

インテル® C++ Composer XE 2013 Linux* 版インストール・ガイド およびリリースノート

資料番号: 321412-004JA

2012 年 9 月 26 日

目次

1	概要.....	5
1.1	変更履歴.....	5
1.1.1	Update 1 (2013.1).....	5
1.1.2	インテル® C++ Composer XE 2011 からの変更点.....	5
1.2	製品の内容.....	6
1.3	動作環境.....	6
1.3.1	IA-64 アーキテクチャー (インテル® Itanium®) 開発のサポートを終了	8
1.4	ドキュメント	8
1.5	サンプル	8
1.6	日本語サポート	8
1.7	テクニカルサポート	9
2	インストール	9
2.1	インテル® Software Manager	9
2.2	インテル® メニー・インテグレートッド・コア (インテル® MIC) プラットフォーム・ソフトウェア・スタック (MPSS) のインストール	10
2.3	クラスターでのインストール.....	10
2.4	サイレント・インストール.....	10
2.5	ライセンスサーバーの使用.....	10
2.6	Eclipse* 統合のインストール.....	10
2.7	既知のインストールの問題.....	10
2.8	インストール先フォルダー.....	11
2.9	削除/アンインストール.....	13
3	インテル® メニー・インテグレートッド・コア (インテル® MIC) アーキテクチャー.....	13
3.1	インテル® Composer XE 2013 Linux* 版のインテル® MIC アーキテクチャー対応について	13
3.2	互換性	14
3.3	はじめに	14
3.4	製品のドキュメント	14

3.5	デバッガー	14
3.6	インテル® マス・カーネル・ライブラリー (インテル® MKL)	14
3.7	注意事項	14
3.7.1	インテル® C++ コンパイラー	14
3.7.2	デバッグとインテル® デバッガー	20
4	インテル® C++ コンパイラー	22
4.1	互換性	22
4.2	新機能と変更された機能	23
4.2.1	インライン・アセンブリと組込み関数での新しいインテル® アーキテクチャー (開発コード名: Broadwell) のサポート (インテル® Composer XE 2013 Update 1)	23
4.2.2	マニュアル CPU ディスパッチに core_4th_gen_avx を追加 (インテル® Composer XE 2013 Update 1)	25
4.2.3	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	25
4.3	新規および変更されたコンパイラー・オプション	26
4.3.1	インテル® Composer XE 2013 の新規および変更されたコンパイラー・オプション	26
4.3.2	-check-pointers=w オプションの追加 (インテル® Composer XE 2013 Update 1)	27
4.3.3	-opt-assume-safe-padding オプションと -opt-streaming-cache-evict and -opt-threads-per-core オプションの追加 (インテル® Composer XE 2013 Update 1)	27
4.3.4	-ipp-link オプション	27
4.3.5	-fimf-domain-exclusion	27
4.4	その他の変更	27
4.4.1	Microsoft* のループプラグマ構文のサポート (インテル® Composer XE 2013 Update 1)	27
4.4.2	インテル® Composer XE 2013 の新しい警告レベル -w3 および警告レベルの変更	27
4.4.3	__regcall 関数および要素関数 (__declspec(vector) など) とのバイナリー互換性の変更	28
4.4.4	乱数ジェネレーター関数のベクトル化用の新しい libirng ライブラリーの追加 (インテル® Composer XE 2013)	28
4.4.5	コンパイラー環境の設定	28
4.4.6	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	28
4.4.7	インテル® Composer XE 2013 の新しい警告レベル -w3 および警告レベルの変更	29
4.4.8	インテル® Cilk™ Plus の “scalar” 節の削除	29
4.4.9	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 (2011 Update 6)	29
4.5	既知の問題	29

4.5.1	インテル® Cilk™ Plus の既知の問題	29
4.5.2	ガイド付き自動並列化の既知の問題	30
4.5.3	スタティック解析の既知の問題	30
5	インテル® デバッガー (IDB)	30
5.1	インテル® デバッガーのサポート終了予定	31
5.2	Java* ランタイム環境の設定	31
5.3	デバッガーの起動	31
5.4	その他のドキュメント	31
5.5	デバッガー機能	32
5.5.1	IDB の主な機能	32
5.5.2	インテル® Inspector XE 2011 Update 6 による IDB の “break into debug” のサ ポート	32
5.6	既知の問題と変更点	32
5.6.1	Pardus* システムのデフォルトの .gdbinit スクリプトでデバッガーがク ラッシュ	32
5.6.2	Pardus* システムでスレッド情報が利用できない	32
5.6.3	Thread Data Sharing Filters (スレッドデータ共有フィルター) が正しく動作し ない	32
5.6.4	コアファイルのデバッグ	32
5.6.5	シェルで \$HOME が設定されていないとデバッガーがクラッシュ	32
5.6.6	コマンドライン・パラメーター -idb と -dbx は未サポート	33
5.6.7	プロセッサのデバッグレジスター (ハードウェア・ベース) を使用した ウォッチポイント (インテル® Composer XE 2011 Update 6)	33
5.6.8	位置独立実行ファイル (PIE) のデバッグは未サポート	33
5.6.9	コマンドライン・パラメーター -parallel は未サポート	33
5.6.10	[Signals (シグナル)] ダイアログが動作しない	34
5.6.11	GUI のサイズ調整	34
5.6.12	\$cdir ディレクトリー、\$cwd ディレクトリー	34
5.6.13	info stack の使用	34
5.6.14	\$stepg0 のデフォルト値の変更	34
5.6.15	一部の Linux* システムでの SIGTRAP エラー	34
5.6.16	MPI プロセスのデバッグには idb GUI は使用不可	35
5.6.17	GUI でのスレッド同期ポイントの作成	35
5.6.18	[Data Breakpoint (データ・ブレイクポイント)] ダイアログ	35
5.6.19	IA-32 アーキテクチャー向けのスタック・アライメント	35
5.6.20	GNOME 環境の問題	35
5.6.21	オンラインヘルプへのアクセス	35
6	Eclipse* 統合	35
6.1	提供されている統合	35

6.1.1	統合に関する注意事項.....	36
6.2	Eclipse* でのインテル® C++ Eclipse* 製品拡張のインストール方法.....	36
6.2.1	Eclipse* へのインテル® デバッガーの統合.....	37
6.3	Eclipse*、CDT、および JRE の入手方法とインストール方法.....	37
6.3.1	JRE、Eclipse*、CDT のインストール.....	37
6.4	インテル® C++ コンパイラーで開発するための Eclipse* の起動.....	38
6.5	Fedora* システムでのインストール.....	38
6.6	コンパイラー・バージョンの選択.....	38
7	インテル® インテグレートッド・パフォーマンス・プリミティブ.....	39
7.1	別途ダウンロード可能なインテル® IPP スタティック・スレッド・ライブラリー.....	39
7.2	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.....	39
7.3	インテル® IPP コードサンプル.....	39
8	インテル® マス・カーネル・ライブラリー.....	39
8.1	注意事項.....	39
8.2	本バージョンでの変更.....	40
8.2.1	インテル® MKL 11.0 Update 1 の新機能.....	40
8.2.2	最初のリリースでの変更.....	40
8.3	権利の帰属.....	41
9	インテル® スレッディング・ビルディング・ブロック.....	42
10	著作権と商標について.....	42

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。

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 (2013.1)

- インテル® C++ コンパイラー XE 13.0.1
- インテル® デバッガー 13.0.1
- [インテル® マス・カーネル・ライブラリー 11.0 Update 1](#)
- インテル® インテグレートッド・パフォーマンス・プリミティブ 7.1 Update 1
- インテル® スレディング・ビルディング・ブロック 4.1 Update 1
- [Eclipse* プラットフォームのバージョン 4.2 および 3.8、Eclipse CDT 8.1 のサポート](#)
- [インライン・アセンブリと組み込み関数での新しいインテル® アーキテクチャー \(開発コード名: Broadwell\) のサポート](#)
- [-opt-streaming-cache-evict および -opt-threads-per-core のサポート \(インテル® メニー・インテグレートッド・コア \(インテル® MIC\) アーキテクチャー用\)](#)
- [-check-pointers=w](#)
- [Microsoft* ループプラグマのサポート](#)
- 報告された問題の修正

1.1.2 インテル® C++ Composer XE 2011 からの変更点

- インテル® メニー・インテグレートッド・コア (インテル® MIC) アーキテクチャーのコプロセッサに作業をオフロードするアプリケーション、またはコプロセッサ上でネイティブに実行するアプリケーションの開発がサポートされました。詳細は、「[インテル® メニー・インテグレートッド・コア \(インテル® MIC\) アーキテクチャー](#)」セクションを参照してください。
- インテル® C++ コンパイラーがバージョン 13.0 にアップデート
- インテル® デバッガーがバージョン 13.0 にアップデート
 - [インテル® デバッガーのサポート終了について](#)
- [インテル® マス・カーネル・ライブラリーがバージョン 11.0 にアップデート](#)
 - インテル® Pentium® III プロセッサのサポートが終了。詳細は、[削除された機能に関するナレッジベースの記事](#) (英語) を参照してください。
- インテル® インテグレートッド・パフォーマンス・プリミティブがバージョン 7.1 にアップデート
 - [インテル® IPP スタティック・スレッド・ライブラリーを別のパッケージで提供](#)
- [インテル® スレディング・ビルディング・ブロックがバージョン 4.1 にアップデート](#)
- Fedora* 17、SUSE* Linux* Enterprise Server 11 SP2、Ubuntu* 11.10 および Ubuntu* 12.04 のサポート
- 次のバージョンの Linux* ディストリビューションのサポートを終了:
 - Red Hat* Enterprise Linux* 4
 - SUSE* Linux* Enterprise Server 11 SP1
 - Fedora* 15
 - Ubuntu* 11.04
 - Ubuntu* 10.04
 - Asianux*

