インテル® C++ Composer XE 2013 SP1 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
2014 年 4 月 9 日

目次

1 概要

このドキュメントでは、製品のインストール方法、新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題について説明します。

インテル® C++ Composer XE 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 3

- インテル® C++ コンパイラー XE 14.0.3
- インテル® マス・カーネル・ライブラリー 11.1 Update 3
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 8.1 Update 1
- インテル® スレッディング・ビルディング・ブロック 4.2 Update 4
- インテル独自の x86intrin.h の提供
- 報告された問題の修正

1.1.2 Update 2

- インテル® C++ コンパイラー XE 14.0.2
- インテル® マス・カーネル・ライブラリー 11.1 Update 2
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 8.1
- インテル® スレッディング・ビルディング・ブロック 4.2 Update 3
- 新しいインテル® Cilk™ Plus STL ベクトル・レデューサー (インテル® C++ Composer XE 2013 SP1 Update 2)
- KMP_DYNAMIC_MODE 環境変数による “asat” サポートの廃止
- 報告された問題の修正

1.1.3 Update 1

- バージョン 14.0 日本語版の初期リリース
- インテル® C++ コンパイラー XE 14.0.1
- インテル® マス・カーネル・ライブラリー 11.1 Update 1
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 8.0 Update 1
- インテル® スレッディング・ビルディング・ブロック 4.2 Update 1
- GNU* プロジェクト・デバッガー (GDB) によるインテル® 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
- インテル® MIC アーキテクチャー向けの新しいコンパイラー・オプション `/Qopt-gather-scatter-unroll (opt-gather-scatter-unroll)`
- 新しいコンパイラー・オプション /Q[a]xMIC-AVX512(-[a]xMIC-AVX512)
- マルチプロセッサー通信環境 (MPC) 統合並列ランタイム向けにすべてのスタティック・データをプライベート化する新しいコンパイラー・オプション -f[no-]mpc_privatize
- インテル® Memory Protection Extensions (インテル® MPX) 向けの新しいコンパイラー・オプション `/Qcheck-pointers-mpx (-check-pointers-mpx)`
- インテル® Cilk™ Plus SIMD 対応関数 (例: `__declspec(vector(uniform(this)))` が “`uniform(this)`” に対応

- 新しい組込み関数 `_allow_cpu_features`
- 新しい数値文字列変換ライブラリー: `libistrconv`
- 報告された問題の修正

1.1.4 インテル® Composer XE 2013 からの変更点

- オンライン・インストーラー
- GUI インストーラー
- インテル® C++ コンパイラー XE 14.0.0
- GNU* プロジェクト・デバッガー (GDB*)
- インテル® デバッガーのサポート終了予定
- Fedora* 18 および 19 をサポート
- Ubuntu* 13.04、Debian* 7.0 をサポート
- 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:
  - Fedora* 17
  - Ubuntu* 11.10
  - Pardus* 2011.2
- C++11 の機能 (-std=c++11)
- OpenMP* 4.0 の一部サポート
- インテル® Cilk™ Plus の変更
- DWARF V4 サポート
- __INTEL_COMPILER_UPDATE 事前定義済みマクロ
- ポインター型のアライメント修飾子
- フォルス・シェアリングを回避する変数定義属性
- -mtune パフォーマンス・チューニング・オプション
- 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
- ほかの OpenMP* 機能とは関係なく特定の OpenMP* 4.0 機能を有効/無効にする -openmp-offload/-openmp-simd オプションの追加
- __GXX_EXPERIMENTAL_CXX0X__ マクロの未サポート
- Silvermont† マイクロアーキテクチャー向けの -xATOM_SSE4.2 オプションの追加
- インテル® マス・カーネル・ライブラリー11.1
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 8.0 Update 1
- インテル® スレッディング・ビルディング・ブロック 4.2

1.2 製品の内容

インテル® C++ Composer XE 2013 SP1 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 14.0.3。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・システム、およびインテル® Xeon Phi™ コプロセッサ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*) 7.5
- インテル® デバッガー 13.0
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 8.1 Update 1
- インテル® マス・カーネル・ライブラリー 11.1 Update 3
- インテル® スレッディング・ビルディング・ブロック 4.2 Update 4
- 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)
- 機能を最大限に活用できるよう、マルチコアまたはマルチプロセッサー・システムの使用を推奨します。
- RAM 1GB (2GB 推奨)
- 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* 18,19
    - Red Hat* Enterprise Linux* 5、6
    - SUSE* Linux* Enterprise Server 10、11
    - Ubuntu* 12.04 LTS、13.04
    - 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 4.1-4.8 をサポート
    - binutils バージョン 2.17-2.23 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。

GNU GDB を使用するためのその他の条件*

- 本製品とともに提供される GNU* GDB を使用するには、Python* 2.4、2.6、または 2.7 が必要です。

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 (1.6t) – 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ー (例: インテル® Pentium® 4 プロセッサー) が必要な IA-32 アーキテクチャー・アプリケーションをビルドします。コンパイラー・オプションを使用して任意の IA-32 アーキテクチャー・プロセッサー上で動作するコードを生成できます。ただし、アプリケーションでインテル® インテグレーテッド・パフォーマンス・プリミティブまたはインテル® スレッディング・ビルディング・ブロックを使用している場合、そのアプリケーションの実行には、インテル® SSE2 命令対応のプロセッサーが必要です。
- 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。

1.3.1 Red Hat* Enterprise Linux* 5 および SUSE* Linux* Enterprise Server 10 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 SUSE* Linux* Enterprise Server 10 はサポートされなくなる予定です。

1.3.2 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ー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。 改訂 #20110804

1.5 サンプル

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

1.6 日本語サポート

インテル® コンパイラーは、日本語ユーザー向けのサポートを提供しています。エラーメッセージ、ビジュアル開発環境ダイアログ、ドキュメントの一部が英語のほかに日本語でも提供されています。エラーメッセージやダイアログの言語は、システムの言語設定に依存します。日本語版ドキュメントは、Documentation および Samples ディレクトリー以下の ja_JP サブディレクトリーにあります。

日本語版は、インテル® C++ Composer XE 2013 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。

1.7 テクニカルサポート

インストール時に製品の登録を行わなかった場合は、インテル® ソフトウェア開発製品レジストレーション・センター (http://registrationcenter.intel.com) で登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

テクニカルサポート、製品のアップデート、ユーザーフォーラム、FAQ、ヒント、およびその他のサポート情報は、http://www.intel.com/software/products/support/ (英語) を参照してください。

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

2 インストール

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 –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 新しい GUI インストーラー (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 では、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 新しいオンライン・インストーラー (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 では、デフォルトの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。このインストール・パッケージを利用するには、インターネット接続が必要です。また、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。インターネットに接続されていないマシンにインストールする場合は、オンライン・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。

