インテル® Fortran 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: 321415-004JA
2013 年 3 月 5 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。

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 製品アップデート

Update 3 – 2013 年 3 月

- インテル® Fortran コンパイラーが 13.1.1 にアップデート
- インテル® マス・カーネル・ライブラリーが 11.0 Update 3 にアップデート
- 報告された問題の修正
  - コンパイラーの修正リスト
  - インテル® MKL の修正リスト

Update 2 – 2013 年 1 月

- インテル® Fortran コンパイラーが 13.1.0 にアップデート
  - OpenMP* 4.0 の追加の宣言子、節およびプロシージャーをサポート
  - KMP_PLACE_THREADS ランタイム環境変数を追加
  - Co-Array アプリケーションのパフォーマンスを向上できるオプションを追加 (Update 1)
  - -vec-report7 コンパイラー・オプションを追加
  - 読み取り中の書式なしシーケンシャル・ファイルのバッファリングの変更についての説明を追加
  - OFFLOAD 宣言子を使用してコンパイルされたモジュールファイルの命名規則についての説明を追加
  - 連続していない部分配列がオフロード領域に渡されない制限についての注意を追加
- インテル® マス・カーネル・ライブラリーが 11.0 Update 2 にアップデート
- 報告された問題の修正
  - コンパイラーの修正リスト: http://intel.ly/S5ulAb
  - インテル® MKL の修正リスト: http://intel.ly/S5uw3R

Update 1 – 2012 年 10 月

- インテル® Fortran コンパイラーが 13.0.1 にアップデート
  - 派生型の ALLOCATABLE または POINTER コンポーネントでの ATTRIBUTES ALIGN 宣言子の指定をサポート
- インテル® マス・カーネル・ライブラリーが 11.0 Update 1 にアップデート
- ドキュメントに -fimf-domain-exclusion の説明を追加
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー用オフロード・ライブラリーの Update 1 でのバイナリー変更に伴う互換性問題
- インテル® Xeon Phi™ コプロセッサーのステッピング A 用にコンパイルする場合の -opt-streaming-stores never オプションの使用についての注意を追加
- 報告された問題の修正
  - コンパイラーの修正リスト: http://intel.ly/S5ulAb (英語)
  - インテル® MKL の修正リスト: http://intel.ly/S5uw3R (英語)

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

- 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コプロセッサー (インテル® Xeon Phi™ 製品ファミリー) 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ー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® Xeon Phi™ コプロセッサー」セクションを参照してください。
- インテル® Fortran 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予定
- 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® MKL でインテル® Pentium® III プロセッサーのサポートが終了。インテル® MKL でサポートされる最小の命令セットはインテル® SSE2 になります。詳細は、「インテル® MKL」セクションを参照してください。
- SUSE* Linux* Enterprise Server 11 SP2 のサポートを追加。SP1 はサポートされなくなりました。
- Fedora* 17、Ubuntu* 12.04、Ubuntu* 11.10 のサポートを追加
- 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が終了:
  - Asianux*
  - Red Hat* Enterprise Linux* 4
  - Fedora* 15
  - Ubuntu* 11.04
- 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
- 報告された問題の修正

1.2 製品の内容

インテル® Fortran Composer XE 2013 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® Fortran コンパイラー XE 13.1.1。Linux* オペレーティング・システムを実行する IA-32、インテル® 64 アーキテクチャー・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。
- インテル® デバッガー 13.0
- インテル® マス・カーネル・ライブラリー 11.0 Update 3
- 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® Xeon Phi™ コプロセッサーに作業をオフロード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) をインストールする必要があります。

- 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)
- インテル® Xeon Phi™ コプロセッサーの開発/テスト用:
    - 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)
- IA-32 対応アプリケーションまたはインテル® 64 対応アプリケーションを開発する場合は、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。)
    - Debian* 6.0
    - Fedora* 17
    - Red Hat* Enterprise Linux* 5、6
    - SUSE* Linux* Enterprise Server 10、11 SP2
    - Ubuntu* 11.10、12.04
    - 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

- IA-32 アーキテクチャー・システムまたはインテル® 64 アーキテクチャー・システム
- Java* ランタイム環境 (JRE) 6.0 (1.6)
    - 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。

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。インテル® MKL では最小命令セットとしてインテル® SSE2 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。
- 一部の最適化オプションには、アプリケーションを実行するプロセッサーの種類に関する制限があります。詳細は、オプションの説明を参照してください。

1.4 ドキュメント

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

1.5 最適化に関する注意事項

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

1.6 日本語サポート

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

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

日本語サポート版を英語のオペレーティング・システムで使用する場合や日本語のオペレーティング・システムで英語サポート版を使用する場合は、http://intel.ly/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・センターでラインセンスを登録してください。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。

2.1 クラスターでのインストール

インストールするマシンにインテル® Cluster Studio XE のライセンスがあり、クラスターメンバーの場合、そのクラスターの複数のノードに製品をインストールすることができます。

複数のノードにインストールするには、次の手順に従います。

1. 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します。
2. インストールのステップ 4 (オプション) で、「クラスター・インストール」を選択します。
3. クラスターノードの IP アドレス、ホスト名、完全修飾ドメイン名 (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` ファイルが見つかると、"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。オプションを選択します。
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。

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

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) は、インテル® Composer XE 2013 Linux* 版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ーのインストールに必要な手順については、インテル® MPSS のドキュメントを参照してください。

2.3 インテル® Software Manager

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

将来の製品設計の参考のため、製品使用状況に関する匿名情報をインテルに提供する、インテル® ソフトウェア向上プログラムに参加できます。このプログラムは、デフォルトで無効になっていますが、インストール中または後から有効にして参加できます。参加はいつでも取りやめることができます。詳細は、http://intel.ly/SoftwareImprovementProgram (英語) を参照してください。

2.4 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/oPEdEe (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 FLEXlm* ライセンス・マネージャーに関する情報も記述されています。

2.6 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。

2.7 インストール先フォルダー

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

<install-dir> 以下には次のサブディレクトリーがあります。

- bin - インストールされている最新バージョンの実行ファイルへのシンボリック・リンク
- lib - インストールされている最新バージョンの lib ディレクトリーへのシンボリック・リンク
- include - インストールされている最新バージョンの include ディレクトリーへのシンボリック・リンク
- man - インストールされている最新バージョンの man ページが含まれているディレクトリーへのシンボリック・リンク
- 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
- 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 製品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 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 製品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
- mkl - mkl 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
- 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
- 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
- eclipse_support - インテル® Fortran コンパイラーとインテル® C++ コンパイラーで共有されるインテル® デバッガーにより作成されるディレクトリーへのシンボリック・リンク。インテル® Fortran コンパイラーでは Eclipse* をサポートしていません。