- 製品のアップデートとライセンスのアクティベーションを管理する [インテル® Software Manager](#) の追加
- [新しい C++11 機能](#)
- [将来のインテル® プロセッサに対するサポートの強化](#)
- [範囲外のメモリーチェック](#)
- [インテル® Composer XE 2013 の新しい警告レベル -w3 および警告レベルの変更](#)
- [スタティック解析の改良](#)

1.2 製品の内容

インテル® C++ Composer XE 2013 Update 1 Linux* 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 13.0.1。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、およびインテル® メニー・インテグレートッド・コア (インテル® MIC) 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® デバッガー 13.0.1
- インテル® インテグレートッド・パフォーマンス・プリミティブ 7.1 Update 1
- インテル® マス・カーネル・ライブラリー 11.0 Update 1
- インテル® スレディング・ビルディング・ブロック 4.1 Update 1
- Eclipse* 開発環境への統合
- 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 プロセッサ以降、または互換性のあるインテル以外のプロセッサ)
 - 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™ プロセッサ
 - [インテル® MIC プラットフォーム・ソフトウェア・スタック \(MPSS\)](#)
- 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・システムの使用を推奨します。
- RAM 1GB (2GB 推奨)
- 2.5GB 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* 17
 - Red Hat* Enterprise Linux* 5、6

- SUSE* Linux* Enterprise Server 10、11 SP2
- Ubuntu* 11.10、12.04
- Debian* 6.0
- インテル® Cluster Ready
- Pardus* 2011.2 (x64 のみ)
- 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)
- -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。

インテル® デバッガーのグラフィカル・ユーザー・インターフェイスを使用するためのその他の要件

- Java* ランタイム環境 (JRE) 6.0 (1.6†) - 5.0 推奨
 - IA-32 アーキテクチャー・システムでは 32 ビット版の JRE、インテル® 64 アーキテクチャー・システムでは 64 ビット版の JRE を使用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 4.2 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
- Eclipse* Platform 3.7 および次の両方
 - Eclipse* C/C++ Development Tools (CDT) 8.0 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 (例: インテル® Pentium® 4 プロセッサ) 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® インテグレートッド・パフォーマンス・プリミティブまたはインテル® スレディング・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・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・

モデルでも動作します。特定のプロセッサ・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。

1.3.1 IA-64 アーキテクチャー (インテル® Itanium®) 開発のサポートを終了

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。

1.4 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。

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向けには、インテル製マイクロプロセッサ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固有の最適化は、インテル製マイクロプロセッサ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。

日本語サポート版は、インテル® C++ Composer XE 2013 のアップデートとして提供されます。

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

1.7 テクニカルサポート

インストール時に製品の登録を行わなかった場合は、[インテル® ソフトウェア開発製品レジストレーション・センター](#)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。

DVD 版を購入した場合は、DVD をドライブに挿入し、DVD のトップレベル・ディレクトリーにディレクトリーを変更 (cd) して、次のコマンドでインストールを開始します。

```
./install.sh
```

ダウンロード版を購入した場合は、次のコマンドを使用して、書き込み可能な任意のディレクトリーに展開します。

```
tar -xzf 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。

2.1 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。

2.2 インテル® メニー・インテグレートッド・コア (インテル® MIC) プラットフォーム・ソフトウェア・スタック (MPSS) のインストール

インテル® MIC プラットフォーム・ソフトウェア・スタック (MPSS) は、インテル® Composer XE 2013 Linux* 版 (インテル® MIC アーキテクチャー対応) 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。

ユーザー空間およびカーネルドライバのインストールに必要な手順については、インテル® MIC プラットフォーム・ソフトウェア・スタック (MPSS) のドキュメントを参照してください。

2.3 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® Cluster Studio 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。

2.4 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。

2.5 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。

2.6 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。

2.7 既知のインストールの問題

- Linux* ディストリビューションの Security-Enhanced Linux (SELinux) 機能を有効にしている場合は、インテル® C++ コンパイラーをインストールする前に SELINUX モードを `permissive` に変更する必要があります。詳細は、Linux* ディストリビューションのドキュメントを参照してください。インストールが完了したら、SELINUX モードを元の値に戻してください。
- 一部の Linux* バージョンでは、自動マウントデバイスに "実行" 許可がなく、インストール・スクリプトを直接 DVD から実行すると、次のようなエラーメッセージ

が表示されることがあります。

```
bash: ./install.sh:/bin/bash: bad interpreter:Permission
denied
```

このエラーが表示された場合は、次の例のように実行許可を含めて DVD を再マウントします。