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”
```

2.3 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® Composer XE 2013 SP1 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。

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。

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® 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。

2.6 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。

2.7 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 SELinux (Security-enhanced Linux*)

以前のバージョンのインテル® Composer XE では、インストールの際に SELINUX モードを permissive に設定する必要がありました。インテル® Composer XE 2013 以降では、これは必要なくなりました。

2.10 既知のインストールの問題

- 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 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。

2.11 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。

ディレクトリー構成はインテル® 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® インテグレーテッド・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® スレッディング・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
- ism-インテル® Software Manager 関連のファイル
- 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_sp1 - 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 SP1 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_sp1.<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_2013_sp1 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 SP1 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
- 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_2013_sp1.<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® Composer XE 2013 SP1 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® インテグレーテッド・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
- mkl - 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
- Uninstall –アンインストール用のファイル
- .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® 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。

2.12 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>/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 14.0 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・インテグレーテッド・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™ コプロセッサー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 の機能セットが拡張されます。

コプロセッサーが利用可能な場合、コードのこれらのセクションはコプロセッサーで実行されます。コプロセッサー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、*コプロセッサー*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します。

- 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® マス・カーネル・ライブラリー (インテル® MKL)
- インテル® スレッディング・ビルディング・ブロック (インテル® TBB)
- Eclipse* IDE 統合
- デバッガー:
    - GNU* GDB
    - インテル® デバッガー

3.2 互換性

本リリースでは、インテル® Xeon Phi™ コプロセッサー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。

オフロード・ライブラリーのバイナリー互換性の変更に関連して、すべてのコードをインテル® Composer XE 2013 Update 1 でリビルドすることを推奨します。

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ー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、「コンパイラー環境の設定」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 SP1 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は、Composer XE のドキュメント・ディレクトリーに含まれています。

3.5 デバッガー

インテル® Composer XE 2013 SP1 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は、以下の場所にあります。

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

3.5.1 GNU* GDB

「GNU* GDB デバッガー」セクションを参照してください。

3.5.2 インテル® デバッガー
「インテル® デバッガー (IDB)」セクションを参照してください。

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 アーキテクチャー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++ コンパイラー

3.7.1.1 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。

3.7.1.2 インテル® Composer XE 2013 SP1 の新しいオフロード節
インテル® Composer XE 2013 SP1 では、オフロード宣言子に 3 つの節 “mandatory”、“optional”、“status” が追加されています。

- “mandatory”: “status” 節が指定されていない場合、カードが利用できないとオフロードコードはアボートします。“status” 節が指定されている場合、ユーザーコードで指定された動作になります。
- “optional”: カードが利用できないとオフロードコードは CPU で実行されます。

これらの節は、オフロード関連のコンパイラー・オプションの設定よりも優先されます。

3.7.1.3 インテル Composer XE 2013 SP1 の -offload オプションの変更
インテル Composer XE 2013 SP1 では、-offload オプションでキーワードを指定できるようになりました。

-offload=none: コンパイル時にすべてのオフロード宣言子が無視され、警告が出力されます。

-offload=mandatory (デフォルト): すべてのオフロード宣言子が処理されます。カードが利用できない場合、プログラムはアボートします。

-offload=optional: すべてのオフロード宣言子が処理されます。カードが利用できない場合、オフロードコードは CPU で実行されます。

これらのオプションよりも、ユーザーが指定したオフロード節が優先されます。

3.7.1.4 インテル® Composer XE 2013 SP1 で追加されたオフロード動作を制御する環境変数
いくつかの新しい環境変数が追加されました。

- OFFLOAD_DEVICES: 変数で指定されたインテル® Xeon™ Phi コプロセッサー・カードのみ使用するようにプロセスを制御します。

- OFFLOAD_INIT: オフロードランタイムにインテル® Xeon™ Phi コプロセッサーを初期化するタイミングのヒントを与えます。
- OFFLOAD_REPORT: オフロードでさまざまなレベルのトレースと統計情報をサポートします。
- OFFLOAD_ACTIVE_WAIT: DMA 転送中、CPU がビジー状態を維持できるようにします。

3.7.1.5 インテル® Composer XE 2013 の初期リリースでビルドしたアプリケーションをインテル® Composer XE 2013 Update 1 のオフロード・ライブラリーを使用して実行するとランタイムエラーまたはクラッシュが発生

インテル® Composer XE 2013 Update 1 のオフロード・ライブラリーではバイナリー互換性が変更されているため、インテル® Composer XE 2013 初期リリースのコンパイラーでビルドされたバイナリーを、インテル® Composer XE 2013 Update 1 以降のオフロード・ライブラリーを使用して実行すると、ランタイムエラーが発生したり、クラッシュします。例えば、次のようなエラーが発生します。

エラー 1:

```
***Warning: offload to device #0 : failed (警告: デバイス #0 へのオフロード : 失敗しました)
```

エラー 2:

```
Segmentation fault (core dumped) (セグメンテーション違反 (コアがダンプされました))
```

エラー 3:

```
terminate called after throwing an instance of 'COIRESULT'
('COIRESULT' のインスタンスをスローした後に terminate が呼び出されました)
terminate called recursively (terminate が繰り返し呼び出されました)
```

エラー 4:

```
CARD--ERROR:1 myoiPageFaultHandler:(nil) Out of Range! (カード--エラー:1 myoiPageFaultHandler: (nil) 範囲外です!)
CARD--ERROR:1 _myoiPageFaultHandler:(nil) switch to default signal handle (カード--エラー:1 _myoiPageFaultHandler: (nil) デフォルトのシグナルハンドルに)切り替えます)
CARD--ERROR:1 Segment Fault! (カード--エラー:1 セグメンテーション違反です!)
HOST--ERROR:myoiScifGetRecvId:Call recv() Header Failed ! errno = 104 (ホスト--エラー:myoiScifGetRecvId: recv() ハンドラーの呼び出しに失敗しました! errno = 104)
HOST--ERROR:myoiScifSend:Call send() Failed! errno = 104 (ホスト--エラー:myoiScifSend: send() の呼び出しに失敗しました! errno = 104)
HOST--ERROR:myoiSend:Fail to send message! (ホスト--エラー:myoiSend: メッセージの送信に失敗しました!)
HOST--ERROR:myoiBcastToOthers:Fail to send message to 1! (ホスト--エラー:myoiBcastToOthers: 1 へのメッセージの送信に失敗しました!)
HOST--ERROR:myoiBcast:Fail to send message to others! (ホスト--エラー:myoiBcast: ほかへのメッセージの送信に失敗しました!)
```

これらの問題を解決するには、インテル® Composer XE 2013 Update 1 以降のコンパイラーで、新しいオフロード・ライブラリーを使用するように、すべてのファイルを再コンパイルしてください。

3.7.1.6 デフォルトのコード生成でインテル® Xeon Phi™ コプロセッサー 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™ コプロセッサー 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。

3.7.1.7 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"
```

