インテル® Fortran 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: 321415-005JA
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® Fortran 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 製品アップデート

Update 3 - 2014 年 4 月

- インテル® Fortran コンパイラーが 14.0.3 にアップデート
- インテル® マス・カーネル・ライブラリーが 11.1 Update 3 にアップデート

Update 2 - 2014 年 2 月

- インテル® Fortran コンパイラーが 14.0.2 にアップデート
  - -switch fe_debug_use_inherit 内部コマンドライン・オプションの追加
- インテル® マス・カーネル・ライブラリーが 11.1 Update2 にアップデート

Update 1 - 2013 年 10 月

- インテル® Fortran コンパイラーが 14.0.1 にアップデート
  - -assume std_value の追加
  - -[a]xMIC-AVX512 コンパイラー・オプションの追加
  - -f[no-]mpc_privatize コンパイラー・オプションの追加
  - -opt-gather-scatter-unroll=n コンパイラー・オプションの追加
- インテル® マス・カーネル・ライブラリーが 11.1 Update 1 にアップデート
- 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

1.1.2 インテル® Fortran Composer XE 2013 からの変更点

- インテル® Fortran コンパイラーがバージョン 14.0 にアップデート
  - インテル® Xeon Phi™コプロセッサーで実行する Co-Array アプリケーションの開発をサポート
- インテル® マス・カーネル・ライブラリーがバージョン 11.1 にアップデート
- 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
- GUI インストーラー
- オプションで必要なコンポーネントのみダウンロードするオンライン・インストーラー
- Ubuntu* 13.04 LTS、Fedora* 18、19、および、Debian* 7.0 のサポートを追加
- 次の 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
- 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 SP1 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 14.0。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® マス・カーネル・ライブラリー11.1
- 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、「Intel® Architecture Platform Terminology」 (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、7
    - Fedora* 18、19
    - Red Hat* Enterprise Linux* 5 (サポート終了予定)、6
    - SUSE* Linux* Enterprise Server 10 (サポート終了予定)、11 SP2
    - Ubuntu* 12.04 LTS、13.04
    - 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)(本リストは、インテル社により動作確認が行われたコンポーネント・バージョンのリストです。その他のバージ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。)
    - gcc 4.1、4.3-4.4、4.6-4.8
    - binutils 2.17、2.20-2.23
- `-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。

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、「Changing Language Setting to see English on a Japanese OS environment or Vice Versa on Linux*」 (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。

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

2.1 GUI インストーラー

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環境では (ssh ターミナルから実行する場合など)、デフォルトでコマンドライン・インストーラーになります。

2.2 オンライン・インストーラー

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 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。

2.4 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® Visual Fortran Composer XE 2013 SP1 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。

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 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、「Intel® Software Improvement Program」 (英語) を参照してください。

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、「Intel® Compilers for Linux* Silent Installation Guide」 (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、Licensing: Setting Up the Client for a Floating License (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。

2.8 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 (英語) で評価版のシリアル番号を入手してください。

2.9 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_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 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_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 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_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
- 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 SP1 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。

2.10 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>/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 14 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、最新のバージョンで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

- ユーザー定義の派生型 I/O

3.2.2 OpenMP* 機能

OpenMP* 4.0 の次の宣言子、節、およびプロシージャーがコンパイラーでサポートされます。これらの機能の一部は、暫定仕様に基づきインテル® Fortran Composer XE 2013 Update 2 でサポートされました。また、以前サポートされていたいくつかの構文 (DECLARE TARGET MIRROR、DECLARE TARGET LINKABLE、MAPTO、MAPFROM、SCRATCH) はサポートされなくなりました。さらに、一部の構文は以前の仕様から変更されています。

詳細は、コンパイラー・ドキュメントまたは上記の OpenMP* 仕様へのリンク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

その他の宣言子:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP

節:

- MAP

プロシージャー:

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.2.1 KMP_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。

3.2.2.2 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。

3.2.3 Co-Array とインテル® Xeon Phi™ コプロセッサー

インテル® Xeon Phi™ コプロセッサー上でネイティブ実行する、またはインテル® Xeon Phi™ コプロセッサーとインテル® 64 アーキテクチャー・ベースのホストシステムの組み合わせで実行する、Co-Array を使用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™ コプロセッサーでの Co-Array の使用」を参照してください。

3.2.4 新しい宣言子と追加された宣言子

インテル® Composer XE 2013 SP1 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。