```
mount /media/<dvd_label> -o remount,exec
```

その後、再度インストールを行ってください。

- 「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 Ubuntu*、Debian* または Pardus* 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、Ubuntu*、Debian* または Pardus* 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。Ubuntu*、Debian* または Pardus* 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。

2.8 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。

本リリースではディレクトリー構成がインテル® コンパイラー 11.1 から変更されています。

インテル® C++ Composer XE 2011 の以前のリリースとインテル® C++ Composer XE 2013 では、トップレベルのインストール・ディレクトリーが異なりますが、引き続き composerxe シンボリック・リンクを使用して最新の製品インストールを参照することができます。

<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® インテグレートッド・パフォーマンス・プリミティブ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® マス・カーネル・ライブラリー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® スレディング・ビルディング・ブロックのディレクトリーへのシンボリック・リンク

- `composerxe - composer_xe_2013` ディレクトリーへのシンボリック・リンク
- `composer_xe_2013` - インストールされている最新バージョンのインテル® Composer XE 2013 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_2013.<n>.<pkg>` - 特定のリリース番号のファイルが含まれている物理ディレクトリー。`<n>` はリリース番号、`<pkg>` はパッケージビルド ID。

各 `composer_xe_2013` 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® Composer XE 2013 コンパイラー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
- `mk1` - `mk1` 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_2013.<n>.<pkg>` ディレクトリーには、特定のリリース番号のインテル® Composer XE 2013 コンパイラー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® インテグレートッド・パフォーマンス・プリミティブのライブラリーとヘッダーファイル
- `mk1` - インテル® マス・カーネル・ライブラリーのライブラリーとヘッダーファイル
- `tbb` - インテル® スレッディング・ビルディング・ブロック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
- `.scripts` - インストール用スクリプト

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リース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/リリース番号のインテル® Composer XE 2013 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。このディレクトリー構成は、将来のリリースでも保持される予定です。

2.9 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>/bin/uninstall.sh`
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 13.0 for Linux* OS (インテル (R) C++ Compiler XE 12.0 Linux* OS 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® メニー・インテグレートッド・コア (インテル® MIC) アーキテクチャー

このセクションでは、インテル® Composer XE 2013 Linux* 版 (インテル® MIC アーキテクチャー対応) の変更点、新機能、および最新情報をまとめています。

3.1 インテル® Composer XE 2013 Linux* 版のインテル® MIC アーキテクチャー対応について

インテル® Composer XE 2013 Linux* 版 (インテル® MIC アーキテクチャー対応) は、インテル® Xeon Phi™ コプロセッサで実行するコードの事前定義済みセクションを有効にすることにより、インテル® C++ Composer XE 2013 とインテル® Fortran Composer XE 2013 の機能セットを拡張します。

コプロセッサが利用可能な場合、コードのこれらのセクションはコプロセッサで実行されます。コプロセッサが利用できない場合は、ホスト CPU で実行されます。

本リリースノートでは、オフロード操作のターゲットについて、*コプロセッサ*と*ターゲット*という2つの用語を使用しています。

現在、インテル® Composer XE 2013 の次のコンポーネントでインテル® MIC アーキテクチャーをサポートしています。

- インテル® C++ コンパイラーおよびインテル® Fortran コンパイラー
- インテル® デバッガー (インテル® IDB)
- インテル® マス・カーネル・ライブラリー (インテル® MKL)
- インテル® スレディング・ビルディング・ブロック (インテル® TBB)
- Eclipse* IDE 統合

3.2 互換性

本リリースでは、インテル® Xeon Phi™ コプロセッサがサポートされました。詳細は、「[テクニカルサポート](#)」セクションを参照してください。

このプロセッサの A0 ステッピングをサポートするコードを生成する必要がある場合は、「[-opt-streaming-stores never の使用についての注意](#)」を参照してください。

3.3 はじめに

インテル® 64 アーキテクチャー用とインテル® MIC アーキテクチャー用のコードは同じコンパイラを使用して生成します。インテル® 64 アーキテクチャー用に環境を設定する場合は、「[コンパイラ環境の設定](#)」セクションを参照してください。詳細は、「[注意事項](#)」セクションを参照してください。

3.4 製品のドキュメント

インテル® Composer XE 2013 のインテル® MIC アーキテクチャーに関連するドキュメントには現在も修正が加えられています。ドキュメントの最新の情報については、Web サイト <http://intel.ly/MxPFYx> (英語) を参照してください。

3.5 デバッガー

グラフィカルなデバッガーを使用するには、Eclipse* 統合で idb_mpm デバッガー・スタートアップ・スクリプト (bin/intel64_mic) を使用するように設定します。インテル® MIC アーキテクチャーで実行しているコードにアタッチするか、CPU からオフロードされたコードをデバッグできます。

SSH 接続でリモートシステムのデバッガーを使用するには、ローカルの X ディスプレイで SSH 表示によるラグが最小限になるように DISPLAY 環境変数を設定する必要があります。

パッケージには、これらのデバッガーのコマンドライン・バージョンも含まれています。名前はそれぞれ、idbc (インテル® 64 アーキテクチャーのホスト向け) および idbc_mic (インテル® MIC アーキテクチャーのターゲット向け) となります。

コマンドライン・バージョンのデバッガーでは自動アタッチ機能はサポートされないことに注意してください。

3.6 インテル® マス・カーネル・ライブラリー (インテル® MKL)

インテル® MIC のサポートについての詳細は、「[インテル® MKL](#)」セクションを参照してください。

3.7 注意事項

3.7.1 インテル® C++ コンパイラ

3.7.1.1 デフォルトのコード生成でインテル® Xeon Phi™ コプロセッサ A0 ステッピングのサポートを削除 (インテル® Composer XE 2013 Update 1)

インテル® Composer XE 2013 Update 1 は、インテル® Xeon Phi™ コプロセッサ B0 ステッピングで導入された新しいストリーミング・ストア命令をデフォルトで生成します。これらの命令は A0 ステッピングではサポートされていないため、ランタイムエラーが発生します。アプリケーションを A0 ステッピングで実行する必要がある場合は、`-opt-streaming-stores never` オプションを指定して、これらの命令が生成されないようにしてください。このオプションを指定すると、B0 以降のステッピングではパフォーマンスが低下することに注意してください。

3.7.1.2 *-opt-assume-safe-padding* オプション (インテル® Composer XE 2013 Update 1)

デフォルトでは、コンパイラーは、必要に応じてパディングを挿入できる場合でも、メモリーのオブジェクトのサイズがプログラムで記述されたサイズよりも大きいと仮定すべきではありません。-opt-assume-safe-padding オプションが指定されると、コンパイラーは、動的に割り当てられるメモリーがパディングされ、プログラムのソースで指定されたサイズを超えても 64 バイトまでのアクセスでは問題が発生しないことを仮定します。このオプションは、一部のインテル® MIC 命令で要求される安全性を緩和してより高速なコードを生成するために、最適化で使用するオブジェクト境界を多少超えてアクセスしても安全であることをプログラマーが宣言できるように追加されました。このオプションを指定しても、コンパイラーはスタティックな自動オブジェクトのパディングを行いません。パディングを行うのはプログラマーの責任です。このオプションは、コンパイラーが安全に行える仮定を変更するだけです。デフォルトは無効 (-no-opt-assume-safe-padding) です。

3.7.1.3 *-opt-streaming-cache-evict* オプションと *-opt-threads-per-core* オプション (インテル® Composer XE 2013 Update 1)

-opt-streaming-cache-evict オプションは、インテル® MIC アーキテクチャーでストリーミング・ロード/ストアが使用されたときのキャッシュ退避レベルを指定します。

-opt-streaming-cache-evict =0

キャッシュ退避命令は生成されません。

-opt-streaming-cache-evict =1

ストリーミング・ロード/ストア後に L1 キャッシュ退避命令が生成されます。

-opt-streaming-cache-evict =2

ストリーミング・ロード/ストア後に L2 キャッシュ退避命令が生成されます。

-opt-streaming-cache-evict =3

ストリーミング・ロード/ストア後に L1 および L2 キャッシュ退避命令が生成されます。

-opt-threads-per-core オプションは、アプリケーションで使用するインテル® MIC アーキテクチャーのコアあたりのハードウェア・スレッド数を指定します。指定できるスレッド数は、1 から 4 までです。

3.7.1.4 新しい *-fimf-domain-exclusion* コンパイラー・オプション

次のコンパイラー・オプションの情報がドキュメントから削除されました。

fimf-domain-exclusion

関数が評価されたドメインを示します。

相当する IDE オプション

なし

アーキテクチャー

インテル® MIC アーキテクチャー向けのインテル® 64 アーキテクチャー

構文

Linux*: `-fimf-domain-exclusion=classList[:funcList]`

OS X*: なし

Windows*: なし

引数

classList 次の項目のカンマ区切りのリスト。

- 1 つ以上の浮動小数点値クラス名。プログラムの正当性に影響を与えることなく関数ドメインから除外できます。
- 1 つの簡略表記トークン。

クラス名は次のとおりです。

- extremes
- nans
- infinities
- denormals
- zeros

簡略表記トークンは次のとおりです。

- none: すべてのクラス名をドメインから除外しません。
- all: すべてのクラス名をドメインから除外します。
- common: extremes,nans,infinities,denormals と同じです。

クラストークンの順序は任意です。各トークンを 2 回以上指定できます。

funcList 関数名のカンマ句区切りのリスト。

この引数を指定しない場合、すべての数値演算ライブラリー関数にドメインの制約が適用されます。

デフォルト

なし

説明

関数が評価されたドメインを示し、*funcList* で指定された関数が数のクラスで標準に準拠した結果を生成しない場合、プログラムが正しく動作することを明示します。

3.7.1.5 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"`

3.7.1.6 メモリー割り当てのパフォーマンスのチューニング

この説明は、インテル® C++ Composer XE 2013 ユーザーガイドの同様の説明を更新したものです。

コプロセッサのユーザー割り当てデータでは、`malloc` や `_mm_malloc` の代わりに `mmap()` で大きな (2MB の) ページ割り当てを使用すると、アプリケーションのパフォーマンスが向上する場合があります。

すべてのアプリケーションで大きなページサイズを使用するメリットがあるわけではありません。一般に、大きなページサイズがパフォーマンスに影響するかどうかはデータ・アクセス・パターンに大きく依存します。アプリケーションが異なるページに割り当てられた複数のデータ構造にアクセスする場合、コプロセッサの 2MB ページに限られた TLB (トランслэшヨン・ルックアサイド・バッファ) エントリーしかない場合、パフォーマンスの低下を引き起こします。

`malloc` および `_mm_malloc` のデフォルトのページサイズは 4KB です。アプリケーションで 2MB ページを使用するには、下記の方法を使用します。

```
#include <string.h>
#include <sys/mman.h>
#include <stdio.h>
#include <errno.h>

#define TWO_MB (2*1024*1024ULL)
#define MAP_HUGETLB 0x40000 /* ヒュージ・ページ・マッピングを作成 */

/* ヒュージ・ページ・サポートを使用してメモリーを割り当て */
#define MALLOC_2M(size) \
    mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_ANONYMOUS | MAP_SHARED | \
        MAP_HUGETLB | MAP_POPULATE, -1, 0)

/* 割り当てたメモリーを解放 */
#define FREE_2M(addr, size) munmap(addr, (size + TWO_MB & ~(TWO_MB-1)))

/**
 * allocate_huge_pages - 2MB ページサポートを使用してメモリーを割り当て
 * @size - 割り当てるメモリーのサイズ
 */

static inline void* allocate_huge_pages(size_t size)
{
    size_t sz = size + TWO_MB & ~(TWO_MB-1);
    void* mem = MALLOC_2M(sz);
    if (mem == MAP_FAILED)
        printf("mmap allocation failed\n");
    return mem;
}
```

3.7.1.7 `-opt-prefetch-distance` コンパイラー・オプション

構文:

`-opt-prefetch-distance=n1[,n2]`

`n1` ≥ 0 (負ではない整数値)、`n2` も同様。`n2` はオプションです。`n2` が指定された場合、`n1` $\geq n2$ でなければなりません。

使用例:

```
-opt-prefetch-distance=64,32
-opt-prefetch-distance=16,2
-opt-prefetch-distance=24
```

説明:

このオプションは、パフォーマンス・チューニングに使用します。このオプションを使用して、ユーザーはプリフェッチの距離を指定できます。単位は、ループ内部のコンパイラで生成されるプリフェッチで使用する (ベクトル化された) 反復回数です。`n1` の値は `vprefetch1` 命令 (メモリーから L2 キャッシュへのプリフェッチ) の距離として使用され、`n2` の値 (指定された場合) は `vprefetch0` 命令 (L2 キャッシュから L1 キャッシュへのプリフェッチ) の距離として使用されます。このオプションが使用されなかった場合、プリフェッチの距離はコンパイラのヒューリスティックに基づいて決定されます。インテル® MIC アーキテクチャーでは `-opt-prefetch` オプションがデフォルトでオンになっていることに注意してください。

`-opt-prefetch=0` が指定されると、このオプションは無視されます。

3.7.1.8 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* タグが追加されるようになりました。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 MIC 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;
```

3.7.1.9 ランタイム型情報 (RTTI) は未サポート

仮想共有メモリ・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。

3.7.1.10 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への転送

インテル® MIC プラットフォーム・ソフトウェア・スタック (MPSS) に、`/lib` 以下のインテル® コンパイラのランタイム・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・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・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。