各 composer_xe_2013.<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 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
- compiler - 共有ライブラリーとインクルード/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
- eclipse_support - インテル® Fortran コンパイラーとインテル® C++ コンパイラーで共有されるインテル® デバッガーにより作成されるディレクトリー。インテル® Fortran コンパイラーでは Eclipse* をサポートしていません。
- man - man ページ
- 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

- `mpirt` – Fortran Co-Array サポートに使用されるインテル® MPI ライブラリーのランタイムファイル
- `Samples` – サンプルプログラムとチュートリアル・ファイル

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 製品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製品が使用されます。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。

2.8 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® C++ コンパイラーをインストールしている場合は、C++ コンパイラーもリストに表示されます。

3 インテル® Fortran コンパイラー

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

3.1 互換性

一般に、インテル® Fortran コンパイラー Linux* 版の以前のバージョン (8.0 以降) でコンパイルされたオブジェクト・コードおよびモジュールは、バージョン 13 でもそのまま使用できます。ただし、次の例外があります。

- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた CLASS キーワードを使用して多相変数を宣言しているソースは再コンパイルする必要があります。
- マルチファイルのプロシージャー間の最適化 (-ipo) オプションを使用してビルドされたオブジェクトは再コンパイルする必要があります。
- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた REAL(16)、REAL*16、COMPLEX(16)、COMPLEX*32 データ型を使用しているオブジェクトは再コンパイルする必要があります。
- バージョン 10.0 よりも前のコンパイラーを使用してインテル® 64 アーキテクチャー用にビルドされたモジュール変数を含むオブジェクトは再コンパイルする必要があります。Fortran 以外のソースからこれらの変数を参照する場合、不正な先頭の下線を削除するように外部名を変更する必要があります。
- バージョン 11.0 よりも前のコンパイラーを使用してコンパイルされた、派生型宣言の外部で ATTRIBUTES ALIGN 宣言子を指定したモジュールは再コンパイルする必要があります。この問題が発生した場合、問題を通知するメッセージが表示されます。
- 派生型宣言の内部で ATTRIBUTES ALIGN 宣言子を指定したモジュールは 13.0.1 以前のコンパイラーでは使用できません。

3.1.1 REAL(16) および COMPLEX(16) データ型のスタック・アライメントの変更

コンパイラーのバージョン 12.0 以前は、REAL(16) または COMPLEX(16) (REAL*16 または COMPLEX*32) 項目が値で渡されたとき、スタックアドレスは 4 バイトでアラインされていました。パフォーマンスを向上させるため、バージョン 12 (以降) では、コンパイラーはこれらの項目を 16 バイトでアラインします。引数は 16 バイト境界でアラインされます。この変更は、gcc とも互換です。

この変更は、主にライブラリーが生成した REAL(16) 値の計算を行うライブラリー (組込み関数を含む) の呼び出しに影響します。以前のバージョンでコンパイルしたコードをバージョン 12 のライブラリーとリンクする場合、またはアプリケーションをインテルのランタイム・ライブラリーの共有バージョンにリンクする場合、正しくない結果が返される可能性があります。

コンパイラーのバージョン 12.0 以前でコンパイルされている場合、この問題を回避するには、REAL(16) および COMPLEX(16) データ型を使用しているすべての Fortran ソースを再コンパイルしてください。

3.2 新機能と変更された機能

3.2.1 Fortran 2003 の機能

- 多相変数のデフォルトの初期化
- 外部プロシージャーを参照する場合、汎用インターフェイス・ブロックの MODULE PROCEDURE からキーワード MODULE を省略

3.2.2 Fortran 2008 の機能

- ATOMIC_DEFINE および ATOMIC_REF

3.2.3 インテル® メニー・インテグレーテッド・コア (インテル® MIC)

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コプロセッサー (インテル® Xeon Phi™ 製品ファミリー) に作業をオフロードするアプリケーションの開発をサポートしました。次の機能が追加されました。

- ATTRIBUTES OFFLOAD 宣言子
- OFFLOAD および END OFFLOAD 宣言子
- OMP OFFLOAD 宣言子
- 組込み関数 OFFLOAD_GET_DEVICE_NUMBER および OFFLOAD_NUMBER_OF_DEVICES
- サンプルプログラム (samples/mic_samples)

インテル® MIC を使用するアプリケーションの作成、ビルド、実行についての詳細は、インテル® Xeon Phi™ コプロセッサーおよびコンパイラーのドキュメントを参照してください。

3.2.4 OFFLOAD 宣言子を使用してコンパイルされたモジュールファイルの命名規則

インテル® Xeon Phi™ コプロセッサーに作業をオフロードするため OFFLOAD 宣言子を使用した Fortran ソースは、ネイティブ実行用とコプロセッサーでの実行用に 2 回コンパイルされます。ソースにモジュール・サブプログラムが含まれる場合も、同様に 2 回コンパイルされます。モジュールソースに実行ターゲットへの依存性がある場合、コンパイルされたモジュールファイルは異なります。しかし、同じファイル命名規則が使用された場合、インテル® MIC アーキテクチャー用にコンパイルされたモジュールファイルが、ネイティブコード用にコンパイルされたモジュールファイルに上書きされます。

インテル® Fortran コンパイラーは、モジュールのソースファイルにオフロード領域が含まれている場合、デフォルトの .mod ファイルではなく .modmic ファイルを使用してインテル® MIC アーキテクチャー用のモジュールファイルを生成することでこの問題を解決しています。

インテル® MIC アーキテクチャー用にコンパイルするとき、コンパイラーは最初に .modmic ファイルを探し、このファイルが存在しない場合は .mod ファイルを探します。このような処理が適切に行われるように、メイクファイルまたはビルドスクリプトを調整する必要があります。

