの IA-32 コンポーネントの評価が必要な場合は、インテル® ソフトウェア評価センター (http://intel.ly/nJS8y8 (英語)) で評価版のシリアル番号を入手してください。

2.10 インストール先フォルダー

コンパイラーは、デフォルトでは `/opt/intel` にインストールされます。本リリースノートでは、この場所を `<install-dir>` と表記します。コンパイラーは、別の場所にインストールしたり、"非 root" で任意の場所にインストールすることもできます。

`<install-dir>` 以下には次のサブディレクトリーがあります。

- `bin` – インストールされている最新バージョンの実行ファイルへのシンボリック・リンク
- `lib` – インストールされている最新バージョンの lib ディレクトリーへのシンボリック・リンク

- include – インストールされている最新バージョンの include ディレクトリーへのシンボリック・リンク
- man – インストールされている最新バージョンの man ページが含まれているディレクトリーへのシンボリック・リンク
- ipp – インストールされている最新バージョンのインテル® IPP のディレクトリーへのシンボリック・リンク
- mkl – インストールされている最新バージョンのインテル® MKL のディレクトリーへのシンボリック・リンク
- tbb – インストールされている最新バージョンのインテル® TBB のディレクトリーへのシンボリック・リンク
- ism – インテル® Software Manager 関連のファイル
- composerxe – composer_xe_2015 ディレクトリーへのシンボリック・リンク
- composer_xe_2015 – インストールされている最新バージョンのインテル® Parallel Studio XE 2015 コンパイラーのサブディレクトリーへのシンボリック・リンク
- composer_xe_2015.<n>.<pkg> – 特定のリビジョン番号のファイルが含まれている物理ディレクトリー。<n> はリビジョン番号、<pkg> はパッケージビルド ID。

各 composer_xe_2015 ディレクトリーには、インストールされている最新のインテル® Parallel Studio XE 2015 Composer Edition を参照する次のサブディレクトリーが含まれています。

- bin – コンパイラー環境とホスト環境用のコンパイラー実行ファイルへのシンボリック・リンクを設定するためのスクリプト
- pkg_bin – コンパイラーの bin ディレクトリーへのシンボリック・リンク
- include – コンパイラーの include ディレクトリーへのシンボリック・リンク
- lib – コンパイラーの lib ディレクトリーへのシンボリック・リンク
- ipp – ipp ディレクトリーへのシンボリック・リンク
- mkl – mkl ディレクトリーへのシンボリック・リンク
- tbb – tbb ディレクトリーへのシンボリック・リンク
- debugger – debugger ディレクトリーへのシンボリック・リンク
- eclipse_support – eclipse_support ディレクトリーへのシンボリック・リンク
- man – man ディレクトリーへのシンボリック・リンク
- Documentation – Documentation ディレクトリーへのシンボリック・リンク
- Samples – Samples ディレクトリーへのシンボリック・リンク

各 composer_xe_2015.<n>.<pkg> ディレクトリーには、特定のリビジョン番号のインテル® Parallel Studio XE 2015 Composer Edition を参照する次のサブディレクトリーが含まれています。

- bin – すべての実行ファイル
- pkg_bin – bin ディレクトリーへのシンボリック・リンク
- compiler – 共有ライブラリーとヘッダーファイル
- debugger – デバッガーファイル
- Documentation – ドキュメント・ファイル
- man – man ページ
- eclipse_support – Eclipse* 統合をサポートするためのファイル
- ipp – インテル® IPP のライブラリーとヘッダーファイル
- mkl – インテル® MKL のライブラリーとヘッダーファイル
- tbb – インテル® TBB のライブラリーとヘッダーファイル
- Samples – サンプルプログラムとチュートリアル・ファイル

- `uninstall` – アンインストール用のファイル

インテル® C++ コンパイラーとインテル® Fortran コンパイラーの両方がインストールされている場合、所定のバージョンおよびリビジョン番号のフォルダーが共有されます。

このディレクトリー構成により、任意のバージョン/リビジョン番号のインテル® コンパイラーを選択することができます。`<install-dir>/bin` にある `compilervars.sh [.csh]` スクリプトを参照すると、インストールされている最新のコンパイラーが使用されます。