3.7.1.8 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 (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;
```

3.7.1.9 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。

3.7.1.10 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。

3.7.1.11 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。

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 SP1 には、インテル® C++ コンパイラー XE 14.0 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。

- 独自の x86intrin.h ファイルの提供 (インテル® C++ Composer XE 2013 SP1 Update 3)
- 新しいインテル® Cilk™ Plus STL ベクトル・レデューサー (インテル® C++ Composer XE 2013 SP1 Update 2)
- 新しい組込み関数 _allow_cpu_features (インテル® C++ Composer XE 2013 SP1 Update 1)

- インテル® Cilk™ Plus SIMD 対応関数の uniform 節 (例: __declspec(vector(uniform(this)))) が this ポインターに対応 (インテル® C++ Composer XE 2013 SP1 Update 1)
- 新しい数値文字列変換ライブラリー libistrconv (インテル® C++ Composer XE 2013 SP1 Update 1)
- C++11 の機能 (-std=c++11)
  - 初期化子リストの完全実装(N2672 と N3217 を参照)
  - インライン名前空間の完全実装(N2535 を参照)
  - 非スタティック・データ・メンバー初期化子の完全実装(N2756 を参照)
  - 一般化された定数式の完全実装(N2235 を参照)
  - 無制限共用体の完全実装(N2544 を参照)
  - コンストラクターのデリゲート(N1986 を参照)
  - *this の rvalue 参照(N2439 を参照)
  - raw 文字列(N2442 を参照)
  - 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代入演算子(N3053 を参照)
  - __bases および __direct_bases 型特性
  - クラス定義で状況依存キーワード "final" を、メンバー関数宣言で "final"と "override" を使用可能(N2928、N3206、N3272 を参照)
  - "noexcept" 指定子と演算子の完全実装(N3050 を参照)。noexcept の遅延インスタンス化を含む (Core issue 1330)。
- OpenMP* 4.0 RC1 と TR1 の一部サポート
- インテル® C++ Composer XE 2013 SP1 のインテル® Cilk™ Plus の変更点
- DWARF V4 サポート
- __INTEL_COMPILER_UPDATE 事前定義済みマクロ
- ポインター型のアライメント修飾子
- フォルス・シェアリングを回避する変数定義属性
- -mtune パフォーマンス・チューニング・オプション

4.2.1 独自の x86intrin.h ファイルの提供 (インテル® C++ Composer XE 2013 SP1 Update 3)

一部の非インテル組込み関数のサポートに関する問題を回避するため、インテル® C++ Composer XE 独自の x86intrin.h ヘッダーファイルが提供されるようになりました。

4.2.2 新しいインテル® Cilk™ Plus STL ベクトル・レデューサー (インテル® C++ Composer XE 2013 SP1 Update 2)

Update 2 で、reducer_vector クラスが追加されました。このクラスを使用するには、ヘッダーファイル"cilk/reducer_vector.h"をインクルードする必要があります。レデューサー・タイプは cilk::reducer< cilk::op_vector<type>>です。詳細は、ヘッダーファイルのコメントを参照してください。

4.2.3 新しい組込み関数 _allow_cpu_features (インテル® C++ Composer XE 2013 SP1 Update 1)

新しい組込み関数 _allow_cpu_features([xxx][,xxx])が immintrin.h に追加されています。この組込み関数は、特定の最適化が行えるように、この組込み関数に続くコード領域を特定の機能を備えたプロセッサー向けにできることをコンパイラーに知らせます。

注: この組込み関数のサポートはまだ準備段階です。指定したコード領域ですべてのコンパイラーの最適化フェーズが実行されるわけではありません。

詳細は、コンパイラー・ドキュメント、コードサンプル、「New intrinsic _allow_cpu_features support」(英語) の記事を参照してください。

4.2.4 インテル® Cilk™ Plus SIMD 対応関数の uniform 節 (例: __declspec(vector(uniform(this)))) が this ポインターに対応 (インテル® C++ Composer XE 2013 SP1 Update 1)

同一のクラス・オブジェクトが SIMD 対応クラスメンバー関数を呼び出す場合、呼び出し先の SIMD 宣言で明示的に "uniform(this)" 節を指定することでパフォーマンスが向上する可能性があります (呼び出し先でどの程度 "this" キーワードが使用されているかに依存します)。使用モデルは、仮引数に適用される uniform 節と同じです。

詳細は、コンパイラー・ドキュメントを参照してください。

4.2.5 新しい数値文字列変換ライブラリー libistrconv (インテル® C++ Composer XE 2013 SP1 Update 1)

新しい数値文字列変換ライブラリー libistrconv は、ASCII 文字列と C データ型の変換を行う、パフォーマンスが最適化されたルーチン群を提供します。新しい API は、"istrconv.h" ヘッダーファイルで宣言されています。

詳細は、コンパイラー・ドキュメントを参照してください。

4.2.6 OpenMP* 4.0 RC2 機能のサポート (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 は、OpenMP* 4.0 機能の一部をサポートしています。OpenMP* 4.0 仕様 (http://openmp.org) で定義されている、次の機能がサポートされています。

- TEAMS プラグマ、宣言子、節
- DISTRIBUTE プラグマ、宣言子、節
- SIMD プラグマ、宣言子、節
- TARGET プラグマ、宣言子、節 (装着されているコプロセッサーまたはデバイス)
- #pragma omp taskgroup 構造
- アトミック節 seq_cst
- 6 つの新しい形式のアトミックなキャプチャーと更新:
  - Atomic swap:{v = x; x = expr;}
  - Atomic update: x = expr binop x;
  - Atomic capture 1: v = x = x binop expr;
  - Atomic capture 2: v =x = expr binop x;
  - Atomic capture 3:{x = expr binop x; v = x;}
  - Atomic capture 4:{v = x; x = expr binop x;}
- proc_bind(<type>)節。<type>は "spread"、"close"、または "master"。
- OMP_PLACES 環境変数
- OMP_PROC_BIND 環境変数
- omp_get_proc_bind() API

詳細は、http://intel.ly/W7CHjb (英語) を参照してください。

4.2.7 インテル® C++ Composer XE 2013 SP1 のインテル® Cilk™ Plus の変更点

インテル® C++ Composer XE 2013 SP1 では、インテル® Cilk™ Plus の次の新機能が追加されています。

- gcc と OpenMP* のベクトル関数の実装との互換性を向上するため、SIMD 対応関数の実装が変更されています。この変更により、インテル® C++ コンパイラーの以前のバージョン (13.1 以前) とバイナリー互換性がなくなります。インテル® C++ コンパイラー 14.0 で SIMD 対応関数を使用するすべてのコードをリビルドするか、以前の実装を使用するには -vecabi=legacy コンパイラー・オプションを指定する必要があります。
- 新しい乗算レデューサーが cilk/reducer_opmul.h で定義されています。

- ビット単位のリダクション操作をサポートするため、次の 3 つの新しい配列表記のリダクション組込み関数が追加されています。
  - __sec_reduce_and
  - __sec_reduce_or
  - __sec_reduce_xor

4.2.8 仮定されるデータ・アライメントを指定するポインターとポインター型の新しい属性

インテル® Composer XE 2013 SP1 では、__declspec(align_value(N))と __attribute__((align_value(N)))が追加されています。これらの属性が指定されたポインター型では、コンパイラーは指定されたアライメント “N” を仮定できます。次に例を示します。

```
typedef float float_a16
__attribute__((align_value (16)));

void foo(float_a16 *restrict dest, float_a16 *restrict src){
```

上記のコードは、src 引数と dest 引数がユーザーによって 16 バイト境界でアライメントされるべきであることをコンパイラーに知らせます。

4.2.9 フォルス・シェアリングを回避する変数定義属性

インテル® Composer XE 2013 SP1 では、__declspec(avoid_false_share)/__attribute__((avoid_false_share))と __declspec(avoid_false_share(identifier))/__attribute__((avoid_false_share(identifier)))が追加されています。ほかの変数とのフォルス・シェアリングを回避するため、これらの属性が指定された変数で、コンパイラーは適切なパディングを追加するか、アライメントします。identifier が指定されている場合、同じ identifier の変数を除くほかの変数とのフォルス・シェアリングを回避するため、identifier が指定された変数はパディングが追加されるか、アライメントされます。これらの属性は、関数スコープ、グローバルスコープ、名前空間スコープの変数定義で指定します。関数スコープで指定する場合、identifier の有効範囲は現在の関数になります。名前空間スコープやグローバルスコープで指定する場合、identifier の有効範囲は現在のコンパイル単位になります。

4.2.10 インテル® Composer XE 2013 SP1 の新しい __INTEL_COMPILER_UPDATE 事前定義マクロ

新しい__INTEL_COMPILER_UPDATE 事前定義マクロにより、使用しているインテル® コンパイラーのマイナー・アップデート番号を取得できるようになりました。例えば、バージョン 14.0.2 の場合、このマクロは “2” に前処理されます。

4.2.11 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。

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® Memory Protection Extensions (インテル® MPX) 向けの新しいコンパイラー・オプション /Qcheck-pointers-mpx (-check-pointers-mpx) (Update 1)

このオプションを指定すると、コンパイラーは、インテル® Memory Protection Extensions (インテル® MPX) を使用してポイントチェッカーのパフォーマンスを高速化するコードを生成します。ターゲット・プラットフォームがインテル® MPX をサポートしていない場合、ポインターチェッカーは通常の速度で動作します。インテル® MPX の詳細は、「Introduction to Intel® Memory Protection Extensions」 (英語) を参照してください。

新しいオプションの詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。

4.3.2 マルチプロセッサー通信環境 (MPC) 統合並列ランタイム向けにすべてのスタティック・データをプライベート化する新しいコンパイラー・オプション -f[no-]mpc_privatize(Update 1)

このオプションを指定すると、標準の Linux* ディストリビューションでサポートされていないスレッド・ローカル・ストレージ (TLS) を使用するランタイムルーチンが呼び出されます。このオプションは、MPC 統合並列ランタイムと併せて使用する場合のみ指定できます。デフォルトではオフ (`-fno-mpc_privatize`) に設定されています。

このオプションは、インテル® 64 およびインテル® MIC アーキテクチャー向けのインテル® C++/Fortran Composer XE 2013 SP1 Linux* 版でのみ利用できます。

詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。

4.3.3 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) 命令用の新しいコンパイラー・オプション /Q[a]xMIC-AVX512(-[a]xMIC-AVX512) (Update 1)

このオプションを指定すると、インテル® プロセッサー向けのインテル® AVX-512 の基本命令、競合検出命令、指数および逆数命令、プリフェッチ命令、および CORE-AVX2 で有効になる命令を生成します。インテル® AVX-512 命令対応のインテル® プロセッサー向けに最適化します。

詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。新しい命令の詳細は、「AVX-512 instructions」 (英語) を参照してください。

4.3.4 インテル® MIC アーキテクチャー向けの /Qopt-gather-scatter-unroll (-opt-gather-scatter-unroll) (Update 1)

このオプションを使用して、インテル® MIC アーキテクチャーの集約 (Gather) と分散 (Scatter) ループに対して別のループ・アンロール・シーケンスを指定できます。このオプションは、インテル® MIC アーキテクチャー向けインテル® 64 アーキテクチャーでのみ利用できます。

詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。

4.3.5 インテル® Composer XE 2013 SP1 の新規および変更されたコンパイラー・オプション

- -[no-]openmp-offload
- -[no-]openmp-simd
- -xATOM_SSE4.2
- -xATOM_SSSE3
- -vecabi=<arg>
- -gdwarf-4
- -standalone

- -offload=<arg>
- -mtune=<arch>
- -mlong-double-64
- -mlong-double-80

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。

4.3.6 -[no-]openmp-offload と -[no-]openmp-simd の追加 (インテル® Composer XE 2013 SP1)

この 2 つのオプションにより、OpenMP* 4.0 の TARGET 機能と SIMD 機能を、ほかの OpenMP* 機能 (-openmp で有効になる) とは別に有効/無効にできます。-openmp を指定すると、-openmp-offload と -openmp-simd も指定されるため、これらの機能を利用できます。

4.3.7 -mtune の追加 (インテル® Composer XE 2013 SP1)

-mtune=<arch>を使用して、gcc* の等価なオプションの動作と同様に、特定のアーキテクチャー向けにコンパイラーによるチューニングが可能になりました。

4.3.8 -gdwarf-4 の追加 (インテル® Composer XE 2013 SP1)

-gdwarf-4 オプションを指定して、DWARF V4 デバッグ情報を生成できるようになりました。

4.3.9 -vec-report7 の追加 (インテル® Composer XE 2013 Update 2)

インテル® Composer XE 2013 Update 2 では、ループのベクトル化についてより詳細で高度な情報を提供する新しいベクトル化レポートレベルが追加されています。詳細は、http://intel.ly/XeSkW6 (英語) を参照してください。

4.3.10 -gcc-version のサポート終了予定 (インテル® Composer XE 2013 Update 2)

-gcc-version の代わりのオプションとして -gcc-name が追加されました。-gcc-version は古いオプション (非推奨) です。将来のリリースでは削除される可能性があります。

4.4 その他の変更

4.4.1 KMP_DYNAMIC_MODE 環境変数による “asat” サポートの廃止

KMP_DYNAMIC_MODE 環境変数による “asat” (自動自己割り当てスレッド) のサポートが廃止されました。将来のリリースで削除される予定です。

4.4.2 インテル® Composer XE 2013 SP1 でインライン展開を有効にするには __attribute__((always_inline)) でインラインキーワードが必要

インテル® コンパイラーの以前のバージョンでは、”always_inline” 属性を指定して宣言されたルーチンは常にインライン展開されました。インテル® Composer XE 2013 SP1 でルーチンがインライン展開されるようにするには、ルーチンもインラインにする必要があります (”inline” キーワードを指定して明示的に宣言するか、クラス内でメンバー関数を定義して暗黙的にインラインにします)。コンパイラーの動作は gcc* と同じになり、次のような警告も出力します。