3.2.5 新しい宣言子と追加された宣言子

インテル® Composer XE 2013 では、次のコンパイラー宣言子が追加、変更されています。詳細は、ドキュメントを参照してください。

- ATTRIBUTES CVF
- ATTRIBUTES OFFLOAD
- OFFLOAD/END OFFLOAD
- ORDERED/END ORDERED
- SIMD VECTORLENGTHFOR

3.2.6 OpenMP* の変更

インテル® Composer XE 2013 では、OpenMP* サポートに関して次の点が変更されています。

- OMP OFFLOAD 宣言子

3.2.6.1 OpenMP 4.0 の変更 (13.1.0)*

13.1.0 (インテル® Composer XE 2013 Update 2) コンパイラーでは、OpenMP* 4.0 で追加される次の宣言子、節およびプロシージャーがサポートされました。製品のドキュメントが更新されるまで、http://intel.ly/VPV24X (英語) を参照してください。

SIMD 宣言子:

- OMP SIMD
- OMP DECLARE SIMD
- OMP DO SIMD
- OMP PARALLEL DO SIMD

コプロセッサー宣言子:

- OMP TARGET DATA
- OMP TARGET
- OMP TARGET UPDATE
- OMP DECLARE TARGET
- OMP DECLARE TARGET MIRROR
- OMP DECLARE TARGET LINKABLE

節:

- MAP
- MAPFROM
- MAPTO
- SCRATCH

プロシージャー:

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.6.2KMP_PLACE_THREADS 環境変数 (13.1.0)

この環境変数を使用すると、ユーザーは明示的なアフィニティー設定やプロセス・アフィニティー・マスクを記述する代わりに、OpenMP* アプリケーションで使用するコア数およびコアごとのスレッド数を簡単に指定することができます。

構文

```
value = ( int [ "C" | "T" ] [ delim ] | delim ) [ int [ "T" ]
[ delim ] ] [ int [ "O" ] ];

delim = "," | "x";
```

効果

使用するコア数、オフセット値 (オプション)、コアごとのスレッド数を指定します。"C" はコア数、"T" はスレッド数、"O" はオフセットを示します。コア数またはスレッド数のいずれかを指定する必要があります。省略した場合、デフォルト値は利用可能なコア数 (スレッド数) です。

例

5C,3T,10 - オフセット 1 で 5 コア、コアごとに 3 スレッド使用します。
5,3,1 - 上記と同じです。
24 - 最初の 24 コア、コアごとに利用可能なすべてのスレッドを使用します。
2T - すべてのコア、コアごとに 2 スレッド使用します。
,2 - 上記と同じです。
3x2 - 3 コア、コアごとに 2 スレッド使用します。
4C120 - オフセット 12 で 4 コア、コアごとに利用可能なすべてのスレッドを使用します。

3.2.7 派生型のコンポーネントでの ATTRIBUTES ALIGN 宣言子の指定 (13.0.1)

コンパイラー 13.0.1 では、派生型の ALLOCATABLE または POINTER コンポーネントに ATTRIBUTES ALIGN 宣言子が指定されます。宣言子は派生型宣言内に配置しなければなりません。拡張型の場合、宣言子は親の型のコンポーネントを指定してはなりません。

この宣言子が指定されると、コンパイラーは明示的な ALLOCATE または (ALLOCATABLE コンポーネントに対する) Fortran 言語規則に従った暗黙の割り当てによりコンポーネントが割り当てられたときに指定されたアライメントを適用します。

派生型コンポーネントに ATTRIBUTES ALIGN 宣言子を含むモジュールはバージョン 13.0.1 よりも前のコンパイラーで使用できません。

3.2.8 その他のコンパイラーの変更

- G フォーマット編集記述子の出力が、より適切に Fortran 2008 規格に準拠するように変更されました。この変更には、選択されたフォーマットでの丸めの効果と -0 に丸められた値の表現が含まれます。
- D、E、G、EN、ES フォーマットを使用した出力で指数フィールドが暗黙の指数幅をオーバーフローすると、出力フィールドはアスタリスクで埋められます。以前のバージョンでは、指数の表示が規格に準拠していませんでした。
- コンパイラーは RANDOM_NUMBER および RANF 組込みサブルーチンへの参照をベクトル化するようになりました。

3.2.9 ファイル・バッファリング動作の変更 (13.1)

インテル® Fortran Composer XE 2013 (コンパイラー 13.0) 以前のバージョンでは、Fortran ランタイム・ライブラリーは、可変長の書式なしシーケンシャル・ファイルのレコードを読み

取るときにすべての入力をバッファリングしていました。このデフォルトのバッファリングは、任意のサイズの可変長レコードをメモリーに保持できる大きな内部バッファーを割り当てることで行われます。非常に大きなレコードの場合、メモリーが過度に使用され、最悪の場合は利用可能なメモリーを使い果たす可能性があります。しかし、レコードを読み取るときのデフォルトのバッファリング動作を変更する方法は用意されていませんでした(レコードを書き込むときにレコードのバッファリングを要求または拒否することは可能でした)。

このデフォルトのバッファリング動作は、インテル® Fortran Composer XE 2013 で変更され、これらのレコードはすべてデフォルトではバッファリング**されず**、ディスクからユーザープログラムの変数に直接読み込まれるようになりました。この変更はメモリーを確保する必要があるプログラムを支援するために行われたものですが、多くの小さなコンポーネントで構成されているレコードを読み取るときにパフォーマンスが低下する場合があります。実際、一部のユーザーから、パフォーマンスの低下が報告されました。

このため、インテル® Fortran Composer XE 2013 Update 2 (コンパイラー 13.1) では、ユーザーがこれらの可変長書式なしレコードをバッファリングするかどうかを選択できる手法が提供されました。デフォルトの動作は 13.0 と同じで、これらのレコードはデフォルトではバッファリングされません。13.1 でこの種の I/O を使用したときにパフォーマンスが低下する場合は、レコードの出力のバッファリングを有効にするかどうかを選択する場合と同じ方法で入力のバッファリングを有効にすることができます。

- ファイルの OPEN 文で BUFFERED="YES" を指定する
- 環境変数 FORT_BUFFERED に YES、TRUE、またはゼロ以外の整数値を指定する
- コンパイラーのコマンドラインで -assume buffered_io を指定する

これらのメカニズムは、これまで、可変長書式なしシーケンシャル・ファイルの書き込みを行う場合にのみ適用されていたものです。これらのメカニズムを使用すると、Fortran ランタイム・ライブラリーは、ファイルのレコードのサイズに関係なく、ファイルの入力レコードをすべてバッファリングします。