2.11 削除/アンインストール

製品の削除 (アンインストール) は、製品をインストールしたユーザー (root または非 root ユーザー) で実行してください。インストールに sudo を使用した場合は、アンインストールの際にも使用する必要があります。インストールされているパフォーマンス・ライブラリー・コンポーネントや Eclipse* 統合コンポーネントを残したまま、コンパイラーのみを削除することはできません。

1. 端末を開いて、`<install-dir>` 以外のフォルダーに移動 (cd) します。
2. その後、次のコマンドを使用します。`<install-dir>/composer_xe_2015.<n>.<pkg>/uninstall.sh` (コマンドラインでアンインストールする場合) または `<install-dir>/composer_xe_2015.<n>.<pkg>/uninstall-GUI.sh` (GUI でアンインストールする場合)。
3. 画面の指示に従ってオプションを選択します。
4. 別のコンポーネントを削除するには、ステップ 2 と 3 を繰り返します。

同じバージョンのインテル® Fortran コンパイラーをインストールしている場合は、Fortran コンパイラーも削除されます。

使用している Eclipse* にインテル® C++ コンパイラーの Eclipse* 統合機能が追加されている場合は、Eclipse* の構成からインテルの統合拡張を削除して、構成を更新する必要があります。そのためには、[Help (ヘルプ)] メニューから [About Eclipse (Eclipse について)] を開いて [Installation Details (インストール詳細)] をクリックします。そして、[Installed Software (インストール済みのソフトウェア)] から [Intel(R) C++ Compiler XE 15.0 for Linux* OS (インテル(R) C++ コンパイラー XE 15.0 Linux* 版)]" を選択して [Uninstall... (アンインストール...)] をクリックします。処理が完了したら [Finish (完了)] をクリックして、Eclipse* の再起動を求められたら [Yes (はい)] を選択します。

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 11.0 で、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されるようになりました。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++ 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・テクノロジー向けネイティブコード生成をサポート
- インテル® グラフィックス・テクノロジーへのオフロードをサポート
- -ansi-alias がデフォルトでオン (警告なしでランタイムの動作が変更される可能性あり)
- 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
- C++11 言語をフルサポート (15.0 での新機能を含む) (-std=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)
    - 利用可能な言語機能はインストールされている gcc のバージョンに依存することに注意してください。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 gcc コンパイラー、ヘッダーファイル、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 コンパイラーおよび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、allocator_traits をサポートしていません。
- IA-32 およびインテル® 64 アーキテクチャー向けインテル® 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 (インテル® コンパイラー 15.0.1)
- OpenMP* 4.0 の機能を追加サポート
- #pragma simd で MIN/MAX リダクションをサポート
- インテル® C++ コンパイラー 15.0 のインテル® Cilk™ Plus の変更点
- 複数の関数バージョン (GNU* 互換、CPU ディスパッチ用)
- aligned_new ヘッダー
- ラムダ関数のデバッグを強化
- デフォルトのデバッグ情報を DWARF バージョン 3 形式に変更
- 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
- GNU* C 標準インクルード・ファイルを提供
- 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
- スタティック解析は非推奨 (廃止予定)
- _bittest および _bittestandcomplement 組込み関数をサポート (インテル® C++ コンパイラー 15.0)

3.2.1 IA-32 およびインテル® 64 アーキテクチャー向けインテル® 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 (インテル® コンパイラー 15.0.1)

インテル® コンパイラー 15.0.1 では、現在のインテル® 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® AVX-512 命令のサポートに加えて、インテル® AVX-512 命令対応の IA-32 およびインテル® 64 アーキテクチャー・ベースのプロセッサーでインテル® AVX-512 命令がサポートされるようになりました。これらの命令は、インライン・アセンブリー、組込み関数 (zmmintrin.h インクルード・ファイルを使用)、/Q[a]xCORE-AVX512 (Windows*) または -[a]xCORE-AVX512 (Linux*/OS X*) コンパイラー・オプションによりサポートされます。

3.2.2 #pragma simd で MIN/MAX リダクションをサポート

インテル® コンパイラー 15.0 では、#pragma simd で MIN/MAX リダクションをサポートしました。

```
#pragma simd reduction(max:simdMax)
   for(int i = 0; i < size; ++i)
      if(x[i] > simdMax)
         simdMax = x[i];


#pragma simd reduction(min:simdMin)
   for(int i = 0; i < size; ++i)
      if(x[i] < simdMin)
         simdMin = x[i];
```

3.2.3 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。

3.2.4 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。

3.2.5 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。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。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。