```
// t.cpp
__attribute__((always_inline)) int foo2(int x)  // "inline" キーワードも追加する必要がある
{
```

```
  return x;
}

icpc -c t.cpp
t.cpp(2): 警告 #3414: "always_inline" 属性はインライン展開されない関数では
無視されます。
__attribute__((always_inline)) int foo2(int x)
^
```

4.4.3 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。コンパイラー環境を設定すると、インテル® デバッガー (本製品とともに提供される GNU* GDB (gdb-ia))、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。

4.4.4 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。

4.5 既知の問題

4.5.1 __GXX_EXPERIMENTAL_CXX0X__マクロの未サポート

GNU* 4.8 以降の環境で、-std=c++11オプションまたは-std=gnu++0xオプションを使用すると、次のような診断が出力される可能性があります。

```
This file requires compiler and library support for the upcoming ISO
C++ standard, C++0x.This support is currently experimental, and must
be enabled with the -std=c++0x or -std=gnu++0x compiler options. (こ
のファイルには、新しい ISO C++ 標準規格である C++0x のコンパイラー・サポートと
ライブラリー・サポートが必要です。現在、これらのサポートは試験的に提供されており、
-std=c++0x コンパイラー・オプションまたは -std=gnu++0x コンパイラー・オプショ
ンを指定して有効にする必要があります。)
```

現在、インテル® コンパイラーでは、C++ 標準ライブラリーのヘッダーで __GXX_EXPERIMENTAL_CXX0X__マクロにより有効になるいくつかの C++11 機能をサポートしていないため、gcc 4.8 モードでこのマクロを定義していません。そのため、-std=c++11モードまたは-std=gnu++0xモードでC++ 標準ライブラリーを使用する場合、g++ と互換性の問題が発生する可能性があります。

4.5.2 ドキュメントに含まれていない 10 進浮動小数点のステータスをチェックする関数

10 進の浮動小数点演算中に発生する例外を検出するには、次の浮動小数点例外関数を使用します。

関数	説明
fe_dec_feclearexcept	サポートされている浮動小数点例外をクリアします。
fe_dec_fegetexceptflag	浮動小数点ステータスフラグの実装定義の表現を格納します。
fe_dec_feraiseexcept	サポートされている浮動小数点例外を発行します。
fe_dec_fesetexceptflag	浮動小数点ステータスフラグを設定します。
fe_dec_fetestexcept	指定されている浮動小数点例外フラグのサブセットのうち、現在設定されているものを特定します。

10 進の浮動小数点例外関数は、fenv.h ヘッダーファイルで定義されています。

バイナリー形式の浮動小数点例外関数については、ISO C99 で説明されています。

DFP を使用してソースをコンパイルするには、プリプロセッサー・マクロ __STDC_WANT_DEC_FP__を使用します。

4.5.3 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。

4.5.4 ガイド付き自動並列化の既知の問題

プログラム全体のプロシージャー間の最適化 (-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。

4.5.5 スタティック解析の既知の問題

4.5.5.1 仮想関数を含む C++ クラスに対する正しくないメッセージ

スタティック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。

5 GNU* GDB デバッガー

このセクションでは、インテル® Composer XE 2013 SP1 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。

5.1 機能

インテル® Composer XE 2013 SP1 とともに提供される GNU* GDB は、GDB 7.5 を拡張したものです。このデバッガーは、将来のリリースでインテル® デバッガーの代わりに使用される予定です。GDB 7.5 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
- 分岐トレーススストア (*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。)

5.2 必要条件

本製品とともに提供される GNU* GDB を使用するには、Python* 2.4、2.6、または 2.7 が必要です。

5.3 GNU* GDB の使用

インテル® Composer XE 2013 SP1 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:
  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。
- 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。
  - 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。
  - 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。ネイティブ・アプリケーションは、コマンドラインでデバッグする必要があります。

上記の GNU* GDB を使用するには、source コマンドで次のスクリプトを実行します。
source <install-dir>/bin/debuggervars.[sh|csh]

毎回デバッガーを使用する前に、必ず上記のコマンドを実行してください。

5.3.1 IA-32/インテル® 64 デバッガー

インテル® Composer XE とともに提供される GNU* GDB を起動するには、次のコマンドを使用します。