つまり、13.0 より前のデフォルトの動作に戻ることになります。

3.3 新規および変更されたコンパイラー・オプション

詳細は、コンパイラーのドキュメントを参照してください。

- -align array8byte
- -align array16byte
- -align array32byte
- -align array64byte
- -align array128byte
- -align array256byte
- -assume [no]std_intent_in
- -diag-enable sc-enums
- -diag-enable sc-{full|concise|precise}
- -diag-enable sc-single-file
- -fimf-domain-exclusion=classlist[:funclist] (下記を参照)
- -guide-profile=<file|dir>[,[file|dir],...]
- -mmic
- -no-offload
- -offload-attribute-target=<name>
- -offload-option<target>,<tool>,"option list"
- -openmp-link <library>
- -vec-report6
- -vec-report7 (13.1.0)

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

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

次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

`-fimf-domain-exclusion=c`*`lassList`*`[:`*`funcList`*`]`

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。

3.3.2 新しい -vec-report7 コンパイラー・オプション (13.1.0)

-vec-report7 オプションは、ロード、ストア、型変換、その他についての統計とメトリクスを含む、ループのベクトル化に関するより詳細な情報を出力します。この情報は、指定したループをベクトル化した場合にどの程度パフォーマンスの向上が見込めるか理解するのに役立ちます。

このオプションを指定したときに出力される情報は、http://intel.ly/XeSkW6 (英語) から入手できる新しいベクトル化解析ツールで処理することもできます。

3.4 その他の変更および注意

3.4.1 コンパイラー環境の設定

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

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、インテル® パフォーマンス・ライブラリー、インテル® C++ コンパイラー (インストールされている場合) の環境も設定されます。

3.5 既知の問題

3.5.1 Co-Array の問題

Fortran 2008 Co-Array サポートの既知の問題の一覧は、「Co-Array の既知の問題」を参照してください。

3.6 Co-Array

共有メモリー構成で Co-Array を使用するプログラムの実行に特別なプロシージャーは必要ありません。実行ファイルを実行するだけでかまいません。根本的な並列化の実装にはインテル® MPI が使用されます。コンパイラーをインストールすると、共有メモリーでの実行に必要なインテル® MPI ランタイム・ライブラリーが自動的にインストールされます。インテル® クラスター・ツールキット製品 (オプション) をインストールすると、分散メモリーでの実行に必要なインテル® MPI ランタイム・ライブラリーがインストールされます。別の MPI 実装または OpenMP* を使用する Co-Array アプリケーションはサポートしていません。

デフォルトでは、作成されるイメージの数は現在のシステムの実行ユニットの数と同じです。メインプログラムをコンパイルする ifort コマンドで `-coarray=num-images <n>` オプションを指定することで、この設定を変更することができます。また、環境変数 `FOR_COARRAY_NUM_IMAGES` でイメージ数を指定することもできます。

3.6.1 Coarray アプリケーションのデバッグ方法

Co-Array アプリケーションのデバッグ方法を以下に説明します。

1. デバッグするコードの前にストールループを追加します。
   例:

```
LOGICAL VOLATILE ::WAIT_FOR_DEBUGGER
LOGICAL, VOLATILE ::TICK
   :
DO WHILE(WAIT_FOR_DEBUGGER)
  TICK = .NOT.TICK
  END DO
!デバッグするコード
!
```

ループがコンパイラーによって削除されないように VOLATILE を使用します。問題が 1 つのイメージでのみ見つかった場合は、ループを

```
IF (THIS_IMAGE() .EQ.4) THEN
```

のように囲みます。

2. デバッグをオンにしてコンパイルおよびリンクします (-g)。
3. アプリケーションを実行するマシンに少なくとも N + 1 (N はアプリケーションのイメージ数) のターミナルウィンドウを作成します。
4. ターミナルウィンドウでアプリケーションを開始します。

```
linuxprompt> ./my_app
```

5. その他のターミナルウィンドウで、デフォルトのディレクトリーがアプリケーションの実行ファイルの場所と同じになるように設定します。1 つのウィンドウで "ps" コマンドを使用してプログラムを実行しているプロセスを調べます。

```
linuxprompt> ps –ef | grep 'whoami' | grep my_app
```

複数のプロセスが表示されます。最も古いプロセスがステップ 4 で開始したプロセスです。このプロセスは MPI ランチャーを起動して他のプロセスが終了するのを待機しています。このプロセスをデバッグしないでください。

他のプロセスは以下のようになります。

```
<ユーザー名>  25653 25650 98 15:06 ?        00:00:49 my_app
<ユーザー名>  25654 25651 97 15:06 ?        00:00:48 my_app
<ユーザー名>  25655 25649 98 15:06 ?        00:00:49 my_app
```

最初の番号はプロセスの PID です (例えば、最初の行では 25653)。

"my_app" P1、P2、P3、… を実行している N 個のプロセスの PID を呼び出します。

6. 各ウィンドウで (最初のウィンドウを除く) デバッガーを開始し、アタッチしたときにプロセスを停止するように設定します。

```
linuxprompt> idb –idb
(idb) set $stoponattach = 1
```

または

```
linuxprompt> gdb
```

7. プロセス (ウィンドウ 1 では P1、ウィンドウ 2 では P2、...) にアタッチします。

   ```
   (idb) attach <P1> my_app
   ```

   または

   ```
   (gdb) attach <P1>
   ```

8. ストールループを抜けます。

   ```
   (idb) assign WAIT_FOR_DEBUGGER = .FALSE.
   ```

   または

   ```
   (gdb) set WAIT_FOR_DEBUGGER = .false.
   ```

9. デバッグを開始します。

idb を使用している場合、idb のマルチプロセス機能を使用して、N 個のウィンドウではなく 1 つのウィンドウで実行できます。最初に、各プロセスにアタッチしてストールループを抜けます (ステップ 7 および 8)。

```
(idb) attach <P1> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P2> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P3> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
```

"process" コマンドを使用してデバッグ対象を別のプロセスに切り替えます。

```
(idb) process <Pn>
```

デバッグ対象ではないプロセスはブレークポイントとウォッチポイントがセットされた状態のまま実行されません。

3.6.2 Co-Array のパフォーマンスを向上させるコンパイラー・オプション (13.0.1)

Co-Array アプリケーションのパフォーマンスを向上させる作業は現在も行われています。その成果の一部は 13.0.1 (Composer XE 2013 Update 1) コンパイラーに実装されていますが、デフォルトでは無効になっています。実装されている最適化を有効にするには、アプリケーションをコンパイルするときに次のコンパイラー・オプションを指定します。

```
-switch coarray_opts
```