- [NO]FMA

3.2.5 その他の機能

これらの機能に関する詳細は、コンパイラー・ドキュメントを参照してください。

- ESTABLISHQQ ライブラリー・ルーチンは、Fortran ランタイム・ライブラリーがランタイムエラーを出力する直前にユーザールーチンを呼び出すように指定します。このルーチンは、モジュール IFPORT で宣言されています。
- `-[no-]wrap-margin` コマンドライン・オプションと FORT_FMT_NO_WRAP_MARGIN 環境変数は、リスト指定出力で前のレコードが 80 文字を超える場合、新しいレコードを開始するかどうかを制御します。
- 新しい事前定義済みシンボル `__INTEL_COMPILER_UPDATE`、`__INTEL_OFFLOAD`、`__MIC__`
- 新しい環境変数 FOR_FORCE_STACK_TRACE。1 に設定した場合、実行時に診断メッセージが出力されると、コンパイラーはトレースバックを提供します。FOR_FORCE_STACK_TRACE は、FOR_DISABLE_STACK_TRACE よりも優先されます。

3.2.6 ファイル・バッファリング動作の変更 (13.1)

インテル® Visual 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® Visual 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® Visual 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

これらの手法は、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。これらの手法を使用すると、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。

- -assume std_value (14.0.1)
- -[a]xMIC-AVX512 (14.0.1)
- -f[no-]mpc_privatize(14.0.1)
- -fimf-domain-exclusion
- -fma
- -fmerge-constants
- -foptimize-sibling-calls
- -mtune=<arch>
- -openmp-offload
- -openmp-simd
- -opt-assume-safe-padding
- -offload=<arg>
- -opt-gather-scatter-unroll=n (14.0.1)
- -opt-prefetch-distance
- -opt-streaming-cache-evict
- -opt-threads-per-core
- -static-libstdc++
- -switch fe_debug_use_inherit (14.0.2)
- -vecabi
- -wrap-margin
- -xATOM_SSE4.2

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 Fortran 2003 の VALUE 属性に影響する新しいオプション

インテル® Fortran コンパイラーの Fortran 2003 の VALUE 属性の実装は、BIND(C) 言語バインド仕様が含まれないプロシージャーで使用された場合、標準の仕様と一致しません。コンパイラーのデフォルトの動作は、Fortran 2003 の VALUE 属性を、引数を「値で」受け渡す DEC$ ATTRIBUTES VALUE 宣言子と同じように扱います。標準の動作では、代わりに引数の再定義可能なコピーを渡します。また、この従来の動作では、VALUE 属性を含む OPTIONAL 属性を使用できません。プロシージャーに BIND(C) 言語バインド仕様が含まれる場合、標準の実装と同じになり、VALUE 属性を含む引数は値で受け渡しされることに注意してください。

コンパイラーのバージョン 14 では、標準に準拠した実装を利用できますが、以前の実装を想定している既存のアプリケーションで問題が発生する可能性があるため、デフォルトでは有効になっていません。標準の動作にするには、`/assume:std_value` (Windows*) または `-assume std_value` (Linux* および OS X*) コンパイラー・オプションを追加します。このオプションは、ドキュメント化されていません。Windows* で Visual Studio* を使用している場合は、[Command Line (コマンドライン)] - [Additional Options (追加のオプション)] でこのオプションを追加できます。`/standard-semantics` (Windows*) または `-standard-semantics` (Linux* および OS X*) が有効な場合は、`std_value` を意味します。

インテル® Fortran コンパイラーの将来のメジャーリリースでは、VALUE のデフォルトの動作が標準と一致するように変更される予定です。

3.3.2 新しい -[a]xMIC-AVX512 コンパイラー・オプション (14.0.1)
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) 命令対応のインテル® プロセッサー向けに最適化します。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。

3.3.3 新しい -f[no-]mpc_privatize コンパイラー・オプション (14.0.1)
このオプションは、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します。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 `-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。

3.3.4 新しい -opt-gather-scatter-unroll=n コンパイラー・オプション (14.0.1)
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・シーケンスを指定し、集約 (Gather)/分散 (Scatter) 処理のパフォーマンスを向上できる可能性があります。このオプションは、インテル® MIC アーキテクチャーにのみ適用されます。

3.3.5 新しい -switch fe_debug_use_inherit 内部コマンドライン・オプション (14.0.2)
現在、gdb デバッガーで拡張派生型の親フィールドを調べるには、親の名前をリストする必要があります。内部コマンドライン・オプション `-switch fe_debug_use_inherit` をデバッグ・コマンドラインに追加すると、短縮構文を使用して親フィールドを調べることができます。

次に例を示します。

```
TYPE BASE
   integer Base_Counter
END TYPE BASE

TYPE, EXTENDS (BASE) ::Type2
END TYPE TYPE2

TYPE(Type2) ::Foo
```

Fortran では `Foo%Base_Counter` または `Foo%base%base_counter` のいずれかを参照することができます。`fe_debug_use_inherit` オプションを設定しないで、`gdb` から前者の形式を使用することはできません。ただし、`fe_debug_use_inherit` オプションを設定すると、`gdb` 内から後者の形式を使用できないことに注意してください。