$ gdb-ia

このデバッガーは、ネイティブの IA-32 またはインテル® 64 対応アプリケーション用に設計されています。使用法は、従来の GNU* GDB デバッガーと同じです。ただし、一部拡張されています。詳細は、「ドキュメント」を参照してください。

5.3.2 インテル® Xeon Phi™ コプロセッサー・デバッガー

ホストとターゲットのさまざまな違いから、インテル® Xeon Phi™ コプロセッサー向けアプリケーションのデバッグは、ローカル・アプリケーションのデバッグとは異なります。GNU* GDB はホストで実行します。ホストシステムは、コプロセッサーとは別のマシンの場合もあれば、コプロセッサー・カードが搭載されているマシンの場合もあります。ターゲット (コプロセッサー) ではデバッグ・エージェント (gdbserver) を実行します。ホストはこのデバッグ・エージェントに接続します。

ホスト側でデバッグセッションを開始する方法は 2 つあります。

1. コマンドライン:
   最初に、次のようなコマンドを使用して、デバッグ・エージェントをインテル® Xeon Phi™ コプロセッサーに転送する必要があります。
   ```
   $ scp <install-dir>/debugger/gdb/target/mic/bin/gdbserver ↵
         mic0:/tmp
   ```

   次に、以下のようなコマンドを使用して、GNU* GDB を起動しコプロセッサーに接続します。
   ```
   $ gdb-mic
   (gdb) target extended-remote | ssh mic0 /tmp/gdbserver --multi
   –
   ```

   次のようなコマンドを使用して、コプロセッサーにアプリケーションをロードして実行します。
   ```
   (gdb) file <path_on_host>/application
   (gdb) set remote exec-file <path_on_target>/application
   ```

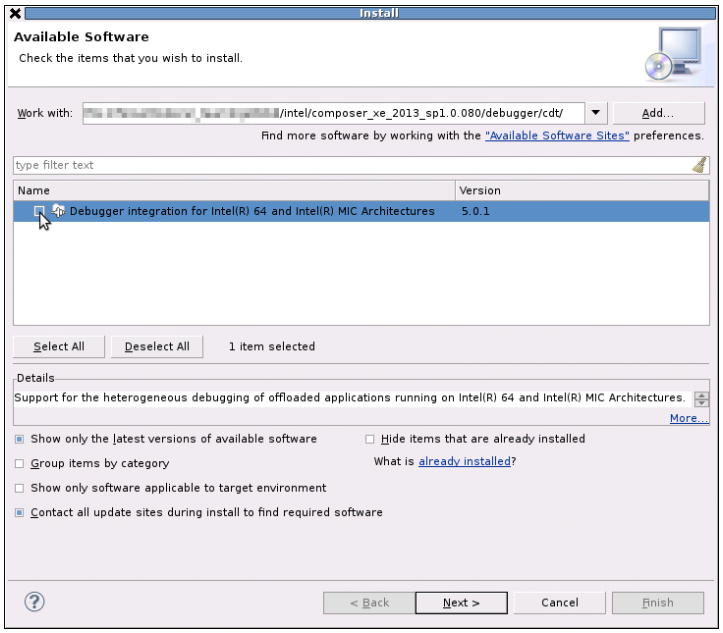
   次のようなコマンドを使用して、PID <pid>を指定してコプロセッサーで実行中のプロセスにアタッチします。
   ```
   (gdb) file <path_on_host>/application
   (gdb) attach <pid>
   ```

2. Eclipse* IDE:
   Eclipse* IDE が起動されている環境で、前述のように source コマンドで debuggervars.[sh|csh]スクリプトが実行されていることを確認します。

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をデバッグするには、Eclipse* IDE を起動する前に環境変数を設定する必要があります。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) のバージョンに応じて、次の変数を設定します。

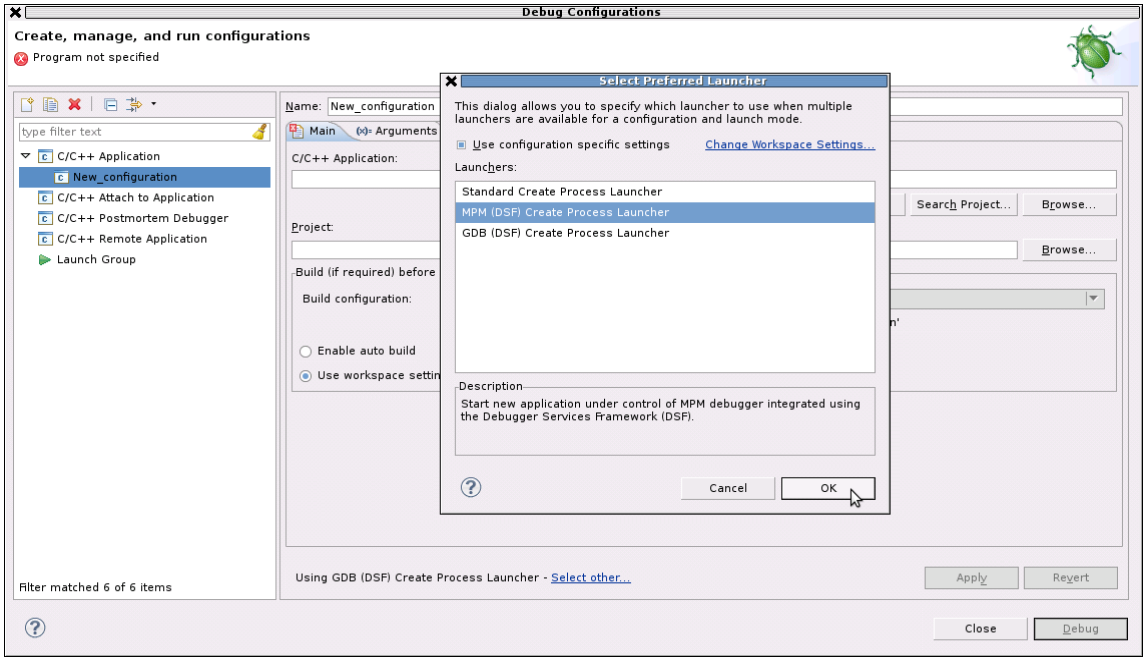
- インテル® MPSS 3.1:
  `AMPLXE_COI_DEBUG_SUPPORT=TRUE`
  `MYO_WATCHDOG_MONITOR=-1`
- インテル® MPSS 2.1:
  `COI_SEP_DISABLE=FALSE`
  `MYO_WATCHDOG_MONITOR=-1`

インテル® Xeon Phi™ コプロセッサー向けアプリケーションをデバッグするための新しい GNU* GDB を Eclipse* IDE に統合するには、プラグインをインストールする必要があります。プラグインは、`<install-dir>/debugger/cdt/`にあります。

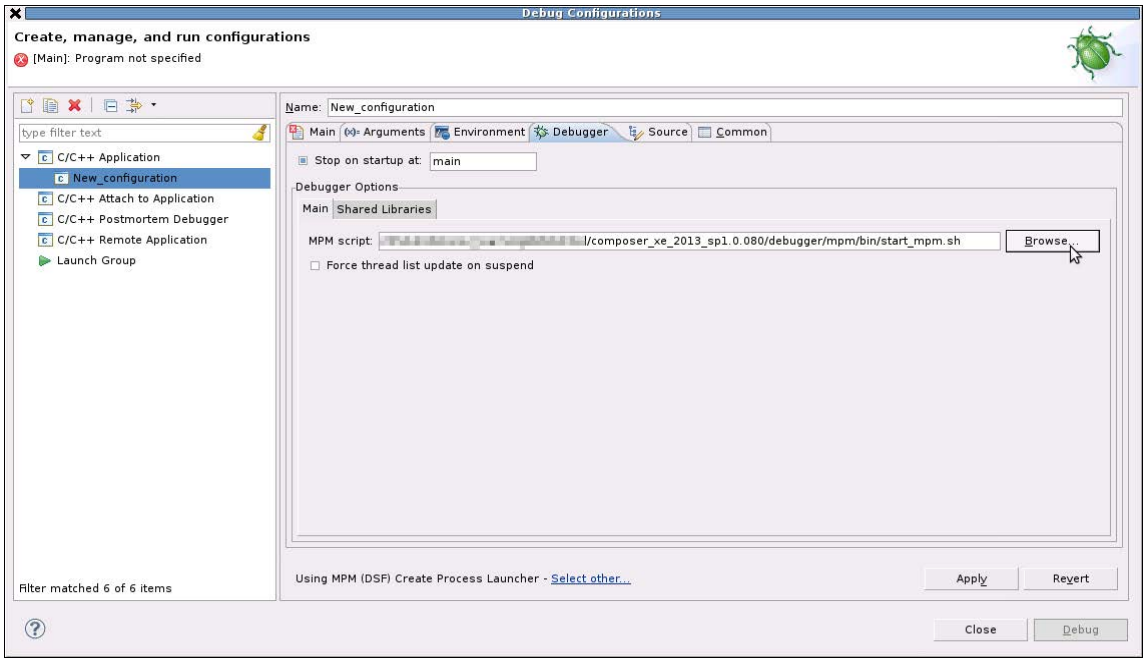


[Group items by category (項目をカテゴリー別にグループ化)] をオフにしてください。署名されていないコンテンツに関する警告が表示されることがありますが、これは無視してかまいません。

インストールし再起動したら、[C/C++ Application] 用の新しいデバッグ構成を作成し、[Select other...] をクリックして [MPM (DSF) Create Process Launcher] を選択します。



[Debugger] タブに移動して、次のスクリプトを選択します。
`<install-dir>/debugger/mpm/bin/start_mpm.sh`



注:
現在、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のみデバッグすることができます。

5.4 ドキュメント

インテル® Composer XE 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/mic/eclmigdb_
config_guide.pdf
```

5.5 既知の問題と変更点

5.5.1 Eclipse* プラグインによるオフロードデバッグがインテル® 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) 3.2 で動作しない

インテル® Composer XE 2013 SP1 以降、Eclipse* プラグインによるオフロードデバッグはインテル® MPSS 3.2 で動作しません。インテル® Composer XE 2013 SP1 パッケージで必要な設定ファイルがインテル® MPSS 3.2 で削除されたことが原因です。インテル® MPSS の以前のバージョンでは、この問題はありません。これは、インテル® MPSS の将来のバージョンで修正される予定です。

5.5.2 MYO デバッグ・ライブラリーがインテル® 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) 3.2 でデフォルト・インストールされない

MYO デバッグ・ライブラリーは、インテル® MPSS 3.2 ではデフォルトでインストールされなくなりました。(インテル® MPSS の以前のバージョンではデフォルトでインストールされていました。)オフロードデバッグ用の Eclipse* プラグインを使用して MYO 対応アプリケーションをデバッグするには、MYO デバッグ・ライブラリーを手動でインストールする必要があります。インテル® MPSS 3.2 以降、MYO デバッグ・ライブラリーは `mpss-*.tar` ファイルのパッケージ `mpss-myo-dbg-*`に含まれています。