3.2.6 _bittest および _bittestandcomplement 組込み関数をサポート (インテル® C++ コンパイラー 15.0)

現在 Windows* でサポートされている _bittest および _bittestandcomplement 組込み関数が Linux* でサポートされました。

3.2.7 コンパイルエラーを回避するインテル・バージョンの x86intrin.h を提供 (インテル® C++ コンパイラー 15.0)

gcc 4.9 で提供される x86intrin.h をインクルードしてコンパイルすると、インテル® コンパイラーがサポートしない組込み関数への参照が行われます。この問題を解決するため、インテル® C++ コンパイラーは、コンパイルエラーを回避するインテル・バージョンの x86intrin.h を提供しています。

3.2.8 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 (-opt-report、-vec-report、-openmp-report、-par-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。

-vec-report、-openmp-report、-par-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。

3.2.9 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® コンパイラー 15.0 では、次の OpenMP* 4.0 機能が追加されました。

- `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。

3.2.10 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
- 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 __attribute__((vector_variant(...)))
- 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。

3.2.11 複数の関数バージョン (GNU* 互換、CPU ディスパッチ用)

gcc との互換性を高めるため CPU ディスパッチ用の新しいインターフェイスが追加されました。詳細は、http://gcc.gnu.org/onlinedocs/gcc/Function-Multiversioning.html (英語) を参照してください。

3.2.12 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。

3.2.13 GNU* C 標準インクルード・ファイルを提供

limits.h、stddef.h、stdbool.h、stdarg.h, stdint.h、および iso646.h ファイルがコンパイラー・パッケージの一部として提供されるようになりました。icc/icpc を使用したコンパイルでは、これらのヘッダーがインクルードされます。これらのヘッダーをインクルードしないようにするには、-no-gcc-include-dir コンパイラー・オプションを使用します。

3.2.14 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。

3.2.15 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。

3.3 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「コンパイラー・オプション」を参照してください。

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

- -xCORE-AVX512
- -axCORE-AVX512
- -mgpu-arch=<arch>
- -q[no-]opt-multi-version-aggressive
- -qopt-ra-region-strategy[=keyword]
- -qopt-malloc-options={0|1|2|3|4}
- -qopt-calloc
- -q[no-]opt-jump-tables=<arg>
- -qopt-block-factor=<n>
- -qopt-streaming-stores=<keyword>
- -qopt-subscript-in-range

- -q[no-]opt-matmul
- -q[no-]opt-mem-layout-trans[=<level>]
- -q[no-]opt-prefetch[=n]
- -qopt-prefetch-distance=n1[,n2]
- -qopt-threads-per-core=n
- -qopt-streaming-cache-evict=n
- -qopt-gather-scatter-unroll=n
- -qno-opt-gather-scatter-unroll
- -q[no-]opt-dynamic-align
- -qopenmp
- -qopenmp-stubs
- -qopenmp-lib=<ver>
- -qopenmp-link=<library>
- -qopenmp-task=<arg>
- -qopenmp-threadprivate=<ver>
- -q[no-]openmp-simd
- -q[no-]openmp-offload
- -qopt-assume-safe-padding
- -q[no-]offload=<arg>
- -qoffload-attribute-target=<name>
- -qoffload-option,<target>,<tool>,"オプションリスト"
    - o 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ツールとして jit を追加
- -f[no-]fat-lto-objects
- -prof-gen=threadsafe
- -qopt-report[=n]
- -qopt-report-file=[stdout | stderr | <file>]
- -qopt-report-per-object
- -qopt-report-phase=<phase>[,<phase>,...]
- -qopt-report-routine=<name>[,<name>,...]
- -qopt-report-filter=<string>
- -qopt-report-format=[text|vs]
- -qopt-report-names=[mangled|unmangled]
- -qopt-report-help
- -q[no-]opt-report-embed
- -[no-]check-pointers-narrowing
- -no-gcc-include-dir
- -std=gnu++11
- -debug [no]emit-column
- -debug [no]macros
- -fast および -Ofast に -fp-model fast=2 が含まれます
- -f[no-]eliminate-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 -o で始まるコンパイラー・オプションは非推奨 (廃止予定)

-o で始まるすべてのコンパイラー・オプションは非推奨 (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。