この内部コマンドライン・オプションは、コンパイラーのバージョン 15.0 ではデフォルトで有効になるためサポートされません。

3.4 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をここに追加します
   !
   ```
   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 ?        0:00:48 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、最初の行の 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-ia

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 インテル® Xeon Phi™ コプロセッサーでの Co-Array の使用

インテル® Fortran Composer XE 2013 SP1 で、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™ コプロセッサーにおける Fortran 2008 の Co-Array 機能がサポートされました。次の実行モデルの中から選択できます。

- オフロード領域を含む Co-Array アプリケーション
- コプロセッサーとインテル® Xeon® プロセッサー (ヘテロジニアス環境) で実行する Co-Array アプリケーション
- コプロセッサーでネイティブ実行する Co-Array アプリケーション

すべてのモードにおいて、インテル® MIC アーキテクチャーのアプリケーションと同様に、アプリケーションで参照される全ライブラリー共有オブジェクトをコプロセッサーにコピーする必要があります。これには、インテル® MPI ライブラリー、`libicaf.so` などのインテル® Fortran ライブラリーも含まれます。

次に例を示します。

```
sudo scp
/opt/intel/composer_xe_2013_sp1.NN/compiler/lib/mic/libicaf.so
mic0:/lib64/libicaf.so

sudo scp
/opt/intel/composer_xe_2013_sp1.NN/compiler/lib/mic/libintlc.so
mic0:/lib64/libintlc.so

sudo scp
/opt/intel/composer_xe_2013_sp1.NN/mpirt/lib/mic/libmpi_mt.so
mic0:/lib64/libmpi_mt.so

sudo scp
/opt/intel/composer_xe_2013_sp1.NN/mpirt/bin/mic/mpiexec.hydra
mic0:/bin/mpiexec.hydra

sudo scp /opt/intel/composer_xe_2013_sp1.NN/mpirt/bin/mic/pmi_proxy
mic0:/bin/pmi_proxy
```

これは、コプロセッサーを再起動するたびに行う必要があります。

3.6.2.1 オフロード領域での Co-Array の使用

Co-Array アプリケーションにおけるオフロード領域の使用には次の制限があります。

- オフロード領域内では、常に Co-Array のローカルコピーにアクセスしなければなりません。共通インデックスは許可されません。

- オフロード領域で SYNC ALL、SYNC MEMORY、SYNC IMAGES、または LOCK/UNLOCK の使用は許可されません。
- オフロード領域内で Co-Array の割り当て/割り当て解除は許可されません。

オフロード領域の使用に関する一般的な詳細は、ドキュメントを参照してください。

3.6.2.2 ヘテロジニアス Co-Array アプリケーション

インテル® 64 ホストシステムとインテル® Xeon Phi™ コプロセッサーでそれぞれ一部のイメージを実行する Co-Array アプリケーションを実行することができます。これは「ヘテロジニアス」と呼ばれます。

最初に、-coarray=coprocessor オプションと Co-Array 設定ファイルを指定してアプリケーションをビルドする必要があります。次に例を示します。

```
ifort -coarray=coprocessor \
-coarray-config-file=MixedPlatform.conf   \
mycoarrayprog.f90 -o mycoarrayprog \
```

この例では、設定ファイル名を MixedPlatform.conf としていますが、任意の名前を付けることができます。上記のコマンドを実行すると、2 つの実行ファイル mycoarrayprog と mycoarrayprogMIC が作成されます。

インテル® MIC アーキテクチャー用のネイティブ実行ファイル mycoarrayprogMIC をコプロセッサーのファイル・システムにコピーする必要があります。

MPI 設定ファイルである MixedPlatform.conf は、このヘテロジニアス構成の実行に必要です。設定ファイルの例を次に示します。

```
-n 4 -genv FOR_ICAF_STATUS=true -host myhostname mycoarrayprog : \
-n 4 -host mic0 /home/mydir/mycoarrayprogMIC
```

FOR_ICAF_STATUS=true は必須です。これは、設定ファイルがある場合は常に true です。myhostname は、インテル® 64 アーキテクチャー・ベースのホストシステムの名前です。この例では、/home/mydir がコプロセッサー上のインテル® MIC アーキテクチャー用の実行ファイルのパスです。必要に応じて、このパスは変更してください。

この設定ファイルは、ホストとカードでそれぞれ 4 つずつイメージを実行します。必要に応じて、-n の値を変更できます。

ホストシステムで実行ファイルを実行する前に、環境変数 I_MPI_MIC を ENABLE に設定します。その後、実行ファイルを実行することにより、ホストとコプロセッサーの両方で実行が開始されます。

3.6.2.3 コプロセッサー用のネイティブ Co-Array アプリケーション

コプロセッサーでネイティブ実行する Co-Array アプリケーションをビルドするには、-coarray オプションと –mmic オプションを指定してビルドします。設定ファイルは不要です。次に例を示します。