3.7.1.11 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。

3.7.1.12 オフロードの動作を制御する環境変数

オフロードの動作を制御する環境変数が追加されました。

3.7.1.12.1 MIC_ENV_PREFIX

環境変数の値を各インテル® Xeon Phi™ コプロセッサに渡す一般的な方法です。

`MIC_ENV_PREFIX` には、コプロセッサを示す環境変数として使用する値を設定します。次に例を示します。

```
setenv MIC_ENV_PREFIX MYCARDS
```

は特定のコプロセッサを示す環境変数として文字列 “MYCARDS” を使用します。

```
<mic-prefix>_<var>=<value>
```

形式で環境変数を設定すると、各コプロセッサに `<var>=<value>` が渡されます。

```
<mic-prefix>_<card-number>_<var>=<value>
```

形式で環境変数を設定すると、コプロセッサ `<card-number>` に `<var>=<value>` が渡されます。

```
<mic-prefix>_ENV=<variable1=value1|variable2=value2>
```

形式で環境変数を設定すると、各コプロセッサに <variable1>=<value1> および <variable2>=<value2> が渡されます。

```
<mic-prefix>_<card-number>_ENV=<variable1=value1|variable2=value2>
```

形式で環境変数を設定すると、コプロセッサ <card-number> に <variable1>=<value1> および <variable2>=<value2> が渡されます。

例:

```
setenv MIC_ENV_PREFIX PHI // 使用するプリフィックスを定義
setenv PHI_ABCD abcd      // すべてのコプロセッサで ABCD=abcd を設定
setenv PHI_2_EFGH efgh    // コプロセッサ 2 で EFGH=efgh を設定
setenv PHI_VAR X=x|Y=y    // すべてのコプロセッサで X=x と Y=y を設定
setenv PHI_4_VAR P=p|Q=q  // コプロセッサ 4 で P=p と Q=q を設定
```

3.7.1.12.2 MIC_USE_2MB_BUFFERS

ラージページを使用してバッファを作成するしきい値を設定します。バッファのサイズがしきい値を超えると、バッファはラージページを使用して作成されます。

例:

```
// コプロセッサでサイズが 100KB 以上の変数を割り当てると
// ラージページで割り当てられます
setenv MIC_USE_2MB_BUFFERS 100k
```

3.7.1.12.3 MIC_STACKSIZE

アプリケーションで使用されるすべてのインテル® Xeon Phi™ コプロセッサのオフロードプロセスのスタックサイズを設定します。この環境変数はスタックサイズ全体を設定します。各 OpenMP* スレッドのサイズを変更するには、MIC_OMP_STACKSIZE を使用します。

例:

```
setenv MIC_STACKSIZE 100M // MIC スタックを 100MB に設定
```

3.7.1.12.4 OFFLOAD_DEVICES

OFFLOAD_DEVICES は、変数の値として指定されたコプロセッサのみを使用するようにプロセスを制限します。<value> は物理デバイスの番号のカンマ区切りのリストです。番号は 0 から (システムの物理デバイスの数-1) の範囲になります。オフロードに利用できるデバイスの番号は論理的に付けられます。_Offload_number_of_devices() は許可されたデバイスの番号を返します。offload 宣言子の target 指定子で指定されるデバイスのインデックスは 0 から (許可されたデバイスの数-1) の範囲になります。

例

```
setenv OFFLOAD_DEVICES "1,2"
```

3.7.2 デバッグとインテル® デバッガー

3.7.2.1 MPSS Alpha 2 (2.1.3126-x) でインテル® デバッガーを使用する場合の制限

MPSS Alpha 2 (2.1.3126-x) でインテル® MIC アーキテクチャ対応インテル® デバッガーを使用する場合は、次の制限が適用されます。

- インテル® デバッガーの Eclipse* CDT 統合は、コプロセッサで実行するネイティブ・アプリケーションで "プロセスにアタッチ" をサポートしていません。

- コマンドラインでネイティブ・コプロセッサ・アプリケーションをデバッグすると、リモート・デバッグ・エージェント `idbserver_mic` がアップロードされ、`scp/ssh` を使用して開始されます。このとき、`idbc_mic` を開始するために使用するユーザー ID がコプロセッサ・カードにも存在していると想定されます。このユーザー ID がパスワードなしで認証されるように設定されていない限り、`scp` および `ssh` でパスワードを入力する必要があります。
- コマンドラインでネイティブ・コプロセッサ・アプリケーションをデバッグする場合、コプロセッサに自動的にアップロードされていた共有ライブラリー `libmyodbl-service.so` を手動でアップロードする必要があります。

ファイルがブート時にアップロードされるように、オーバーレイを作成できます。MPSS の `readme.txt` (インテル® プレミアサポートの “Intel® SDP MAKCl Family” 製品から利用できます) に記述されているオーバーレイの使用法の説明に従ってください。この特定のオーバーレイを実装するために必要な手順は、次のようになります。

- 次の内容で `/etc/sysconfig/mic/conf.d/myo.conf` を作成します。

```
# MYO download files
Overlay /
/opt/intel/mic/myo/config/myo.filelist
```
 - 次の内容で `/opt/intel/mic/myo/config/myo.filelist` を作成します。

```
dir /lib64 755 0 0
file /lib64/libmyodbl-service.so
opt/intel/mic/myo/lib/libmyodbl-service.so 755 0 0
```
- コマンドラインでヘテロジニアス・アプリケーションをデバッグすると、オフロードプロセスが `root` として開始されます。`root` 以外のユーザー ID で `idbc_mic` を使用すると、リモート・デバッグ・サーバー `idbserver_mic` でオフロードプロセスを確認できません。この問題を回避するには、コマンドライン・デバッガー `idbc_mic` を `root` として起動します。または、デフォルトの起動オプションに `-mpm-launch=1 -mpm-cardid=<card-id>` オプションを追加します。

```
idbc_mic -mpm-launch=1 -mpm-cardid=<card-id> -tco -
rconnect=tcipip:<cardip>:<port>
```
 - Eclipse* からヘテロジニアス・アプリケーションをデバッグすると、オフロードプロセスを作成するときにエラー “オフロードエラー: デバイス 0 のプロセスを開始できません (エラーコード 1)” が表示されます。この問題を回避するには、オフロードプロセスが正常に作成されるまでデバッグセッションを再起動してください。

3.7.2.2 Eclipse* で debuggee のコマンドライン引数の設定に失敗する

GDB モードで `file` コマンドを使用してアプリケーションをロードした場合、デバッガーは Eclipse* で debuggee のコマンドライン引数を正しく設定できません。debuggee は次のメッセージを出力してアバートします。

```
*** abort -internal failure : get_command_argumentfailed
```

この場合、IDB のコマンドライン引数に実行ファイルを追加します。

3.7.2.3 Eclipse* でローカル変数の表示に失敗する

アプリケーションのデバッグ中は Eclipse* 環境でローカル変数を表示できません。

回避方法: Eclipse* の Expression ビューにローカル変数を入力して値を取得します。

3.7.2.4 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、MIC 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。MIC 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。

3.7.2.5 ソース・ディレクトリーの設定による MIC 側のデバッガーのアサーション

ABR デバッガーでソース・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。

3.7.2.6 デバッガーから `_Cilk_shared` 変数へのアクセス

CPU 側の debuggee 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)
}
```

4 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しています。

4.1 互換性

バージョン 11.0 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。

4.2 新機能と変更された機能

インテル® C++ Composer XE 2013 には、インテル® C++ コンパイラー XE 13.0 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、ドキュメントを参照してください。

- オフロード (プラグマ/キーワード) とネイティブ (-mmic) コンパイルの両方でインテル® MIC アーキテクチャーをサポート
 - オフロードの言語拡張
 - データ・マーシャリング (非共有メモリー) 手法のオフロード拡張
 - 仮想共有メモリー手法のオフロード拡張
- 第3世代インテル® Core™ プロセッサー・ファミリー (-xCORE-AVX-I および -axCORE-AVX-I) とインテル® アドバンスド・ベクトル・エクステンション (インテル® AVX) 対応の将来のインテル® プロセッサー (-xCORE-AVX2 および -axCORE-AVX2) のサポートを強化
- C++11 の機能 (-std=c++0x)
 - 追加の型特性
 - 統一的な初期化構文
 - 汎用化された定数式 (一部サポート)
 - noexcept
 - 範囲に基づく for ループ
 - ラムダから関数ポインターへの変換
 - 暗黙の移動コンストラクターと移動代入演算子
 - gcc 4.6/4.7 ヘッダーの C++ 機能をサポート
- 範囲外のメモリーチェック (-check-pointers)
- スタティック解析の解析を制御する追加オプション (-diag-enable sc-{full|concise|precise})

4.2.1 インライン・アセンブリーと組込み関数での新しいインテル® アーキテクチャー (開発コード名: Broadwell) のサポート (インテル® Composer XE 2013 Update 1)

新しいインテル® アーキテクチャー (開発コード名: Broadwell) の一部の新しい命令がサポートされました。インテル® Composer XE 2013 Update 1 では、インライン・アセンブリーと組込み関数でこれらの命令をサポートします。組込み関数は、`immintrin.h` に定義されています。