3.3.3 -ansi-alias がデフォルトでオン

gcc との互換性を高めるため、-O2 以上で -ansi-alias がデフォルトで有効になります。このオプションが有効な場合、コンパイラーはコードが ANSI 準拠の別名規則に従っていると仮定してコードを生成します。これらの規則に従っていない場合、正しくないコードが生成されることがあります。このオプションを無効にするには、-no-ansi-alias オプションを使用します。

3.3.4 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。

3.3.5 データ・アライメントに関係なく同じコードを実行する -no-opt-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、-no-opt-dynamic-align オプションを使用します。

3.3.6 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、-prof-gen=threadsafe オプションを使用します。

3.3.7 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、-no-check-pointers-narrowing オプションを使用します。

3.4 その他の変更

3.4.1 コンパイラー環境の設定

コンパイラー環境は、`compilervars.sh` スクリプトを使用して設定します。`compilervars.csh` も提供されます。

コマンドの形式は以下のとおりです。

```
source <install-dir>/bin/compilervars.sh argument
```

argument にはターゲット・アーキテクチャーに応じて、`ia32` または `intel64` を指定します。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にビルドする場合は、`intel64` を使用します。コンパイラー環境を設定すると、GNU* GDB (gdb-ia および gdb-mic)、インテル® パフォーマンス・ライブラリー、インテル® Fortran コンパイラー (インストールされている場合) の環境も設定されます。

3.4.2 デフォルトの命令セットがインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) を必要とするものに変更

IA-32 アーキテクチャー向けのコンパイルでは、-msse2 (旧: -xW) がデフォルトです。-msse2 でビルドされたプログラムは、インテル® Pentium® 4 プロセッサーや特定のインテル以外のプロセッサーなど、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) をサポートするプロセッサー上で実行する必要があります。互換性を保証するランタイムチェックは行われません。プログラムがサポートされていないプロセッサーで実行されている場合は、無効な命令フォルトが発生する場合があります。これにより、インテル® SSE 命令が x87 命令の代わりに使用され、高い精度ではなく、宣言された精度で計算が行われることがあるため、浮動小数点結果が変更される可能性があることに注意してください。

すべてのインテル® 64 アーキテクチャー・プロセッサーでインテル® SSE2 がサポートされています。

汎用 IA-32 の以前のデフォルトを使用する場合は、-mia32 を指定してください。

3.4.3 -std=c99 を使用すると "asm" キーワードが認識されない (インテル® C++ コンパイラー 15.0 Linux* 版)

-std=c99 を使用すると、インテル® C++ コンパイラーを開始するときに asm キーワードは認識されません。これは、現在の gcc の動作と同じです。C99 コンパイル時に asm キーワードを認識するようにするには、-std=gnu99 を使用します。

3.5 既知の問題

3.5.1 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

-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。

3.5.2 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

3.5.2.1 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 -offload=mandatory オプションまたは -offload=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、-offload=mandatory コンパイラー・オプションまたは -offload=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。

3.5.2.2 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。

```
-offload-option,mic,compiler,"-z defs"
```

3.5.2.3 コンパイル時の診断の *MIC* タグ

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

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

```
$ icc -c sample.c
sample.c(1): 警告 #1079: *MIC* 関数 "main" の戻り型は "int" でなければなりません。
  void main()
       ^

sample.c(5): 警告 #120: *MIC* 戻り値の型が関数の型と一致しません。
      return 0;
             ^

sample.c(1): 警告 #1079: 関数 "main" の戻り型は "int" でなければなりません。
  void main()
       ^

sample.c(5): 警告 #120: 戻り値の型が関数の型と一致しません。
      return 0;
```

3.5.2.4 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。

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

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) に、/lib 以下のインテル® コンパイラーのランタイム・ライブラリー (例えば、OpenMP* ライブラリー libiomp5.so) が含まれなくなりました。

このため、直接モード (例えば、コプロセッサー・カード上) で OpenMP* アプリケーションを実行する場合は、アプリケーションを実行する前にインテル® MIC アーキテクチャー OpenMP* ライブラリー (<install_dir>/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。