5.5.3 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。

5.5.4 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。

5.5.5 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)
}
```

6 インテル® デバッガー (IDB)

インテル® デバッガー (IDB) は、IA-32 およびインテル® 64 対応アプリケーション並びにインテル® Xeon Phi™ コプロセッサーのホストデバッガーとして利用できます。

6.1 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。これは、このセクションで説明されているすべてのコンポーネントと機能が対象となります。
代わりに GNU* GDB デバッガー・コンポーネントを使用してください。

6.2 インテル® デバッガーの使用

インテル® Composer XE 2013 SP1 とともに提供されるインテル® デバッガー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:
  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。
- 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。
  - 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。
  - 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とインテル® Xeon Phi™ コプロセッサーのネイティブ・アプリケーションの両方で利用できます。

上記のインテル® デバッガーを使用するには、source コマンドで次のスクリプトを実行します。

```
source <install-dir>/bin/idbvars.[sh|csh] [ia32|intel64]
```

使用するアーキテクチャーに応じて、ia32 (IA-32 の場合) または intel64 (インテル® 64 の場合) を選択します。毎回デバッガーを使用する前に、必ず上記のコマンドを実行してください。

6.2.1 IA-32/インテル® 64 デバッガー

インテル® Composer XE とともに提供されるインテル® デバッガーを起動するには、次のコマンドを使用します。

```
$ idb (スタンドアロン GUI の場合)
```

または

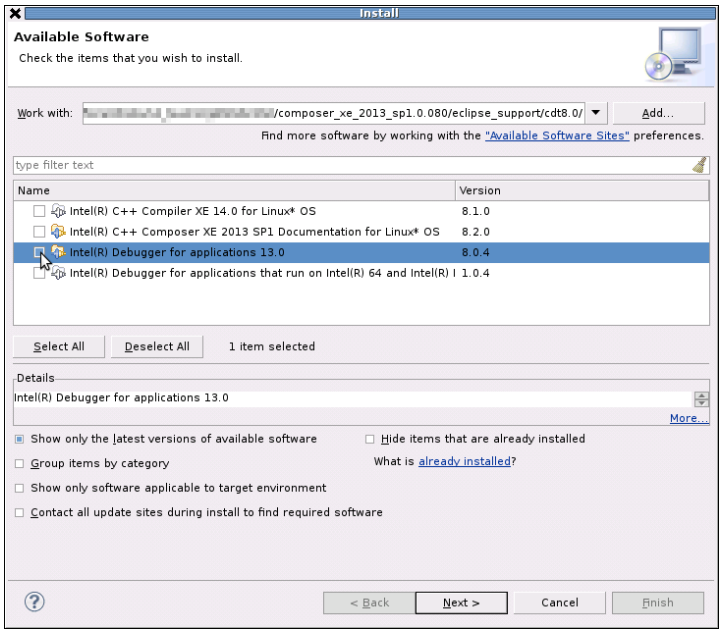
```
$ idbc (コマンドラインの場合)
```

このデバッガーは、ネイティブの IA-32 またはインテル® 64 対応アプリケーション用に設計されています。関連ドキュメントについては、「ドキュメント」セクションを参照してください。

注:

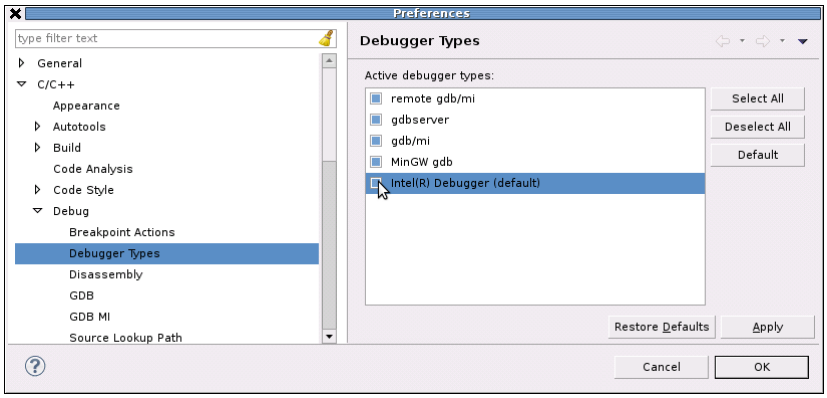
インテル® デバッガーが起動されている環境で、前述のように source コマンドで idbvars.[sh|csh]スクリプトが実行されていなければなりません。

スタンドアロン GUI では、Eclipse* IDE は必要ありません。ただし、Eclipse* IDE 用のプラグインも提供されています。プラグインは<install-dir>/eclipse_support/cdt8.0/ にあります。

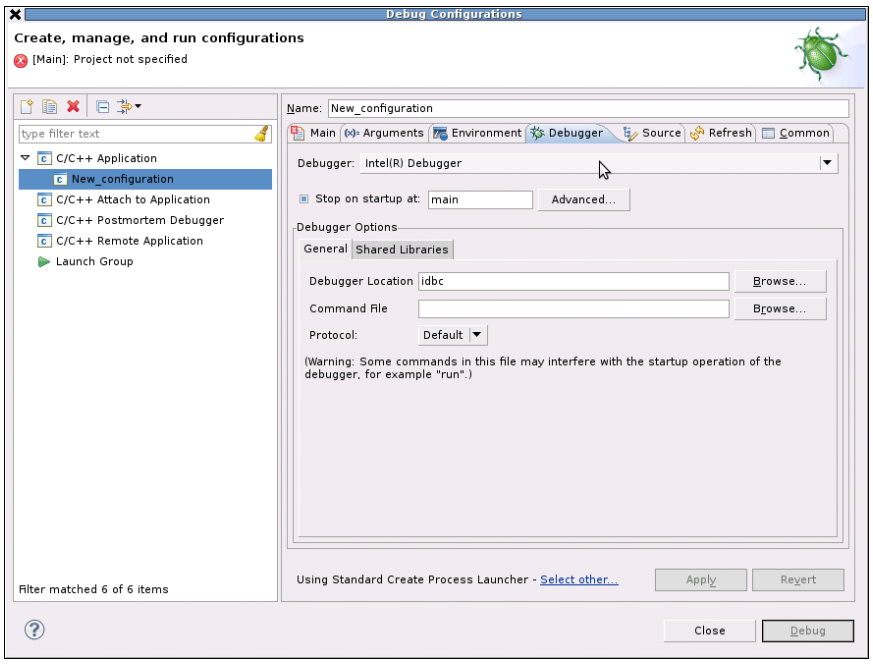


[Group items by category (項目をカテゴリー別にグループ化)] をオフにしてください。署名されていないコンテンツに関する警告が表示されることがありますが、これは無視してかまいません。

使用している Eclipse* IDE のバージョンによっては、デフォルトのデバッガー統合をインテル® デバッガーに切り替えることができます。例えば、[Window] > [Preferences] メニューから切り替えられます。



または、[Run] > [Debug Configurations...] で新しいデバッグ構成を作成することもできます。



デバッガー統合は、[Standard Create Process Launcher] または [Standard Attach to Process Launcher] が選択されている場合のみ変更することができます。

6.2.2 インテル® Xeon Phi™ コプロセッサー・デバッガー

ホストとターゲットのさまざまな違いから、インテル® Xeon Phi™ コプロセッサー向けアプリケーションのデバッグは、ローカル・アプリケーションのデバッグとは異なります。インテル® デバッガーはホストで実行します。ホストシステムは、コプロセッサーとは別のマシンの場合もあれば、コプロセッサー・カードが搭載されているマシンの場合もあります。ターゲット (コプロセッサー) ではデバッグ・エージェント (`idbserver_mic`) を実行します。ホストはこのデバッグ・エージェントに接続します。

先に進む前に、デバッガーを使用している環境で、`source` コマンドで `idbvars.[sh|csh]`スクリプトを実行します。

ホスト側でデバッグセッションを開始する方法は 2 つあります。

1. コマンドライン:
   次に、以下のようなコマンドを使用して、インテル® デバッガーを起動しコプロセッサーに接続します。