```
extern int _rdseed16_step(unsigned short *random_val);
extern int _rdseed32_step(unsigned int *random_val);
extern int _rdseed64_step(unsigned __int64 *random_val);
```

これらの組込み関数は、16/32/64 ビットの乱数整数を生成します。これらの組込み関数は、ハードウェア命令 RDSEED にマップされます。生成された乱数は指定されたメモリーの場所へ書き込まれ、ステータス (ハードウェアによって有効な乱数が返された場合は -1、そうでない場合は 0) が返されます。

`rdseed` と `rdrand` 組込み関数の違いは、`rdseed` が NIST SP 800-90B および NIST SP 800-90C 標準に準拠し、`rdrand` が NIST SP 800-90A 標準に準拠することです。

```
extern unsigned char _addcarry_u32(unsigned char c_in, unsigned int
src1, unsigned int src2, unsigned int *sum_out);

extern unsigned char _addcarry_u64(unsigned char c_in, unsigned
__int64 src1, unsigned __int64 src2, unsigned __int64 *sum_out);
```

これらの組込み関数は、2つの 32/64 ビット整数値 (src1, src2) とキャリーイン値の合計を計算します。キャリーイン値は、c_in 入力値が非ゼロの場合は 1、その他の場合は 0 と見なされます。計算の結果は、sum_out 引数で指定したメモリー位置に格納されます。

```
*sum_out = src1 + src2 + (c_in !=0 ?1 :0)
```

この組込み関数は sum_out で指定されたメモリーアドレスの有効性チェックは行わないため、合計の結果を格納しないでキャリーアウトが発生したかどうか確認するために使用できません。組込み関数の戻り値は、合計により生成されるキャリーアウト値です。計算の結果は、sum_out 引数で指定したメモリー位置に格納されます。

```
extern unsigned char _subborrow_u32(unsigned char b_in, unsigned int
src1, unsigned int src2, unsigned int *diff_out);
```

```
extern unsigned char _subborrow_u64(unsigned char b_in, unsigned
__int64 src1, unsigned __int64 src2, unsigned __int64 *diff_out);
```

これらの組込み関数は、32/64 ビットの符号なし整数値 src2 とボローイン値の合計を計算して、合計の結果を 32/64 ビットの符号なし整数値 src1 から引きます。ボローイン値は、c_in 入力値が非ゼロの場合は 1、その他の場合は 0 と見なされます。計算の結果は、diff_out 引数で指定したメモリー位置に格納されます。

```
*diff_out = src1 + (src2 + (b_in !=0 ?1 :0))
```

この組込み関数は diff_out で指定されたメモリーアドレスの有効性チェックは行わないため、減算の結果を格納しないでボローアウトが発生したかどうか確認するために使用できません。組込み関数の戻り値は、減算により生成されるボローアウト値です。減算の結果は、diff_out 引数で指定したメモリー位置に格納されます。

```
extern unsigned char _addcarryx_u32(unsigned char c_in, unsigned int
src1, unsigned int src2, unsigned int *sum_out);
```

```
extern unsigned char _addcarryx_u64(unsigned char c_in, unsigned
__int64 src1, unsigned __int64 src2, unsigned __int64 *sum_out);
```

これらの組込み関数は、2つの 32/64 ビット整数値 (src1, src2) とキャリーイン値の合計を計算します。キャリーイン値は、c_in 入力値が非ゼロの場合は 1、その他の場合は 0 と見なされます。合計は、sum_out 引数で参照されるメモリー位置に格納されます。

```
*sum_out = src1 + src2 + (c_in !=0 ?1 :0)
```

この組込み関数は sum_out で指定されたメモリーアドレスの有効性チェックは行わないため、合計の結果を格納しないでキャリーアウトが発生したかどうか確認するために使用できません。この組込み関数は、コンパイラの決定に応じて ADCX または ADOX 組込み関数に変換されます。これらの組込み関数は、2つのインターリーブされたキャリーあり加算命令シーケンスを、ADCX および ADOX 組込み関数を使用して並列実行できるようにします。組込み関数の戻り値は、合計により生成されるキャリーアウト値です。合計の結果は、sum_out 引数で指定したメモリー位置に格納されます。

_mm_prefetch 組込み関数の新しい _MM_HINT_ET0 ヒント

_MM_HINT_ET0 ヒントは、組込み関数を新しいインテル® アーキテクチャー (開発コード名: Broadwell) でサポートされる PREFETCHW 命令に変更します。_MM_HINT_ET0 を使用する前に、ターゲット CPU が PREFETCHW 命令をサポートしているかどうか確認してください。

4.2.2 マニュアル CPU ディスパッチに `core_4th_gen_avx` を追加 (インテル® Composer XE 2013 Update 1)

`cpu_dispatch` および `cpu_specific` のマニュアル CPU ディスパッチ・メカニズムに `cpuid core_4th_gen_avx` のサポートが追加されました。この `cpuid` は、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) をサポートするプロセッサをターゲットにします。

4.2.3 スタティック解析機能 (旧: 「スタティック・セキュリティ解析」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・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。

4.2.3.1 2011 Update 2 からの “`inspxe-runsc.exe`” 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、2011 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 2011/2013、インテル® Fortran Studio XE 2011/2013 またはインテル® C++ Studio XE 2011/2013 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、結果ファイルの場所を指定する `-result-dir (-r)` コマンドライン・オプションがあります。このオプション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` オプション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 オプションを指定して 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 オプション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。

4.3 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。

4.3.1 インテル® Composer XE 2013 の新規および変更されたコンパイラー・オプション

- -vec-report6
- -f[no-]defer-pop
- -f[no-]optimize-sibling-calls
- -mmic
- -fextend-arguments=[32|64]
- -guide-profile=<file|dir>[,<file|dir>,...]
- -openmp-link <library>
- -opt-prefetch-distance=N[,N]
- -debug [no]pubnames
- -debug [no]profiling
- -grecord-gcc-switches
- -fno-merge-constants
- -check-pointers=<arg>
- -check-pointers-dangling=<arg>
- -std=c++11 (-std=c++0x と同じ)
- -[no-]check-pointers-undimensioned
- -[no-]check-uninit は -check=<keyword>[,<keyword>...] に拡張されました。元の機能には -check:[no]uninit を使用してください。
- -w3
- -W[no-]unused-parameter
- -W[no-]invalid-pch
- -noerror-limit removed
- -diag-enable sc-{full|concise|precise}
- -diag-enable sc-single-file
- -diag-enable sc-enums
- -watch=<keyword>
- -nowatch
- -offload-attribute-target=<name>
- -offload-option,<target>,<tool>,"option list"

- -no-offload
- -fimf-domain-exclusion=classlist[:funclist]
- -ipp-link={static|dynamic|static_thread}
- -fms-dialect=11
- -static-libstdc++
- -[no-]pie
- -opt-streaming-cache-evict=[0|1|2|3]
- -opt-threads-per-core=[1|2|3|4]
- -[no-]opt-assume-safe-padding

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。

4.3.2 -check-pointers=w オプションの追加 (インテル® Composer XE 2013 Update 1)

Update 1 では、ポインターチェッカーに書き込みのみのエラーチェックを実行する機能が追加されました。

4.3.3 -opt-assume-safe-padding オプションと -opt-streaming-cache-evict and -opt-threads-per-core オプションの追加 (インテル® Composer XE 2013 Update 1)

詳細は、「[インテル® メニー・インテグレートッド・コア \(インテル® MIC\) アーキテクチャー](#)」セクションを参照してください。

4.3.4 -ipp-link オプション

このオプションは、使用するインテル® インテグレートッド・パフォーマンス・プリミティブのライブラリーを指定します。-ipp オプションとともに使用されます。static (スタティク・シングルスレッド・ライブラリーにリンク)、dynamic (ダイナミック・ライブラリーにリンク)、static-thread (スタティク・マルチスレッド・ライブラリーにリンク) の3つのオプションがあります。スタティク・マルチスレッド・ライブラリーは [別のパッケージで提供](#)されることに注意してください。

4.3.5 -fimf-domain-exclusion

詳細は、「[インテル® メニー・インテグレートッド・コア \(インテル® MIC\) アーキテクチャー](#)」セクションを参照してください。

4.4 その他の変更

4.4.1 Microsoft* のループプラグマ構文のサポート (インテル® Composer XE 2013 Update 1)

Update 1 では、Microsoft* Visual C++* 2012 コンパイラーの #pragma loop [hint_parallel(n), no_vector, ivdep] のサポートが追加されました。

4.4.2 インテル® Composer XE 2013 の新しい警告レベル -w3 および警告レベルの変更

新しい警告は "icc -help" で次のように表示されます。