3.5.2.6 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。

3.5.3 インテル® グラフィックス・テクノロジーへのオフロードの既知の問題

3.5.3.1 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。

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

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

回避策を次に示します。

- 各プロセスをシリアルに実行します。
- root として実行します。

3.5.3.3 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

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

  ```
  $ icc -static-intel sample.c

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

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

3.5.5 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。

3.5.6 スタティック解析の既知の問題

3.5.6.1 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。

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 GNU* GDB デバッガー

このセクションでは、インテル® Parallel Studio XE 2015 とともに提供される GNU* GDB 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。

4.1 機能

インテル® Parallel Studio XE 2015 とともに提供される GNU* GDB は、GDB 7.7 を拡張したものです。このデバッガーは、以前のインテル® デバッガーの代わりに提供されています。GDB 7.7 の機能に加えて、次のような新機能が追加されています。

- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーのサポート
- インテル® トランザクショナル・シンクロナイゼーション・エクステンション (インテル® TSX) のサポート
- インテル® Memory Protection Extensions (インテル® MPX) 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 512 (インテル® AVX-512) のレジスターサポート
- データ競合の検出 (*pdbx*):
  POSIX* スレッド (pthread) または OpenMP* モデルを使用してスレッド化されたアプリケーションにおけるデータ競合の検出
- 分岐トレーストア (*btrace*):
  クラッシュ、信号通知、例外などのイベントが発生した後に簡単にバックトラックできるように実行フローで実行された分岐を記録

- ポインターチェッカー: インテル® C++ コンパイラーでポインターチェッカー機能を有効にしてコンパイルされたプログラムでポインターの問題の検出を支援 (詳細は、インテル® C++ コンパイラーのドキュメントを参照してください。)
- インテル® Cilk™ Plus のサポートを強化:
  次のコマンドを使用して、デバッグセッション中にインテル® Cilk™ Plus の並列アプリケーションのシリアル実行をオンまたはオフにすることができます。
  `(gdb) set cilk-serialization [on|off]`

4.2 GNU* GDB の使用

インテル® Parallel Studio XE 2015 とともに提供される GNU* GDB にはいくつかの種類があります。

- IA-32/インテル® 64 デバッガー:
  コマンドラインで `gdb-ia` を使用して IA-32 またはインテル® 64 システム上でアプリケーションをデバッグします。
  グラフィカル・ユーザー・インターフェイスを使用する場合は、標準 Eclipse* IDE インターフェイスを使用できます。
- インテル® Xeon Phi™ コプロセッサー・デバッガー:
  リモートでインテル® Xeon Phi™ コプロセッサー・システム上のアプリケーションをデバッグします。デバッガーはホストシステムで実行され、デバッグ・エージェント (`gdbserver`) がコプロセッサーで実行されます。
  次の 2 つのオプションがあります。
  - `gdb-mic` を使用してコマンドラインでデバッガーを使用します。このオプションは、インテル® Xeon Phi™ コプロセッサーのネイティブ・アプリケーションでのみ利用できます。
    グラフィカル・ユーザー・インターフェイスを使用する場合は、標準 Eclipse* IDE インターフェイスを使用できます。
  - インテル® Parallel Studio XE 2015 に含まれている Eclipse* IDE プラグインを使用します。このオプションは、インテル® Xeon Phi™ コプロセッサーにオフロードされるアプリケーションでのみ利用できます。

GNU* GDB の使用方法は、「ドキュメント」セクションを参照してください。

4.3 ドキュメント

製品とともに提供される GNU* GDB 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/gdb/gdb.pdf
<install-dir>/Documentation/[en_US|ja_JP]/debugger/ ↵
                                          gdb/gdb_quickstart_lin.pdf
```

4.4 既知の問題と変更点

4.4.1 GDB* を Eclipse* IDE で使用する場合の問題

インテルから提供されている GNU* GDB バージョンを Eclipse* IDE で使用する場合、「コンパイラー環境の設定」セクションの説明に従って、Eclipse* を開始する前に環境を設定する必要があります。

4.4.2 オフロード・デバッグ・セッションの安全な終了方法

オフロード・アプリケーション終了時の孤児プロセスや stale デバッガーウィンドウのような問題を回避するには、アプリケーションが終了コードに到達する前にデバッグセッションを手動で終了します。次の手順でデバッグセッションを終了することを推奨します。

- アプリケーションが終了コードに到達する前にデバッグセッションを手動で停止します。
- 最初に、インテル® MIC アーキテクチャー側のデバッガーのツールバーで赤の停止ボタンを押します。アプリケーションのオフロードされている部分が終了します。
- 次に、CPU 側のデバッガーで同じ操作を行います。
- 2 つのデバッガーはリンクされたままです。インテル® MIC アーキテクチャー側のデバッガーはデバッグ・エージェントに接続されています。アプリケーションは CPU 側のデバッガーに (設定されたすべてのブレークポイントを含めて) ロードされています。
- この時点で、両方のデバッガーウィンドウを安全に閉じることができます。