現時点では、このオプションを追加することで Co-Array 値を別のイメージからコピーする際のパフォーマンスが向上します。このオプションを使用してもパフォーマンスが大幅に向上しないアプリケーションもあります。アプリケーションをコンパイルするときに、このオプションを指定することを推奨します。エラーが発生した場合は、テクニカルサポートにお知らせください。

将来のメジャーバージョンでは、これらの最適化はデフォルトで有効になり、上記のオプションは削除される予定です。

3.6.3 Co-Array の既知の問題

このバージョンでは、以下の機能は完全には動作しません。

- 派生型 Co-Array の ALLOCATABLE または POINTER コンポーネントの別のイメージの値へのアクセス。一部は動作します。

3.7 Fortran 2003 および Fortran 2008 機能の概要

インテル® Fortran コンパイラーは、Fortran 2003 の多くの機能をサポートしています。現在サポートしていない Fortran 2003 機能についても、今後サポートしていく予定です。現在のコンパイラーでは、以下の Fortran 2003 機能がサポートされています。

- Fortran 文字セットが次の 8 ビット ASCII 文字を含むように拡張: ~ \ [ ] ` ^ { } | # @
- 最大長 63 文字までの名前
- 最大 256 行の文
- 角括弧 [ ] を (/ /) の代わりに配列の区切り文字として使用可能
- コンポーネント名とデフォルト初期化を含む構造コンストラクター
- 型と文字列長仕様を含む配列コンストラクター
- 名前付き PARAMETER 定数は複素定数の一部
- 列挙子
- 割り当て可能な派生型のコンポーネント
- 割り当て可能なスカラー変数
- 無指定文字長エンティティー
- PRIVATE コンポーネントの PUBLIC 型と PUBLIC コンポーネントの PRIVATE 型
- ALLOCATE と DEALLOCATE の ERRMSG キーワード
- ALLOCATE の SOURCE= キーワード
- 型拡張子
- CLASS 宣言
- 多相型エンティティー
- 継承と関連付け
- 遅延バインディングと抽象型
- 型バインド・プロシージャー
- TYPE CONTAINS 宣言
- ABSTRACT 属性
- DEFERRED 属性
- NON_OVERRIDABLE 属性
- 型バインド・プロシージャーの GENERIC キーワード
- FINAL サブルーチン
- ASYNCHRONOUS 属性および文
- BIND(C) 属性および文
- PROTECTED 属性および文
- VALUE 属性および文
- VOLATILE 属性および文
- ポインター・オブジェクトの INTENT 属性
  代入文の左辺と右辺の形状または長さが異なる場合に、左辺の割り当て可能な変数を再割り当て (無指定文字長でない場合、`-assume realloc_lhs` オプションが必要)
- ポインター代入の境界の仕様と境界の再マップ
- ASSOCIATE 構造
- SELECT TYPE 構造
- すべての I/O 文で、次の数値は任意の種類で指定可能: UNIT=, IOSTAT=
- NAMELIST I/O が内部ファイルで許可
- NAMELIST グループのエンティティーの制限の緩和
- 書式付き入出力で IEEE 無限大と NaN の表現方法が変更

- FLUSH 文
- WAIT 文
- OPEN の ACCESS='STREAM' キーワード
- OPEN およびデータ転送文の ASYNCHRONOUS キーワード
- INQUIRE およびデータ転送文の ID キーワード
- データ転送文の POS キーワード
- INQUIRE の PENDING キーワード
- 次の OPEN 数値は任意の種類で指定可能: RECL=
- 次の READ および WRITE 数値は任意の種類で指定可能: REC=、SIZE=
- 次の INQUIRE 数値は任意の種類で指定可能: NEXTREC=、NUMBER=、RECL=、SIZE=
- 開始する新しい I/O が自身以外の内部ファイルを修正しない内部 I/O の場合、再帰 I/O を利用可能
- IEEE 無限大および非数は Fortran 2003 で指定されるフォーマット出力で表示
- BLANK、DECIMAL、DELIM、ENCODING、IOMSG、PAD、ROUND、SIGN、SIZE I/O キーワード
- DC、DP、RD、RC、RN、RP、RU、RZ 書式編集記述子
- I/O フォーマットで、繰り返し指定子が続く場合、P 編集記述子の後のカンマはオプション
- USE 内のユーザー定義演算子名の変更
- USE の INTRINSIC および NON_INTRINSIC キーワード
- IMPORT 文
- 割り当て可能なダミー引数
- 割り当て可能な関数結果
- PROCEDURE 宣言
- 外部プロシージャーを参照する場合、汎用インターフェイス・ブロックの MODULE PROCEDURE からキーワード MODULE を省略
- プロシージャー・ポインター
- ABSTRACT INTERFACE
- PASS 属性と NOPASS 属性
- SYSTEM_CLOCK 組込み関数の COUNT_RATE 引数が任意の種類の REAL で指定可能
- STOP 文の実行で IEEE 浮動小数点例外が発生すると警告を表示
- `-assume noold_maxminloc` が指定された場合、ゼロサイズの配列の MAXLOC または MINLOC でゼロを返す
- 型問い合わせ組込み関数
- COMMAND_ARGUMENT_COUNT 組込み関数
- EXTENDS_TYPE_OF と SAME_TYPE_AS 組込み関数
- GET_COMMAND 組込み関数
- GET_COMMAND_ARGUMENT 組込み関数
- GET_ENVIRONMENT_VARIABLE 組込み関数
- IS_IOSTAT_END 組込み関数
- IS_IOSTAT_EOR 組込み関数
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC 組込み関数 (CHARACTER 引数)
- MOVE_ALLOC 組込み関数
- NEW_LINE 組込み関数
- SELECTED_CHAR_KIND 組込み関数
- 次の組込み関数においてオプションで KIND= 引数を指定可能: ACHAR、COUNT、IACHAR、ICHAR、INDEX、LBOUND、LEN、LEN、TRIM、MAXLOC、MINLOC、SCAN、SHAPE、SIZE、UBOUND、VERIFY
- ISO_C_BINDING 組込みモジュール
- IEEE_EXCEPTIONS、IEEE_ARITHMETIC、IEEE_FEATURES 組込みモジュール
- ISO_FORTRAN_ENV 組込みモジュール