```
-w<n>    診断の制御
        n = 0    エラーのみ有効にします。( -w と同じ )
        n = 1    警告とエラーを有効にします。(デフォルト)
        n = 2    詳細な警告、警告、エラーを有効にします。
        n = 3    リマーク、詳細な警告、警告、エラーを有効にします。
```

これまでリマークは -w2 で表示されていましたが、変更後は新しい警告レベル -w3 で表示されます。

4.4.3 `__regcall` 関数および要素関数 (`__declspec(vector)` など) とのバイナリー互換性の変更

インテル® C++ Composer XE 2013 では、`__regcall` 呼び出し規約の制御方法が変更され、以前のコンパイラのバージョンと非互換になりました。このバージョンから、RBX レジスターは `__regcall` ルーチンの呼び出し先セーブのレジスターと見なされます (以前のバージョンでは IA32 ターゲットの場合のみ)。コンパイラの異なるバージョンでビルドされたバイナリーが一緒に使用されるとランタイムエラーが発生する可能性があるため、コンパイラの `__regcall` ルーチンの名前修飾方式が変更され、`__regcall` ルーチンのマングリングには新しい `__regcall2__` プリフィックス (以前のプリフィックスは `__regcall__`) を使用します。このため、コンパイラの異なるバージョンでビルドされたバイナリーは正しくリンクされません。

関数が `__regcall` インターフェイスまたは `__declspec(vector)` 要素関数インターフェイスで宣言された場合、これらの関数を含むコードをインテル® C++ Composer XE 2013 でビルドしたバイナリーと、同様のコードを以前のコンパイラでビルドしたバイナリーは正しくリンクされません。これらの宣言を使用する場合は、インテル® C++ Composer XE 2013 で必要なコードをすべてリビルドしてください。

4.4.4 乱数ジェネレーター関数のベクトル化用の新しい `libirng` ライブラリーの追加 (インテル® Composer XE 2013)

コンパイラは、C 標準ライブラリーで提供される乱数ジェネレーター関数の `drand48` ファミリーを自動的にベクトル化するようになりました。このサポートを実装するため、新しいライブラリー `libirng.a` および `libirng.so` が追加されました。

4.4.5 コンパイラ環境の設定

コンパイラ環境は、`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。コンパイラ環境を設定すると、インテル® デバッガー、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 (インストールされている場合) の環境も設定されます。

4.4.6 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 プロセッサや特定のインテル以外のプロセッサなど、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。

4.4.7 インテル® Composer XE 2013 の新しい警告レベル -w3 および警告レベルの変更

新しい警告は "icc -help" で次のように表示されます。

```
-w<n>    診断の制御
        n = 0    エラーのみ有効にします。( -w と同じ)
        n = 1    警告とエラーを有効にします。(デフォルト)
        n = 2    詳細な警告、警告、エラーを有効にします。
        n = 3    リマーク、詳細な警告、警告、エラーを有効にします。
```

これまでリマークは -w2 で表示されていましたが、変更後は新しい警告レベル -w3 で表示されます。

4.4.8 インテル® Cilk™ Plus の "scalar" 節の削除

本リリースで、インテル® Cilk™ Plus の要素関数のオプションとして使用されていた "scalar" 節が削除されました。代わりに、機能的に同じ "uniform" 節を使用してください。

4.4.9 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 (2011 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。

4.5 既知の問題

4.5.1 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 libcilkrtso がリンクされます。

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

```
icc: 警告 #10237:-lcilkrtso 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。

4.5.2 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。

4.5.3 スタティック解析の既知の問題

4.5.3.1 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)] ステータスは [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。

5 インテル® デバッガー (IDB)

次の注意事項は、IA-32 アーキテクチャー・システムおよびインテル® 64 アーキテクチャー・システムで実行するインテル® デバッガー (IDB) のグラフィカル・ユーザー・イン

ターフェイス (GUI) についてです。このバージョンでは、`idb` コマンドは GUI を起動します。コマンドライン・インターフェイスを起動するには、`idbc` を使用します。

5.1 インテル® デバッガー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® デバッガーが削除される予定です。削除された場合、次の機能が使用できなくなります。

- `idbc` コマンドライン・デバッガー
- `idb` GUI ベース・デバッガー

5.2 Java® ランタイム環境の設定

インテル® IDB デバッガーのグラフィカル環境は、Java® アプリケーションで構築されており、実行には Java® ランタイム環境 (JRE) が必要です。デバッガーは、6.0 (1.6) JRE をサポートしています。

配布元の手順に従って JRE をインストールします。

最後に、JRE のパスを設定する必要があります。

```
export PATH=<path_to_JRE_bin_dir>:$PATH
```

5.3 デバッガーの起動

デバッガーを起動するには、まず始めに、「[コンパイラー環境の設定](#)」で説明されているコンパイラー環境が設定されていることを確認してください。その後、次のコマンドを使用します。

```
idb
```

または

```
idbc
```

(必要に応じて)

GUI が開始され、コンソールウィンドウが表示されたら、デバッグセッションを開始できます。

注: デバッグする実行ファイルが、デバッグ情報付きでビルドされ、実行可能ファイルであることを確認してください。必要に応じて、アクセス権を変更します。

例: `chmod +x <application_bin_file>`

5.4 その他のドキュメント

インテル® コンパイラー/インテル® デバッガー・オンライン・ヘルプは、デバッガーのグラフィカル・ユーザー・インターフェイスの [Help (ヘルプ)] > [Help Contents (ヘルプ目次)] で表示できます。

[Help (ヘルプ)] ボタンが表示されているデバッガーのダイアログから状況依存ヘルプにもアクセスできます。

5.5 デバッガー機能

5.5.1 IDB の主な機能

デバッガーは、インテル® IDB デバッガーのコマンドライン・バージョンのすべての機能をサポートしています。デバッガー機能は、デバッガー GUI または GUI コマンドラインから呼び出すことができます。グラフィカル環境を使用する場合は、既知の制限を参照してください。

5.5.2 インテル® Inspector XE 2011 Update 6 による IDB の “break into debug” のサポート

インテル® Inspector XE 2011 Update 6 は、インテル® Composer XE 2011 Update 6 に含まれるインテル® デバッガーによる “break into debug” モードをサポートしています。詳細は、インテル® Inspector XE 2011 のリリースノートを参照してください。

5.6 既知の問題と変更点

5.6.1 Pardus* システムのデフォルトの .gdbinit スクリプトでデバッガーがクラッシュ

Pardus* システムで idbc または idb を開始したときにデバッガーがクラッシュする場合は、オプション `-nx` を追加してデフォルトの .gdbinit スクリプトを使用しないようにしてください。

5.6.2 Pardus* システムでスレッド情報が利用できない

Pardus* システムのデフォルトの `libthread_db.so` ライブラリーの問題により、デバッガーでマルチスレッド・アプリケーションをデバッグしたときにスレッド情報が検出できません。

5.6.3 Thread Data Sharing Filters (スレッドデータ共有フィルター) が正しく動作しない

Thread Data Sharing Filters (スレッドデータ共有フィルター) を設定すると、デバッガーが予期しない動作をすることがあります。スレッドはデータ共有検出の後に続行せず、デバッガーは SIG SEGV で終了します。

フィルターが有効な状態でデータ共有検出に関連する問題が発生した場合は、[Thread Data Sharing Filters (スレッドデータ共有フィルター)] ウィンドウのコンテキスト・メニューでフィルターをすべて無効にしてください。

5.6.4 コアファイルのデバッグ

コアファイルをデバッグするには、以下のようにコマンドライン・オプションを指定してデバッガー (コマンドライン・デバッガー idbc または GUI デバッガー idb) を開始する必要があります。

```
idb|idbc <executable> <corefile>
```

または

```
idb|idbc <executable> -core <corefile>
```

コアファイルのデバッグを開始すると、デバッガーはライブプロセス (例えば、新しいプロセスのアタッチや作成) をデバッグできません。また、ライブプロセスをデバッグしているときはコアファイルをデバッグできません。

5.6.5 シェルで \$HOME が設定されていないとデバッガーがクラッシュ

デバッガーを起動したシェルで \$HOME 環境変数が設定されていない場合、“セグメンテーション違反” でデバッガーが終了します。

5.6.6 コマンドライン・パラメーター -idb と -dbx は未サポート

デバッガーのコマンドライン・パラメーター -idb と -dbx は、デバッガー GUI ではサポートされていません。

5.6.7 プロセッサのデバッグレジスター (ハードウェア・ベース) を使用したウォッチポイント (インテル® Composer XE 2011 Update 6)

インテル® Composer XE 2011 Update 6 (IDB 12.1) では、プロセッサのデバッグレジスターを使用したウォッチポイントが完全にサポートされています。設定方法は、使用するプロセッサ・アーキテクチャーにより異なります。IA-32 およびインテル® 64 アーキテクチャー・システムでは次の制限があります (可能な場合、インテル® デバッガーは、適切なエラーメッセージを出力します)。

- ウォッチするメモリ領域のサイズは、1、2、4 または 8 (インテル® 64 のみ) バイトでなければなりません。
- ウォッチするメモリ領域の開始アドレスは、ウォッチするサイズでアラインされていなければなりません。例えば、ウォッチするサイズが 2 バイトの場合、開始アドレスは奇数であってはなりません。
- アクティブ/有効なウォッチポイントは最大 4 つまでサポートされています。使用されていないウォッチポイントを無効にすることで、リソースを解放したり、別のウォッチポイントを作成したり有効にすることができます。
- 次のアクセス方法のみサポートされています。
 - 書き込み: 書き込みアクセスでトリガーされます。
 - 指定: 書き込みまたは読み取りアクセスでトリガーされます。
 - 変更: 実際に値を変更した書き込みアクセスでトリガーされます。
- ウォッチするメモリ領域が複数ある場合、それぞれの領域はオーバーラップしてはなりません。
- ウォッチポイントは、スコープには関係ありませんが、プロセスに関連付けられています。プロセスが実行中である限り、ウォッチポイントはアクティブ/有効です。プロセスが終了されると (例えば、プロセスがリターンした場合など)、ウォッチポイントは無効になります。必要に応じて、ユーザーはウォッチポイントを再度有効にすることができます。
- デバッガーを使用してウォッチするメモリ領域にアクセスすると (例えば、変数に異なる値を割り当てるなど)、ハードウェアの検出がスキップされます。そのため、ウォッチポイントは、デバッグ対象がウォッチするメモリ領域にアクセスした場合のみトリガーされます。
- デバッグ対象が仮想マシン内のゲスト OS で実行されている場合、命令やコード行をステップオーバーすると、プロセスは停止しないで継続することがあります。ウォッチポイントは、実際のハードウェアでデバッグ対象を実行した場合にのみ、動作が保証されています。