```
ifort -coarray -mmic mycoarrayprog.f90 -o \
mycoarrayprog -L/opt/intel/composer_xe_2013_sp1/mpirt/lib/mic
```

上記のコマンドを実行すると、インテル® MIC アーキテクチャー用のネイティブ実行ファイル「mycoarrayprog」が作成されます。

次のように、micnativeloadex ユーティリティーを使用して、アプリケーションを簡単に実行できます。

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex \
mycoarrayprog
```

アプリケーションで参照されるすべての共有イメージが含められ、コプロセッサーでアプリケーションが実行されます。

以下のコマンドを実行すると、このツールのヘルプを表示できます。

```
/opt/intel/mic/coi/tools/micnativeloadex/release/micnativeloadex -h
```

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

- ユーザー定義の派生型 I/O
- 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。

- 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 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。

4.1 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
- Fortran サポートの向上

4.2 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。

4.3 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。

GNU* GDB の使用方法は、「ドキュメント」を参照してください。

4.4 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
```

4.5 既知の問題と変更点

4.5.1 オフロード・デバッグ・セッションの安全な終了方法

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

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

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

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

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

4.5.3 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 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。

5.1 インテル® デバッガーのサポート終了予定

将来のメジャーリリースでは、インテル® デバッガーが削除される予定です。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。

5.2 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。

インテル® デバッガーの使用方法は、「ドキュメント」を参照してください。

5.3 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/

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

5.4 デバッガー機能

5.4.1 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。

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

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

5.5 既知の問題と変更点

5.5.1 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。

5.5.2 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>
```

5.5.3 インテル® デバッガーがインテル® MPSS 3.1 で動作しない

インテル® MPSS 3.1 は、インテル® デバッガーと互換性がありません。インテル® MPSS 3.1 のデバッグは、提供される GNU* GDB バージョンでのみ可能です。

5.5.4 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。

5.5.4.1 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。

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

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

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

5.5.6 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。

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

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

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

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

5.5.9 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。

5.5.10 位置独立実行ファイル (PIE) のデバッグは未サポート

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

5.5.11 コマンドライン・パラメーター -parallel は未サポート

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

5.5.12 [Signals (シグナル)] ダイアログが動作しない

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

5.5.13 GUI のサイズ調整

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

5.5.14 $cdir ディレクトリー、$cwd ディレクトリー

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

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

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

5.5.15 info stack の使用

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

```
info stack [num]
```

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

5.5.16 $stepg0 のデフォルト値の変更

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

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

5.5.17 一部の Linux* システムでの SIGTRAP エラー

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

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

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

5.5.18 MPI プロセスのデバッグには idb GUI は使用不可

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

5.5.19 GUI でのスレッド同期ポイントの作成

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

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

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

5.5.21 IA-32 アーキテクチャー向けのスタック・アライメント

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

5.5.22 GNOME 環境の問題

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

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

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

5.5.23 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/o5DMp9 (英語) から Web ベースのドキュメントを利用できます。

6 インテル® Xeon Phi™ 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™ 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。

6.1 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。

6.2 ドキュメント

最新のドキュメントとアップデートは、「Intel® Composer XE 2013 Documentation Updates for Intel® MIC Architecture」 (英語) を参照してください。

6.3 デバッガー

6.3.1 GNU* GDB

「GNU* GDB デバッガー」セクションを参照してください。

6.3.2 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。

6.4 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。

6.4.1 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。

6.4.2 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。

6.4.3 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。

6.4.4 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。

6.4.5 -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。

6.4.6 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。

6.4.6.1 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。

次に例を示します。

```
// 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
```

6.4.6.2 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。

次に例を示します。

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

6.4.6.3 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。
次に例を示します。

```
setenv MIC_ENV_PREFIXMYCARDS
```
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> と <variable2>=<value2> が渡されます。

次に例を示します。

```
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
```

6.4.7 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。

次に例を示します。

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

7 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。

7.1 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

7.2 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

7.3 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)
- サンプルで Visual Studio* 2013 をサポート
- 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対応並列領域から 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

7.4 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 (開発コード名 IvyTown) 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

7.5 注意事項

- 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® 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 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

7.6 既知の問題

既知の制限事項の詳細なリストは、インテル® デベロッパー・ゾーンにある「Intel® MKL Article List」 (英語) を参照してください。

7.7 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/) をベースにしています。

8　著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、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。

Intel、インテル、Intel ロゴ、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。

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。

* 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。

© 2014 Intel Corporation. 無断での引用、転載を禁じます。