このリリースではまだ実装されていないか、動作しない Fortran 2003 機能の一部を次にリストします。

- ユーザー定義の派生型 I/O
- パラメーター化された派生型
- 初期化式での変形組込み関数 (MERGE や SPREAD など) の使用

インテル® Fortran コンパイラーは、Fortran 2008 規格のいくつかの機能もサポートしています。その他の機能は将来のリリースでサポートされる予定です。現在のコンパイラーでは、以下の Fortran 2008 機能がサポートされています。

- 配列の最大次元数が 31 次元に (Fortran 2008 では 15 次元)
- Co-Array
- CODIMENSION 属性
- SYNC ALL 文
- SYNC IMAGES 文
- SYNC MEMORY 文
- CRITICAL および END CRITICAL 文
- LOCK および UNLOCK 文
- ERROR STOP 文
- ALLOCATE および DEALLOCATE で Co-Array を指定
- 組込みプロシージャー: ATOMIC_DEFINE、ATOMIC_REF、IMAGE_INDEX、LCOBOUND、NUM_IMAGES、THIS_IMAGE、UCOBOUND
- CONTIGUOUS 属性
- ALLOCATE の MOLD キーワード
- DO CONCURRENT
- OPEN の NEWUNIT キーワード
- G0 および G0.d フォーマット編集記述子
- 無制限のフォーマット項目繰り返しカウント指定子
- CONTAINS セクションは空にすることも可能
- 組込みプロシージャー: BESSEL_J0、BESSEL_J1、BESSEL_JN、BESSEL_YN、BGE、BGT、BLE, BLT、DSHIFTL、DSHIFTR、ERF、ERFC、ERFC_SCALED、GAMMA、HYPOT、IALL、IANY、IPARITY、IS_CONTIGUOUS、LEADZ、LOG_GAMMA、MASKL、MASKR、MERGE_BITS、NORM2、PARITY、POPCNT、POPPAR、SHIFTA、SHIFTL、SHIFTR、STORAGE_SIZE、TRAILZ
- 組込みモジュール ISO_FORTRAN_ENV の追加: ATOMIC_INT_KIND、ATOMIC_LOGICAL_KIND、CHARACTER_KINDS、INTEGER_KINDS、INT8、INT16、INT32、INT64、LOCK_TYPE、LOGICAL_KINDS、REAL_KINDS、REAL32、REAL64、REAL128、STAT_LOCKED、STAT_LOCKED_OTHER_IMAGE、STAT_UNLOCKED
- ALLOCATABLE または POINTER 属性を持たない OPTIONAL 仮引数は、対応する実引数に ALLOCATABLE 属性があるのに割り当てられない場合、POINTER 属性があるのに関連付けが解除されている場合、または NULL 組込み関数への参照の場合、無視されます。
- 仮引数がプロシージャー・ポインターの場合、そのポインターの有効な参照先か、または組込み関数 NULL への参照である実引数に関連付けられます。実引数がポインターではない場合、仮引数に INTENT (IN) 属性が含まれていなければなりません。

4 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。

4.1 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

4.2 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) 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
```

4.3 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>`

4.4 その他のドキュメント

*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。

[?] 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。

4.5 デバッガー機能

4.6 既知の問題と変更点

4.6.1 Co-Array の要素を表示できません。

IDB デバッガーは Co-Array 要素を表示できません。回避策は、「Co-Array アプリケーションのデバッグ方法」を参照してください。

4.6.2 [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。

4.6.3 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。

4.6.4 $cdir ディレクトリー、$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。

4.6.5 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。

4.6.6 $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
```

4.6.7 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。

4.6.8 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。

4.6.9 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。

4.6.10 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。

4.6.11 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
```

4.6.12 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、http://intel.ly/ng91l0 (英語) から Web ベースのドキュメントを利用できます。

4.6.13 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。

4.6.14 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。

4.6.15 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 -debx は、デバッガー GUI ではサポートされていません。

4.6.16 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 インテル® Xeon Phi™ コプロセッサー

このセクションでは、インテル® Composer XE 2013 Linux* 版を使用したインテル® Xeon Phi™ コプロセッサー向けの開発の変更点、新機能、および最新情報をまとめています。

5.1 概要

インテル® Fortran Composer XE 2013 は、インテル® MIC アーキテクチャーのコプロセッサー (インテル® Xeon Phi™ 製品ファミリー) に作業をオフロードするアプリケーションの開発がサポートされました。インテル® Xeon Phi™ 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。インテル® Xeon Phi™ コプロセッサーでネイティブに実行するアプリケーションの開発もサポートされました。

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。

5.2 ドキュメント

インテル® Fortran 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。

5.3 デバッガー

インテル® Xeon Phi™ コプロセッサ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、インテル® デバッガーのコマンドライン・バージョンが含まれています。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。

5.4 既知の問題と変更点

このセクションでは、製品ドキュメントの訂正または追加をまとめています。

5.4.1 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、offload 宣言子が見つかったときに、ターゲットのコンパイル診断にのみ追加されます。

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。

5.4.2 直接 (ネイティブ) モードにおける libiomp5.so 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) には、/lib 以下のインテル® コンパイラーのライブラリーは含まれません。

このため、直接モード (例えば、コプロセッサー上) でアプリケーションを実行する場合は、アプリケーションで使用する共有オブジェクト・ライブラリーのコピーを (scp 経由で) 最初にアップロードする必要があります。例えば、アプリケーションを実行する前に OpenMP* ライブラリー (<install_dir>/compiler/lib/mic/libiomp5.so) をコプロセッサー (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、…) にコピーします。

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。

一部のアプリケーションでは、別のライブラリーもアップロードする必要があります。

5.4.3 メモリー割り当てのパフォーマンスのチューニング

次の説明は、製品ドキュメントの説明を修正したものです。

コプロセッサーのユーザー割り当てデータでは、malloc や ALLOCATE 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、コプロセッサーの 2MB ページに限られた TLB (トランスレーション・ルックアサイド・バッファー) エントリーしかないとパフォーマンスの低下を引き起こします。

malloc および ALLOCATE のデフォルトのページサイズは 4KB です。

5.4.4 -opt-streaming-stores never の使用についての注意

インテル® Xeon Phi™ コプロセッサーのステッピングが A の場合、アプリケーションをコンパイルするときに -opt-streaming-stores never オプションを指定する必要があります。このオプションを指定しない場合、ステッピング A でサポートされていない命令が生成されます。ステッピング B 以降は、新しい命令をサポートしています。

5.4.5 インテル® Composer XE 2013 の初期リリースでビルドしたバイナリーと 2013 Update 1 以降のオフロード・ライブラリーを実行するとランタイムエラーまたはクラッシュが発生する

インテル® Composer XE 2013 Update 1 でのオフロード・ライブラリーのバイナリー変更に伴い、インテル® Composer XE 2013 の初期リリースでビルドしたバイナリーと Update 1 以

降のライブラリーを使用するとランタイムエラーまたはクラッシュが発生します。この場合に発生するエラーの例を次に示します。

エラー 1:

```
***Warning: offload to device #0 : failed
```

エラー 2:

```
Segmentation fault (core dumped)
```

エラー 3:

```
terminate called after throwing an instance of 'COIRESULT'

terminate called recursively
```

エラー 4:

```
CARD--ERROR:1 myoiPageFaultHandler: (nil) Out of Range!

CARD--ERROR:1 _myoiPageFaultHandler: (nil) switch to default signal
handle

CARD--ERROR:1 Segment Fault!

HOST--ERROR:myoiScifGetRecvId: Call recv() Header Failed ! errno =
104

^CHOST--ERROR:myoiScifSend: Call send() Failed! errno = 104

HOST--ERROR:myoiSend: Fail to send message!

HOST--ERROR:myoiBcastToOthers: Fail to send message to 1!

HOST--ERROR:myoiBcast: Fail to send message to others!
```

これらの問題を解決するには、アプリケーションをインテル® Composer XE 2013 Update 1 以降でリビルドしてください。

5.4.6 連続していない部分配列がオフロード領域に渡されない

インテル® Fortran Composer XE 2013 Update 2 (コンパイラー 13.1) では、連続していない部分配列はオフロード領域に渡されないように制限されています。次に例を示します。

```
integer, pointer :: p(:)
integer, target :: t(20)

p => t(1:20:2)

…

!dir$ omp offload target(mic)
!$omp parallel do shared(p)
…
```

この例では、p が連続していない部分配列でオフロード領域に渡されます。この場合、次のようなランタイムエラーが発生します。

オフロードエラー: オフロードデータ転送では変数ごとに 1 つの連続するメモリー範囲のみサポートされます。

オフロード領域に渡される部分配列がすべて連続していることを確認してください。

5.4.7 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。

5.4.7.1 MIC_USE_2MB_BUFFERS

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

例:

```
// コプロセッサーでサイズが 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
```

5.4.7.2 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™ コプロセッサー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
```

5.4.7.3 MIC_ENV_PREFIX

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

MIC_ENV_PREFIX には、コプロセッサーを示す環境変数として使用する値を設定します。

例えば、

```
setenv MIC_ENV_PREFIX MYCARDS
```

は特定のコプロセッサー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、各コプロセッサーに <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、コプロセッサー <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、各コプロセッサーに <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、コプロセッサー <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ーで ABCD=abcd を設定

setenv PHI_2_EFGH efgh    // コプロセッサー 2 で EFGH=efgh を設定

setenv PHI_VAR X=x|Y=y    // すべてのコプロセッサーで X=x と Y=y を設定

setenv PHI_4_VAR P=p|Q=q  // コプロセッサー 4 で P=p と Q=q を設定
```

5.4.8 OFFLOAD_DEVICES

OFFLOAD_DEVICES は、変数の値として指定されたコプロセッサー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”
```