4.4.3 ソース・ディレクトリーの設定によるインテル® MIC アーキテクチャー側のデバッガーのアサーション

GNU* GDB でソース・ディレクトリーを設定すると、アサーションが発生します。

解決方法:
アサーションがデバッガーの操作に影響してはなりません。アサーションを回避するには、ソース・ディレクトリーの設定を使用しないでください。デバッガーがファイルを自動的に特定できない場合、ファイルを指定するようにメッセージが表示されます。

4.4.4 デバッガーから _Cilk_shared 変数へのアクセス

CPU 側の debugee がオフロードされたセクションの共有変数にアクセスする**前に** CPU 側のデバッガーからその変数に書き込みを行うと、書き込んだ値が失われる、間違った値が表示される、アプリケーションがクラッシュするなどの問題が発生します。

次のようなコードスニペットについて考えてみます。

```
_Cilk_shared bool is_active;
_Cilk_shared my_target_func() {
// デバッガーから "is_active" へアクセスすると予期しない結果を招く
// ことがあります (書き込んだ値が失われたり、間違った値が表示される)
is_active = true;
// この時点でデバッガーから "is_active" への (読み取りまたは書き込み)
// アクセスは安全であると見なされます (正しい値が表示される)
}
```

5 Eclipse* 統合

インテル® C++ コンパイラーの Eclipse* 機能と関連プラグイン (インテル® C++ Eclipse* 製品拡張) を Eclipse* 統合開発環境 (IDE) に追加すると、Eclipse* でインテル® C++ コンパイラーがサポートされます。これにより、インテル® C++ コンパイラーを Eclipse* 統合開発環境から使用して、アプリケーションを開発することができます。

5.1 提供されている統合

Eclipse* プラットフォーム用のファイルは次のディレクトリーにあります。

```
<install-dir>/eclipse_support/cdt8.0/eclipse
```

統合には、Eclipse* プラットフォームのバージョン 4.3、4.2、3.8、Eclipse* C/C++ Development Tools (CDT) のバージョン 8.1 以降、および Java* ランタイム環境 (JRE) 6.0 (1.6) Update 11 以降が必要です。

5.1.1 統合に関する注意事項

すでに適切なバージョンの Eclipse*、CDT、および JRE が環境にインストールされ、設定されている場合は、このセクションの「Eclipse でのインテル® C++ Eclipse 製品拡張のインストール方法」で説明するように、インテル® C++ Eclipse* 製品拡張を Eclipse* に追加インストールできます。そうでない場合は、このセクションの「Eclipse*、CDT、および JRE の入手方法とインストール方法」で説明するように、最初に Eclipse*、CDT、および JRE を入手して、インストールしてください。そして、その後にインテル® C++ Eclipse* 製品拡張をインストールします。

Eclipse* に以前のインテル® C++ コンパイラー統合がインストールされている場合、最新の統合はインストールできません。統合がインストールされていない Eclipse* に最新のインテル® C++ コンパイラー統合をインストールする必要があります。同じ理由により、Eclipse* 更新機能を使用して最新のインテル® C++ コンパイラー統合をインストールすることはできません。

5.2 Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法

既存の Eclipse* の構成にインテル® C++ Eclipse* 製品拡張を追加するには、Eclipse* から次の手順を実行します。

[Help (ヘルプ)] > [Install New Software... (新規ソフトウェアのインストール...)] を選択して [Available Software (利用可能なソフトウェア)] ページを開きます。[Add... (追加...)] ボタンをクリックし、[Local... (ローカル...)] を選択します。ディレクトリー・ブラウザーが開きます。インテル® C++ コンパイラーのインストール・ディレクトリーにある `eclipse` ディレクトリーを選択します。例えば、root としてコンパイラーをデフォルトのディレクトリーにインストールした場合は、`/opt/intel/composer_xe_2015.<n>.<xxx>/eclipse_support/cdt8.0/eclipse` を選択します。[OK] をクリックして、ディレクトリー・ブラウザーを閉じます。[OK] をクリックして、[Add Site (サイトの追加)] ダイアログを閉じ、インテル® C++ 統合機能の 2 つのボックスを選択します。1 つめは [Intel® C++ Compiler Documentation (インテル® C++ コンパイラー・ドキュメント)]、2 つめは [Intel® C++ Compiler XE 15.0 for Linux* OS (インテル® C++ コンパイラー XE 15.0 Linux* 版)] です。