5.6.8 位置独立実行ファイル (PIE) のデバッグは未サポート

一部のシステムでは、コンパイラーは位置独立実行ファイル (PIE) を生成します。その場合、コンパイル時とリンク時の両方に -fno-pie フラグを指定する必要があります。そうでないと、アプリケーションをデバッグできません。

5.6.9 コマンドライン・パラメーター -parallel は未サポート

デバッガーのコマンドライン・パラメーター -parallel は、シェルのコマンドプロンプトおよびデバッガー GUI のコンソールウィンドウではサポートされていません。

5.6.10 [Signals (シグナル)] ダイアログが動作しない

GUI ダイアログの [Debug (デバッグ)] > [Signal Handling (シグナル処理)]、またはショートカット・キーの Ctrl+S でアクセス可能な [Signals (シグナル)] ダイアログが正しく動作しないことがあります。シグナル・コマンドライン・コマンドを代わりに使用する場合は、インテル® デバッガー (IDB) マニュアルを参照してください。

5.6.11 GUI のサイズ調整

デバッガーの GUI ウィンドウのサイズが小さくなり、一部のウィンドウが表示されていないことがあります。ウィンドウを拡大すると、隠れているウィンドウが表示されます。

5.6.12 \$cdir ディレクトリー、\$cwd ディレクトリー

\$cdir はコンパイル・ディレクトリーです (記録されている場合)。\$cdir は、ディレクトリーが設定されている場合にサポートされます。シンボルとしてサポートされるわけではありません。

\$cwd は現在の作業ディレクトリーです。セマンティクスもシンボルもサポートされていません。

\$cwd と '.' の違いは、\$cwd はデバッグセッション中に変更された現在の作業ディレクトリーを追跡する点です。 '.' は、ソースパスへのエントリーが追加されると直ちに現在のディレクトリーに展開されます。

5.6.13 info stack の使用

デバッガーコマンド info stack は、以下のように、負のフレームカウントの使用方法が現在 gdb とは異なります。

```
info stack [num]
```

num が正の場合は最内の num フレーム、ゼロの場合はすべてのフレーム、負の場合は最内の -num フレームを逆順で出力します。

5.6.14 \$stepg0 のデフォルト値の変更

デバッガー変数 \$stepg0 のデフォルト値が 0 に変更されました。値 "0" の設定では、"step" コマンドを使用する場合、デバッガーはデバッグ情報なしでコードにステップオーバーします。以前のデバッガーバージョンと互換性を保つようにするには、次のようにデバッガー変数を 1 に設定します。

```
(idb) set $stepg0 = 1
```

5.6.15 一部の Linux* システムでの SIGTRAP エラー

一部の Linux* ディストリビューション (例: Red Hat* Enterprise Linux* Server 5.1 (Tikanga)) では、デバッガーがブレークポイントで停止した後、ユーザーがデバッグを続行すると SIGTRAP エラーが発生することがあります。この問題を回避するには、SIGTRAP シグナルを次のようにコマンドラインで定義します。

```
(idb) handle SIGTRAP nopass noprint nostop
SIGTRAP is used by the debugger.
SIGTRAP      No      No      No      Trace/breakpoint trap
(idb)
```

警告: この回避策は、デバッグ対象にシグナルを送信するすべての SIGTRAP がブロックされます。

5.6.16 MPI プロセスのデバッグには idb GUI は使用不可

MPI プロセスのデバッグに idb GUI を使用することはできません。コマンドライン・インターフェイス (idbc) を使用してください。

5.6.17 GUI でのスレッド同期ポイントの作成

単純なコードやデータのブレークポイントでは [Location (場所)] が必須です。スレッド同期ポイントでは [Location (場所)] と [Thread Filter (スレッドフィルター)] の両方が必須です。スレッド同期ポイントは、スレッドの同期を指定します。その他の種類のブレークポイントでは、このフィールドは作成されたブレークポイントの中からリストされているスレッドに関するものだけに制限します。

5.6.18 [Data Breakpoint (データ・ブレークポイント)] ダイアログ

[Within Function (関数内)] フィールドと [Length (長さ)] フィールドは使用されていません。ウォッチする場所は、ウォッチする長さを暗黙的に提供します (効率的な式の型が使用されます)。また、[Read (読み取り)] アクセスも利用できません。

5.6.19 IA-32 アーキテクチャー向けのスタック・アライメント

IA-32 アーキテクチャー向けのデフォルトのスタック・アライメントの変更に伴い、下位呼び出し (デバッグ対象のコードを実行する式の評価など) を使用すると失敗することがあります。場合によっては、デバッグ対象がクラッシュし、デバッグセッションが再起動されることもあります。この機能を使用する場合は、`-falign-stack=<mode>` オプションを使用して 4 バイトのスタック・アライメントでコードをコンパイルしてください。

5.6.20 GNOME 環境の問題

GNOME 2.28 では、デバッガーのメニューアイコンがデフォルトで表示されないことがあります。メニューアイコンを表示するには、[System (システム)] > [Preferences (設定)] > [Appearance (外観の設定)] > [Interface (インターフェイス)] タブで [Show icons in menus (メニューにアイコンを表示)] を有効にします。[Interface (インターフェイス)] タブがない場合は、次のようにコンソールで `gconf` キーを使用してこの変更を行うことができます。

```
gconftool-2 --type boolean --set
/desktop/gnome/interface/buttons_have_icons true
gconftool-2 --type boolean --set
/desktop/gnome/interface/menus_have_icons true
```

5.6.21 オンラインヘルプへのアクセス

システムで IDB デバッガー GUI の [Help (ヘルプ)] メニューからオンラインヘルプにアクセスできない場合は、次の Web ベースのドキュメントを利用できます。

<http://intel.ly/o5DMp9>

6 Eclipse* 統合

インテル® C++ コンパイラーでは、Eclipse* 機能と関連プラグイン (インテル® C++ Eclipse* 製品拡張) がインストールされます。これらを Eclipse* 統合開発環境 (IDE) として追加すると、インテル® C++ コンパイラーが Eclipse* でサポートされます。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。

6.1 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/cdt8.1/eclipse

統合には、Eclipse* プラットフォームのバージョン 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。

<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 3.7、Eclipse* C/C++ Development Tools (CDT) のバージョン 8.0 以降、および Java* ランタイム環境 (JRE) 6.0 (1.6) Update 11 が必要です。

6.1.1 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++ Composer XE 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++ Composer XE 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++ Composer XE 統合をインストールすることはできません。

6.2 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_2013.<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 13.0 for Linux* OS (インテル® C++ コンパイラー XE 12.0 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。インテル® デバッガー (idb) と Eclipse* 製品拡張もインストールした場合は、同じ方法で idb 製品拡張サイトを Eclipse* に追加することで、Eclipse* 内で idb を使用できるようになります。

[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 13.0 User and Reference Guides (インテル® C++ コンパイラー XE 12.1 ユーザー・リファレンス・ガイド)] で表示できます。

6.2.1 Eclipse* へのインテル® デバッガーの統合

上記の手順を実行して Eclipse* を再起動してから、次の手順に従って Eclipse* にインテル® デバッガーを統合します。

- [Run (実行)] > [Debug Configurations... (デバッグ設定...)] を選択してデバッグの起動設定を作成します。
- 表示されるダイアログボックスで [C/C++ Application (C/C++ アプリケーション)] を右クリックして [New (新規)] を選択します。
- 右側にいくつかのタブが表示されます。右下に [Using GDB (DSF) Create Process Launcher - Select other... (GDB (DSF) プロセス作成ランチャーの使用 - その他の選択...)] というラベルが表示されます。これをクリックするとダイアログが表示されます。[Standard Create Process Launcher (標準プロセス作成ランチャー)] を選択して [OK] をクリックします。
- [Debugger (デバッガー)] タブでコンボボックスからインテル® デバッガー (idbc) を選択します。idbc を idbc へのフルパスに置換します。

6.3 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* 4.2 および CDT 8.1 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。

6.3.1 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を作成し、cd でこのディレクトリに移動します。ここでは、このディレクトリを <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。

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

LANG 環境変数を設定していない場合は、設定してください。次に例を示します。

```
setenv LANG ja_JP.UTF8
```

Eclipse* を起動する前に `compilervars.csh` (または `.sh`) スクリプトを実行して、インテル® C++ コンパイラー関連の環境変数を設定します。

```
source <install-dir>/bin/iccvars.csh arch_arg ("arch_arg" は "ia32" または "intel64" のいずれか)
```