```
$ idbc_mic
```

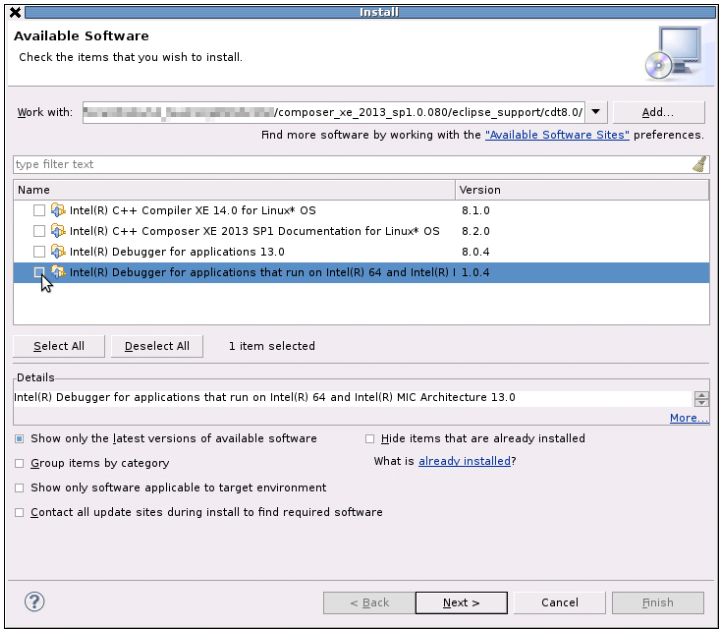
次のようなコマンドを使用して、コプロセッサーにアプリケーションをロードして実行します。

```
(idb) idb file-remote <path_on_target>/application
(idb) file <path_on_host>/application
```

次のようなコマンドを使用して、PID <pid>を指定してコプロセッサーで実行中のプロセスにアタッチします。

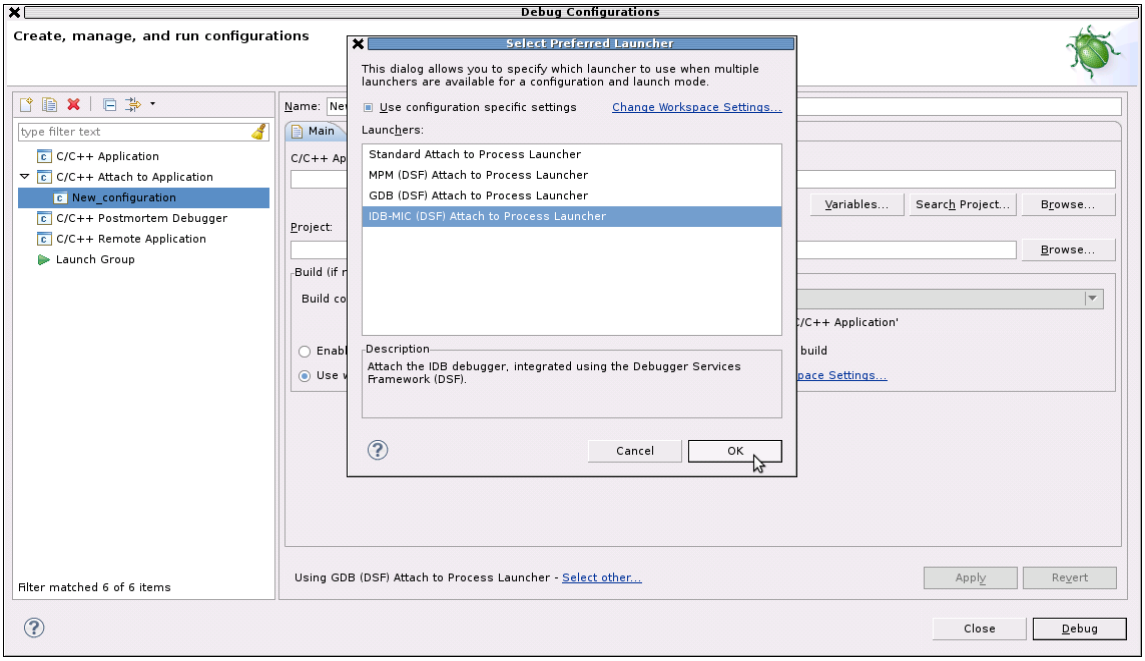
```
(idb) attach <pid><path_on_host>/application
```

2. Eclipse* IDE:
   インテル® Xeon Phi™ コプロセッサー向けアプリケーションをデバッグするための IDB を Eclipse* IDE に統合するには、プラグインをインストールする必要があります。プラグインは<install-dir>/eclipse_support/cdt8.0/にあります。



[Intel® Debugger for applications that run on Intel® 64 and Intel® MIC Architecture] パッケージをインストールします。
[Group items by category (項目をカテゴリー別にグループ化)] をオフにしてください。署名されていないコンテンツに関する警告が表示されることがありますが、これは無視してかまいません。

インストールし再起動したら、[C/C++ Application] または [C/C++ Attach to Application] 用の新しいデバッグ構成を作成し、[Select other...] をクリックして [MPM (DSF) Create Process Launcher] を選択します。



インテル® Xeon Phi™ コプロセッサー向けアプリケーションにアタッチする場合は、[Debugger] タブの [MIC options] でカードを選択する必要があります。

6.3 Java* ランタイム環境の設定

インテル® デバッガー (idb) のスタンドアロン GUI を実行するには、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。そして、$PATH に `java` が追加されていることを確認します。

注:
`source` コマンドで実行した `idbvars.[sh|csh]`スクリプトのアーキテクチャーと JRE のアーキテクチャーが同じでなければなりません。アーキテクチャーが異なる場合、IDB は起動しません (例えば、64 ビットの JRE で 32 ビットの IDB は起動できません)。

6.4 ドキュメント

インテル® デバッガーのドキュメントは、以下の場所にあります。
`<install-dir>/Documentation/[en_US|ja_JP]/debugger/`

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。

6.5 デバッガー機能

6.5.1 IDB の主な機能

インテル® デバッガーのスタンドアロン GUI は、コマンドラインと同じ機能をすべてサポートします。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。

6.5.2 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。

6.6 既知の問題と変更点

6.6.1 インテル® MPSS でのインテル® デバッガーの使用

インテル® MPSS でインテル® MIC アーキテクチャー対応インテル® デバッガーを使用する場合は、次の制限が適用されます。

- コマンドラインでネイティブ・コプロセッサー・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 がコプロセッサー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=tcpip:<cardip>:<port>
  ```

6.6.2 Eclipse* IDE 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* IDE で debuggee のコマンドライン引数を正しく設定できません。debugee は次のメッセージを出力してアボートします。

```
*** abort -internal failure : get_command_argument failed
```

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。

6.6.2.1 Eclipse IDE 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* IDE 環境でローカル変数を表示できません。

回避方法:

[Expression] ウィンドウにローカル変数を入力して値を取得します。

6.6.3 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。

6.6.4 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。

6.6.5 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。

6.6.6 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。

6.6.7 ウォッチポイントの制限

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ー領域のサイズは、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ー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ー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いてはなりません。
- 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ー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ー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。

6.6.8 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。

6.6.9 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。

6.6.10 [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。

6.6.11 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。

6.6.12 $cdir ディレクトリー、$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。

6.6.13 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。

6.6.14 $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
```

6.6.15 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。

6.6.16 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。

6.6.17 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。

6.6.18 [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。

6.6.19 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。

6.6.20 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
```

6.6.21 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

7 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。

7.1 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 4.2、3.8、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 以降が必要です。

7.1.1 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。

7.2 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 14.0 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。

[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 14.0 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。