注: [Group items by category (項目をカテゴリー別にグループ化)] がオンの場合、インテルの機能は表示されません。インテルの機能を表示するには、このオプションをオフにします。

[Next (次へ)] ボタンをクリックします。[Install (インストール)] ダイアログが表示され、インストールする項目を確認できます。[Next (次へ)] をクリックします。契約に同意するかどうかを確認するメッセージが表示されます。契約に同意したら、[Finish (完了)] をクリックします。署名されていないコンテンツを含むソフトウェアをインストールしようとしていることを示す [Security Warning (セキュリティの警告)] ダイアログが表示されたら [OK] をクリックします。これで、インストールが開始します。

Eclipse* の再起動を求められたら [Yes (はい)] を選択します。Eclipse* が再起動したら、インテル® C++ コンパイラーを使用する CDT プロジェクトを作成して作業することができます。詳細は、インテル® C++ コンパイラーのドキュメントを参照してください。インテル® C++ コンパイラーのドキュメントは、[Help (ヘルプ)] > [Help Contents (ヘルプ目次)] > [Intel(R) C++ Compiler XE 15.0 User and Reference Guides (インテル® C++ コンパイラー XE 15.0 ユーザー・リファレンス・ガイド)] で表示できます。

5.2.1 Eclipse* へのGNU* プロジェクト・デバッガーの統合

「GNU* GDB デバッガー」セクションを参照してください。

5.3 Eclipse*、CDT、および JRE の入手方法とインストール方法

Eclipse* は Java* アプリケーションのため、実行には Java* ランタイム環境 (JRE) が必要です。JRE は、オペレーティング環境 (マシン・アーキテクチャー、オペレーティング・システムなど) に応じてバージョンを選択します。また、多くの JRE の中から選択可能です。

Eclipse* 4.3 および CDT 8.2 の両方が含まれたパッケージは、以下の Web サイトから入手できます。

http://www.eclipse.org/downloads/

スクロールして、"Eclipse IDE for C/C++ Developers" を確認してください。必要に応じて、Linux* 32 ビットまたは Linux* 64 ビットをダウンロードしてください。以前のバージョンは、右のリンクを参照してください。

5.3.1 JRE、Eclipse*、CDT のインストール

適切なバージョンの Eclipse*、CDT、および JRE をダウンロードしたら、次の手順に従ってインストールします。

1. 配布元の手順に従って、JRE をインストールします。
2. Eclipse* をインストールするディレクトリーを作成し、cd でこのディレクトリーに移動します。ここでは、このディレクトリーを <eclipse-install-dir> と表記します。
3. Eclipse* パッケージのバイナリー、.tgz ファイルを <eclipse-install-dir> ディレクトリーにコピーします。
4. .tgz ファイルを展開します。
5. eclipse を起動します。

これで、Eclipse* の構成にインテル® C++ 製品拡張を追加する準備が完了です。追加する方法は、「Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法」のセクションで説明されています。Eclipse の初回起動時の設定については、次のセクションを参照してください。

5.4 インテル® C++ コンパイラーで開発するための Eclipse* の起動

Eclipse* を実行するには JRE が必要なため、Eclipse* を起動する前に JRE が利用可能であることを確認してください。PATH 環境変数の値をシステムにインストールされている JRE の java ファイルのフォルダーへのフルパスに設定するか、Eclipse* コマンドの -vm パラメーターでシステムにインストールされている JRE の java 実行ファイルへのフルパスを指定します。次に例を示します。

```
eclipse -vm /JRE folder/bin/java
```

Eclipse* がインストールされているディレクトリーから Eclipse* 実行ファイルを直接起動します。次に例を示します。

```
<eclipse-install-dir>/eclipse/eclipse
```