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参照します。

例:

```
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
```

6.5 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のレベルが低いためです。この新しいセキュリティは、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 セキュリティを調整する必要があります。

6.6 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、9.1、10.0、10.1、11.0、11.1、12.0、12.1、13.0 です。インテル® 64 アーキテクチャー・システムでは、コンパイラー・バージョン 11.0、11.1、12.0、12.1、13.0 がサポートされています。

7 インテル® インテグレートッド・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® インテグレートッド・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/OmWl4d> (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。

7.1 別途ダウンロード可能なインテル® IPP スタティック・スレッド・ライブラリー

インテル® IPP ライブラリーのスタティック・スレッド・バージョンはインテル® Composer XE パッケージに含まれなくなりました。これらのライブラリーは、インテル® Composer XE パッケージをダウンロードした場所から別途ダウンロードできます。

7.2 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。

7.3 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。

8 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 (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、次の Web サイトを参照してください。<http://intel.ly/OeHQqf>

8.1 注意事項

注意事項についての詳細は、[削除された機能に関するナレッジベースの記事](#) (英語) を参照してください。

- インテル® MKL GNU Multiple Precision* (GMP) 関数インターフェイスを削除
- タイミング関数 `mkl_set_cpu_frequency()` を無効化 - インテル® MKL リファレンス・マニュアルで説明されているように、`mkl_get_max_cpu_frequency()`、`mkl_get_clocks_frequency()`、`mkl_get_cpu_frequency()` を使用してください。
- MKL_PARDISO 定数を削除 - `mkl_domain_set_num_threads()` 関数で PARDISO ドメインを指定する場合は `MKL_DOMAIN_PARDISO` を使用してください。
- インテル® MKL 10.2 Update 4 の畳み込み関数と相関関数の特別な後方互換関数を削除
- インテル® Pentium® III プロセッサのサポートが終了。サポートされる最小の命令セットはインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) になります。

- ドキュメント:
 - HTML 形式のインテル® MKL リファレンス・マニュアルの提供を終了
 - man ページおよび Eclipse* ヘルプ統合のドキュメントの提供を終了

8.2 本バージョンでの変更

8.2.1 インテル® MKL 11.0 Update 1 の新機能

- BLAS:
 - [S/D/C/Z]SYMM をインテル® メニー・インテグレートッド・コア (インテル® MIC) アーキテクチャーのネイティブ実行用に最適化
 - AMD* CPU (開発コード名: Bulldozer) における DGEMM および倍精度レベル 3 BLAS のパフォーマンスが向上
- スパース BLAS: インテル® MIC アーキテクチャーにおける複素共役転置およびエルミート行列の CSR MV のパフォーマンスが大幅に向上
- LAPACK:
 - ?(SY/HE)TRD、?(OR/UN)M(LQ/QL/QR/RQ)、?(OR/UN)GQR、?GE(QR/LQ/RQ/QL)F 関数をインテル® MIC アーキテクチャーのネイティブ実行用に最適化
 - インテル® MIC アーキテクチャーにおける ?GETRF および SMP LINPACK のベンチマーク・ネイティブ・パフォーマンスが向上
 - ?GETRF、?GEQRF、?PORTF 関数をインテル® MIC アーキテクチャーの自動オフロード用に最適化
- PARDISO: エルミート行列の対数値の虚部を無視
- クラスター FFT:
 - ハイブリッド・クラスター FFT (MPI + OpenMP*) のパフォーマンスが向上 (最大 2 倍)
 - よりコミュニケーションの少ない新しいクラスター FFT アルゴリズムを 1D FFT 用に実装 - 有効にするには環境変数 MKL_CFFT_SOI_ENABLE を YES または 1 に設定 (詳細はインテル® MKL のドキュメントを参照)
- VSL:
 - ユーザーが定義した平均で記述統計の推定値を計算する VSL_SS_METHOD_FAST_USER_MEAN メソッドのサポートを追加
 - インテル® Xeon® プロセッサ E5-2690 において記述統計の推定値を計算する VSL_SS_METHOD_FAST メソッド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ける raw および中心モーメント、分散共分散の推定値を計算する記述統計アルゴリズム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ける MT2203 および WH BRNG のパフォーマンスが向上
- 転置: 第 2 世代インテル® Core™ マイクロアーキテクチャーにおけるアウトオブプレーズ転置のパフォーマンスが向上 (最大 7 倍)
- サービス関数: 古い名前の 7 つのサービス関数を削除 (詳細は[削除された古いサービス関数についての記事](#)を参照)
- [不具合の修正](#)

8.2.2 最初のリリースでの変更

- インテル® MKL がインテル® メニー・インテグレートッド・コア (インテル® MIC) アーキテクチャー・ベースのインテル® Xeon Phi™ コプロセッサがサポートされるようになりました (Linux* のみ)。インテル® Xeon Phi™ コプロセッサでのインテル® MKL 使用モデルは、自動オフロード、コンパイラ支援オフロード、ネイティブ実行の 3 つです。インテル® MKL の大部分が、これらのコプロセッサ上でネイティブに実行するように移植されました。一部の関数は、ホストとインテル® Xeon Phi™ コプロセッサ間で計算量を自動的に分割するように最適化されました (自動オフロード (AO) 機能)。詳細は、『インテル® MKL ユーザーガイド』を参照してください。ポア

ソン・ライブラリー、反復法スパースソルバー、Trust-Region ソルバーを除く、ほとんどの標準的な Intel® MKL 関数は、Intel® Xeon Phi™ コプロセッサで実行されます。

- 条件付きビット単位演算の再現性 (CBWR): コード分岐の選択により柔軟性を持たせ、アルゴリズムに決定性があることを保証して、パフォーマンスと再現性のある結果のバランスを取る Intel® MKL の新しい機能です。詳細は、『Intel® MKL ユーザーガイド』を参照してください。また、[CBWR ナレッジベースの記事](#) (英語) も参照してください。
- Intel® MKL は、新しい FMA3 命令を含む、新しい Intel® アドバンスド・ベクトル・エクステンション 2 (Intel® AVX2) を使用した最適化もサポートします。詳細は、[Intel® AVX2 のサポートに関するナレッジベースの記事](#) (英語) を参照してください。
- BLAS:
 - [S/D/C/Z]GEMM、[S/D/C/Z]TRMM、[S/D/C/Z]TRSM、[S/D/C/Z]SYRK、[S/D]GEMV、[S/D]AXPY、[S/D]DOT を Intel® MIC アーキテクチャーのネイティブ実行用に最適化 (?TRMM、?TRSM、?GEMM 関数は自動オフロード用に最適化)
 - Intel® アドバンスド・ベクトル・エクステンション (Intel® AVX) 対応の 64 ビット・プログラムで DSYRK/SSYRK のパフォーマンスが向上
- LAPACK:
 - [S/D]GETRF、[S/D]POTRF、[S/D]GEQRF、[S/D]GELQF、[S/D]GEQLF、[S/D]GERQF を Intel® MIC アーキテクチャーのネイティブ実行用に最適化
 - LAPACK バージョン 3.4.1 のサポートを追加
- FFT:
 - 単精度と倍精度の実数-複素数および複素数-複素数の 1、2、3 次元高速フーリエ変換を Intel® MIC アーキテクチャーのネイティブ実行用に最適化
 - 記述子あたりのスレッド数を制限する構成パラメーター DFTI_THREAD_LIMIT を追加
 - 64 ビット整数で指定されたサイズの 1D 実数-複素数変換のサポートを追加
- VML /VSL:
 - 複雑な SinCos および CIS 関数を Intel® MIC アーキテクチャーのネイティブ実行用に最適化
 - MT19937、MT2203、MRG32k3a BRNG、離散一様分布および幾何分布 RNG を Intel® MIC アーキテクチャーのネイティブ実行用に最適化
 - Intel® アドバンスド・ベクトル・エクステンション (Intel® AVX) における viRngGeometric のパフォーマンスが向上
 - データ適合 Integrate1d 関数にスレッド化を実装
- 転置: パフォーマンスを大幅に向上させるために単精度と倍精度のリーディング・ディメンションが行列サイズより大きな正方行列のインプレース転置を並列化
- スレッド制御の柔軟性を高めるローカルスレッド制御関数 (mkl_set_num_threads_local) を実装
- mklvars.* スクリプトで環境変数 \$FPATH の設定と内部変数 MKL_TARGET_ARCH のエクスポートを行わないように変更 (Intel® コンパイラーはこれらの変数を使用しなくなったため、この変更によるユーザーへの影響はありません)
- リンクツール: Intel® MIC アーキテクチャーのサポートを追加
- リンク・ライン・アドバイザー:
 - アーキテクチャー (IA-32/Intel® 64) とインターフェイス・レイヤー (LP64/ILP64) を選択するアドバイザー機能を追加
 - Intel® MIC アーキテクチャーのサポートを追加

8.3 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 Intel 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用に提供されています。

インテル® MKL クラスタ・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。

9 インテル® スレディング・ビルディング・ブロック

インテル® スレディング・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。

10 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ます。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。

<http://www.intel.com/design/literature.htm>

インテル・プロセッサ・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・ナンバーは同一プロセッサ・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・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、Xeon Phi 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。

© 2012 Intel Corporation. 無断での引用、転載を禁じます。