5.4.9 デバッグとインテル® デバッガー

5.4.9.1 インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーでの IDB の使用

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>
  ```

5.4.9.2オフロード・デバッグ・セッションの安全な終了方法

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

- アプリケーションが終了コードに到達する前にデバッグセッションを手動で停止します。
- 最初に、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。

5.4.9.3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。

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。

6 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。

6.1 インテル® MKL 11.0 Update 3 の新機能

- 報告された問題の修正 – 修正リスト
- BLAS:
  - Intel® Optimized MP LINPACK Benchmark for Clusters パッケージを HPL 2.1 に更新
- 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 2 (インテル® AVX2) における DCOOMM のパフォーマンスが向上
- LAPACK:
  - ?LASET、?LACPY、?LANGE、?LANSY を並列化
  - 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ける (C/Z)POTRF のパフォーマンスが向上
- サービス関数:
  - インテル® MIC アーキテクチャーの自動オフロードで使用されるスレッド数の制御を追加
- FFT:
  - インテル® AVX2 における複素数-複素数の 2 の累乗 FFT のパフォーマンスが向上
- VSL:
  - インテル® AVX2 およびインテル® MIC アーキテクチャーにおける SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
- クラスター FFT:
  - ハイブリッド・モードのクラスター FFT のパフォーマンスが向上

- データ・フィッティング:
  - 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 およびインテル® Xeon® プロセッサー E5-2690 における線形および 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
- インテル® MIC アーキテクチャーにおける df?interpolate および df?searchcells1d 関数のパフォーマンスが向上

6.2 インテル® MKL 11.0 Update 2 の新機能

- インテル® MKL Extended Eigensolver (拡張固有値ソルバー) を追加:
  - インテル® MKL Extended Eigensolver は、密行列、LAPACK 帯行列、疎行列 (CSR) 形式の標準定値固有値問題および一般化定値固有値問題を解くハイパフォーマンスなパッケージです。Feast という革新的な高速かつ安定した数値アルゴリズムをベースにしています (「権利の帰属」セクションを参照)。
- BLAS:
  - [C/Z]HERK をインテル® 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ネイティブ実行用に最適化
  - BLAS レベル 3 サブルーチン ?SYMM (全精度) をインテル® MIC アーキテクチャーの自動オフロード (AO) 用に最適化
- スパース BLAS:
  - 0 ベースの DCSRMM のパフォーマンスが大幅に向上
- LAPACK:
  - ?(OR/UN)G(LQ/QL/QR/RQ) 関数の並列バージョンのパフォーマンスが大幅に向上
  - LU (?GETRF)、コレスキー (?POTRF)、および QR (?GEQRF) 分解関数をインテル® MIC アーキテクチャーの自動オフロード用に最適化
  - インテル® MIC アーキテクチャー (60 コア) における LU および SMP Linpack のパフォーマンスが向上
- ScaLAPACK:
  - バージョン 2.0.2 にアップデート。新しい関数を追加:
    - P?HSEQR: 非対称固有値問題
    - P?SYEVR/P?HEEVR: MRRR (Multiple Relatively Robust Representations) アルゴリズム
- FFT:
  - インテル® MIC アーキテクチャーにおける複素数-複素数の 2 の累乗 1D および 2D FFT 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実数-複素数の 2 の累乗および奇数サイズ 1D FFT のパフォーマンスが向上
  - インテル® MIC アーキテクチャー用のコンパイラーによるオフロード支援使用モデルに MKL FFT を使用した例を追加
  - インテル® MIC アーキテクチャーにおける DFTI 記述子のコミット時間が短縮
  - インテル® MIC アーキテクチャーで FFTW インターフェイス・ラッパー・ライブラリーをサポート
- クラスター FFT:
  - FFTW2 ラッパーを含む多次元クラスター FFT での転置順を実装
- VSL:
  - VSL 対数正規乱数ジェネレーター (RNG) で累積分布逆関数 (ICDF) メソッドをサポート
  - データ・フィッティングおよび VSL 要約統計量関数の宣言に const 指定子を追加
  - インテル® MIC アーキテクチャーにおける Wichmann-Hill BRNG のパフォーマンスが向上

- データ・フィッティング:
  - インテル® Xeon® プロセッサー X5570 およびインテル® Xeon® プロセッサー E5-2600 におけるスカラーケース (補間位置の数が 1) とベクトルケースのデフォルト/準一様分割、ソートされた補間位置用の df[d/s]Interpolate1D、df[d/s]InterpolateEx1D、df[d/s]SearchCells1D、df[d/s]SearchCellsEx1D ルーチンのパフォーマンスが向上
  - データ・フィッティング・ルーチンのパラメーターの正当性確認を無効にすることにより、補間位置の数が少ない (12 個未満) 場合のパフォーマンスを向上させる DF_DISABLE_CHECK_FLAG パラメーターを dfiEditVal エディターでサポート
- 転置:
  - アウトオブプレース行列転置 (mkl_?omatcopy2) を並列化。パフォーマンスが大幅に向上
- サービス関数:
  - インテル® MKL メモリー・アロケーターで使用されるピークメモリーの量に関する情報を提供する mkl_peak_mem_usage 関数を追加
  - インテル® MKL メモリー・アロケーターの機能を標準 C ライブラリーのメモリー割り当て API に拡張する mkl_calloc 関数と mkl_realloc 関数を追加
- SMP LINPACK を拡張 (残差チェック):
  - 失敗が検出されるとエラーコード 1 を返し、生成された残差が精度チェックをパスするかどうか結論を出力します。条件付き数値再現性モードがオフの場合、同じ行列で実行ごとに残差が多少異なることがあるので、注意してください。残差チェックは結果が信頼できることを保証します。

6.3 インテル® MKL 11.0 Update 1 の新機能

- 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
- BLAS
  - [S/D/C/Z]SYMM 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
  - 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
  - 複素共役転置とエルミートの CSRMV 機能を最適化
- 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® プロセッサー 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ス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 http://intel.ly/OqbZEL (英語) を参照)

6.4 インテル® MKL 11.0 の新機能

- インテル® MKL がインテル® 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・ベースのインテル® Xeon Phi™ コプロセッサー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™ コプロセッサーでのインテル® MKL 使用モデルは、自動オフロード、コンパイラー支援オフロード、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ー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™ コプロセッサー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インテル® MKL 関数は、インテル® Xeon Phi™ コプロセッサーで実行されます。
- 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インテル® MKL の新しい機能です。詳細は、『インテル® 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 (http://intel.ly/P4yRXR) (英語) も参照してください。
- インテル® MKL は、新しい FMA3 命令を含む、新しいインテル®アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) を使用した最適化もサポートします。詳細は、インテル® AVX2 のサポートに関するナレッジベースの記事 (http://intel.ly/PVmq3h) (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 (?TRMM、?TRSM、?GEMM 関数は自動オフロード用に最適化)
    - 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) 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
- スパース BLAS
    - ?CSRMV、?CSRMM、?CSRSYMV (ユニット対角の場合) 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
- LAPACK
    - [S/D]GETRF、[S/D]POTRF、[S/D]GEQRF、[S/D]GELQF、[S/D]GEQLF、[S/D]GERQF 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
    - 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
    - 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
    - MT19937、MT2203、MRG32k3a BRNG、離散一様分布および幾何分布 RNG 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® アドバンスト・ベクトル・エクステンション (インテル® 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・ディメンジ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 (インテル® 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: インテル® 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/インテル® 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
    - インテル® MIC アーキテクチャーのサポートを追加

6.5 推奨されていない (古い) 機能と削除された機能

詳細は、インテル® MKL の機能に関する記事 (http://intel.ly/LkZKGL (英語)) を参照してください。

- インテル® SSE2 拡張命令セットをサポートしないプロセッサー (インテル® Pentium® III 以前) でのインテル® MKL のサポートを終了
- 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
- 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

6.6 既知の問題

本リリースにおける既知の制限事項の詳細なリストは、http://intel.ly/ptEfAP (英語) を参照してください。

6.7 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 (www.intel.com/software/products/mkl (英語)) へのリンク/URL の提供を正確に行うことが最低限必要です。

インテル® MKL の一部の基となった BLAS の原版は http://www.netlib.org/blas/index.html (英語) から、LAPACK の原版は http://www.netlib.org/lapack/index.html (英語) から入手できます。LAPACK の開発は、E. Anderson、Z. Bai、C. Bischof、S. Blackford、J. Demmel、J. Dongarra、J.

Du Croz、A. Greenbaum、S. Hammarling、A. McKenney、D. Sorensen らによって行われました。LAPACK 用 FORTRAN 90/95 インターフェイスは、http://www.netlib.org/lapack95/index.html (英語) にある LAPACK95 パッケージと類似しています。すべてのインターフェイスは、純粋なプロシージャー用に提供されています。

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

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

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

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

7 著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

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

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

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

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

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

Intel、インテル、Intel ロゴ、Intel Core、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。