7.2.1 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。

7.2.2 Eclipse* へのインテル® デバッガーの統合

「インテル® デバッガー (IDB)」セクションを参照してください。

7.3 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。

7.3.1 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。

7.4 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/compilervars.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
```

7.5 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。

7.6 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、14.0 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、14.0 がサポートされています。

8 インテル® インテグレーテッド・パフォーマンス・プリミティブ

このセクションでは、インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) のこのバージョンでの変更点、新機能、および最新情報をまとめています。

インテル® IPP 8.0 の最新情報は、`<install dir>/composer_xe_2013_sp1.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/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/OG5IF7) のドキュメントのリンクを参照してください。

8.1 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。

8.2 インテル® IPP コードサンプル

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

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

9 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、http://intel.ly/OeHQqf を参照してください。

9.1 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 では、インストールするコンポーネントを選択できるようになりました。PGI* コンパイラー、Compaq* Visual Fortran コンパイラー、SP2DP インターフェイス、BLAS95 および LAPACK95 インターフェイス、クラスターサポート (ScaLAPACK および Cluster DFT)、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に必要なコンポーネントは、インストール時に明示的に選択しない限りインストールされません。
- インテル® MKL クラスター・コンポーネント (ScaLAPACK および Cluster DFT) では、アライメントされていない条件付き数値再現性 (CNR) は利用できません。
- Boost* uBLAS および Java* でのインテル® MKL の使用例は、製品パッケージからは削除され、以下の記事 (英語) からダウンロードすることができます。
  - How to use Intel MKL with Java*
  - How to use Boost* uBLAS with Intel MKL

9.2 本バージョンでの変更

9.2.1 インテル® MKL 11.1 Update 3 の新機能

- 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レベル 3 BLAS 関数のパフォーマンスが向上
  - 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
  - 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) 命令セット用の ?GEMM、?TRSM、DTRMM を最適化
- LAPACK:
  - ?(SY/HE)RDB 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
  - 固有ベクトルが必要な場合の ?SY(EV/EVD) 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
  - インテル® AVX2 対応のプロセッサーにおいて DGETRF のパフォーマンスが向上
  - 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 ?SYRDB の自動オフロードを追加され、固有ベクトルが不要な場合に DSY(EV/EVD) がスピードアップ
- スパース BLAS:
  - 対角行列のサンプルを追加
- インテル® MKL PARDISO:
  - レンダリング・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
  - 非対称行列および OOC 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 OOC モードに対角抽出のサポートを追加
- 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® 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) 命令セット用 FFT の全範囲を最適化
  - 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 FFT のパフォーマンスが向上

9.2.2 インテル® MKL 11.1 Update 2 の新機能

- インテル® Atom™ プロセッサーのサポートを追加
- BLAS:
  - すべてのインテル® アーキテクチャーにおいて、m==1 または n==1 の ?GEMM のパフォーマンスが向上
  - 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・ベースのシステムにおいて MP LINPACK 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、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 ?SYMM のパフォーマンスが向上
    - 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® 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、小行列の {S/D}GEMM シングルスレッド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 DGEMV のパフォーマンスが向上
    - すべてのインテル® アーキテクチャーにおいて、notrans:n>>m および trans:m>>n の {S,D,C,Z}GEMV のスレッド化パフォーマンスが向上
    - 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 DSYR2K のパフォーマンスが向上
    - 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、小行列 (行列サイズ<= 10) の DTRMM のパフォーマンスが向上
    - ZHEMM および ZSYRK のスタック使用が減少
    - 未サポート構成でオフロード MP LINPACK スクリプトを実行したときにより詳細なエラーメッセージを追加
- LAPACK:
    - 固有ベクトルが必要な場合、大きな次元および UPLO=L で (S/D)SYRDB および (S/D)SYEV のパフォーマンスが向上
    - 劣決定の場合の ?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
    - DSYRDB UPLO=L の場合にインテル® Xeon Phi™ コプロセッサーへの自動オフロードを追加
- スパース BLAS:
    - インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令セット用の SpMV カーネルを最適化
    - インテル® ストリーミング SIMD 拡張命令 4.2 (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
- PARDISO:
    - レンダリング・アルゴリズムのアウトオブコア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
- 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:
    - インテル® Xeon Phi™ コプロセッサーにおいて 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:
    - プールされた/グループ化された (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) 平均推定の計算をサポート

9.2.3 インテル® MKL 11.1 Update 1 の新機能

- インテル® AVX-512 命令セットのサポート (特定の最適化のみ)
- BLAS:
    - インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 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 2 (インテル® AVX2) 対応のすべての 64 ビットのインテル® プロセッサーにおいて、DSDOT のパフォーマンスが向上し、マルチスレッドをサポート

    - *TRSM で対角行列のデノーマル数の処理が向上
    - 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、小さな N と大きな M および K で SGEMM のパフォーマンスが向上
    - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて *HEMM の並列パフォーマンスが向上
    - インテル® SSSE3 以降対応のすべてのインテル® プロセッサーにおいて 64 ビットの *SYRK/*HERK の並列パフォーマンスが向上
    - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて 64 ビットの {D,S}SYRK のシリアル・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 DTRSM のパフォーマンスが向上
    - インテル® AVX 対応インテル® プロセッサー向けインテル® Optimized HPL Benchmark の runmultiscript 機能を拡張
    - インテル® MIC アーキテクチャーにおいてインテル® Optimized HPL Benchmark のパフォーマンスが向上
- LAPACK:
    - 並列 LAPACK 関数 (OR/UN)M(QR/RQ/QL/LQ) のメモリー使用率が減少
    - LAPACK 関数のスタックメモリー使用率が減少
    - 固有値のみ必要な場合、大きな次元で (S/D)SYRDB および (S/D)SYEV のパフォーマンスが向上
- ScaLAPACK:
    - デフォルトの NETLIB 複素数型と MKL 複素数型が混在できるように PBLAS ヘッダーを更新
- DFT: 複素数-複素数および実数-複素数の変換を最適化
- 転置: 縦長の行列と横長の行列で mkl_?omatcopy ルーチンのパフォーマンスが向上
- DFTI インターフェイスと FFTW ラッパーがスレッドセーフになり、並列領域から MKL DFT を使用する場合 NUMBER_OF_USER_THREADS パラメーターは任意設定に変更

9.2.4 インテル® MKL 11.1 の新機能

- 条件付きの数値再現性: アライメントされていないデータで条件付き数値再現性 (CNR) モードをサポート
- ノードごとにプロセッサーの種類や搭載されているインテル® Xeon Phi™ コプロセッサーの数が異なるヘテロジニアス・クラスターで MP LINPACK をサポート
- 最近の AMD* システムにおいて CNR=AUTO モードのパフォーマンスが向上
- BLAS:
    - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて [S/D]GEMV のパフォーマンスが向上
    - 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) において [D/Z]GEMM および倍精度のレベル 3 BLAS 関数を最適化
    - 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 において [Z/C]AXPY および [Z/C]DOT[U/C] を最適化
    - インテル® MIC アーキテクチャーにおいて DTRMM のシーケンシャル・バージョンを最適化
    - インテル® AVX2 において DAXPY をチューニング
- LAPACK:
    - 固有値のみ必要な場合、大きな次元で (S/D)SYRDB および (S/D)SYEV のパフォーマンスが向上
    - M,N<10 のような小さなサイズで xGESVD のパフォーマンスが向上。

- VSL:
  - 平均絶対偏差のサポートとサンプルの追加
  - alpha=1 の場合のワイブル乱数ジェネレーター (RNG) のパフォーマンスが向上
  - 外積および平均絶対偏差の行列において、次数 4 までのローデータおよび中央部の統計的総和をサポート
  - S. Joe および F. Y. Kuo により設計された、最大 21,201 次元まで発生できるソボル QRNG の使用法を示す VSL サンプルを追加
  - インテル® MIC アーキテクチャーにおいて SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
- DFT:
  - インテル® MIC アーキテクチャーにおいて倍精度の複素数-複素数変換のパフォーマンスが向上
  - インテル® AVX2 において複素数-複素数 DFT を最適化
  - インテル® Xeon® プロセッサー E5 v2 ファミリーにおいて 2 次元の複素数-複素数 DFT を最適化
  - インテル® Xeon® プロセッサー E5 ファミリー (インテル® AVX) およびインテル® AVX2 において GENE アプリケーション固有のワークロードでパフォーマンスが向上
  - DFTI 計算関数のドキュメントのデータレイアウトが向上
  - 大規模な実数-複素数 FFT のスケーリング
- データ・フィッティング:
  - インテル® Xeon® プロセッサーおよびインテル® MIC アーキテクチャーにおいて df?Interpolate1D および df?SearchCells1D 関数のパフォーマンスが向上
  - インテル® MIC アーキテクチャー、インテル® Xeon® プロセッサー X5570、インテル® Xeon® プロセッサー E5-2690 において、線形および 3 次 Hermite/Bessel/Akima スプライン用 df?construct1d 関数のパフォーマンスが向上
- 転置
  - 正方行列でインプレース転置のパフォーマンスが向上
- インストール時間を短縮するためパッケージに含まれるインテル® MKL のサンプルとテストをアーカイブ

9.3 権利の帰属

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

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

インテル® MKL クラスター・エディションの一部の基となった ScaLAPACK の原版は http://www.netlib.org/scalapack/index.html (英語) から入手できます。ScaLAPACK の開発は、L.

S. Blackford、J. Choi、A. Cleary、E. D'Azevedo、J. Demmel、I. Dhillon、J. Dongarra、S. Hammarling、G. Henry、A. Petitet、K. Stanley、D. Walker、R. C. Whaley らによって行われました。

インテル® MKL 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。

10 インテル® スレッディング・ビルディング・ブロック

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

11 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。

本書で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、http://www.intel.com/design/literature.htm (英語) を参照してください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ を参照してください。

インテル® C++ コンパイラー、インテル® デバッガー、インテル® インテグレーテッド・パフォーマンス・プリミティブ、インテル® マス・カーネル・ライブラリー、およびインテル® スレッディング・ビルディング・ブロックは、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、Itanium、Pentium、Xeon、Intel Xeon Phi、Cilk 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。