5.5 Fedora* システムでのインストール

root アカウントではなくローカルアカウントとして、インテル® C++ コンパイラー Linux* 版を Fedora* 搭載の IA-32 またはインテル® 64 システムにインストールすると、Eclipse* を起動する際に、コンパイラーまたはデバッガーで Eclipse* グラフィカル・ユーザー・インターフェイスが正しく表示されないことがあります。この場合、通常、`JVM Terminated` エラーが表示されます。また、システムレベルの root アカウントでソフトウェアをインストールし、それ以下の権限のユーザーアカウントで実行する場合もエラーが発生します。

これは、Fedora* に実装されているセキュリティーのレベルが低いためです。この新しいセキュリティーは、ダイナミック・ライブラリーなど、システムリソースへのアクセスに悪影響を及ぼすことがあります。一般ユーザーがコンパイラーを使用するためには、システム管理者は *SELinux* セキュリティーを調整する必要があります。

5.6 コンパイラー・バージョンの選択

Eclipse* プロジェクトでは、異なるバージョンのインテル® C++ コンパイラーがインストールされている場合、コンパイラーのバージョンを選択できます。IA-32 アーキテクチャー・システムで統合がサポートされているインテル® コンパイラーのバージョンは、13.0、14.0、15.0 です。インテル® 64 アーキテクチャー・システムで統合がサポートされているインテル® コンパイラーのバージョンは、13.0、14.0、15.0 です。

6 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>/composer_xe_2015.x.xxx/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。

6.1 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。

7 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。

7.1 本バージョンでの変更

7.1.1 インテル® MKL 11.2 Update 1 の新機能

- インテル® MKL for Windows* および Linux* は、現在のインテル® 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® AVX-512 命令のサポートに加えて、次世代のインテル® Xeon® プロセッサー (開発コード名: Skylake) でインテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令をサポートします。
- BLAS:
  - インテル® プロセッサー Skylake (開発コード名) において次の関数を最適化
    - (D/Z)AXPY、(S/D/C/Z)COPY、DTRMM (三角行列が右辺にあり行列の転置がない場合)
  - IA-32 アーキテクチャーとインテル® 64 アーキテクチャー両方でインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) の次のレベル 1 BLAS 関数を最適化
    - (S/D)DOT、(S/D)SCAL、(S/D)ROT、(S/D)ROTM、(S/D/C/Z)SWAP、(S/D/SC/DZ)ASUM
  - インテル® AVX2 において ?GEMM のパフォーマンス (シリアルおよびマルチスレッド) が向上 (IA-32 アーキテクチャー)
  - インテル® AVX およびインテル® AVX2 において beta==0 の場合の ?GEMM のパフォーマンスが向上 (インテル® 64 アーキテクチャー)
  - インテル® AVX において DGEMM のパフォーマンス (シリアルおよびマルチスレッド) が向上 (インテル® 64 アーキテクチャー)
- LAPACK:
  - LAPACK バージョン 3.5 をサポート。このバージョンでは次の新機能を追加。
    - rook ピボット・アルゴリズムを含む対称/エルミート LDLT 因数分解ルーチン
    - 直交列を含む縦長行列と横長行列の 2×1 CSD
  - M>=N で特異ベクトルが必要ないときの (C/Z)GE(SVD/SDD) のパフォーマンスが向上
- FFT:
  - 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、1D バッチ FFT に自動オフロードモードを追加
  - ハイブリッド (OpenMP*+MPI) クラスター FFT のパフォーマンスが向上
  - 大きな 1D 実数-複素数変換の精度が向上
- クラスター用並列直接法スパースソルバー:
  - 同じ並べ替えの多くの因数分解ステップをサポート (maxfct > 1)
- インテル® MKL PARDISO:
  - シュール補行列をサポート (明示的なシュール補行列を得ることおよびシュール補行列により式を解くことを含む)
- スパース BLAS:
  - インテル® プロセッサー Skylake (開発コード名) において SpMV を最適化
  - 行列構造およびインデックスの検証を簡素化する疎行列チェッカー機能をスタンドアロン API として追加 (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「Sparse Matrix Checker Routines」を参照)
  - C/C++ 用スパース BLAS API は定数引数に const 修飾子を使用
- VML:

- 精度動作を制御する新しい環境変数 MKL_VML_MODE を追加。この環境変数は、VML 関数の動作を制御するために使用可能 (vmlSetMode() 関数のアナログ)

7.1.2 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

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。

7.2 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。

8 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.x.xxx/Documentation/<locale>/tbb`) の `CHANGES` というファイルを参照してください。

9 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® デバッガー、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。