インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* インストール・ガイドおよびリリースノート

2015 年 4 月 1 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。リリースノートの最新アップデートについては、インテル® ソフトウェア開発製品レジストレーション・センターにリストされているリリースノートを参照してください。

インテル® Parallel Studio XE は統合的なソフトウェア開発ツールであり、各コンポーネントは異なるライセンスの下で提供されます。詳細は、パッケージに含まれるライセンスと本リリースノートの「著作権と商標について」を参照してください。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。各コンポーネントの新機能の詳細は、各コンポーネントのリリースノートを参照してください。

1.1.1 Update 3

- インテル® Fortran コンパイラーが 15.0.3 にアップデート
- インテル® マス・カーネル・ライブラリー (インテル® MKL) 11.2 Update 3
- 報告された問題の修正

1.1.2 Update 2

- 割り当て多相コンポーネントの代入に関する新しいライブラリー・サポート
- Fedora* 21 をサポート
- インテル® Fortran コンパイラーが 15.0.2 にアップデート
- インテル® MKL 11.2 Update 2
- GNU* プロジェクト・デバッガー (GDB) 7.8 (IA-32/インテル® 64 アーキテクチャー用)
- 報告された問題の修正

1.1.3 Update 1

- IA-32 およびインテル® 64 アーキテクチャー向けインテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令セットをサポート (インテル® コンパイラー 15.0.1)
- 日本語版を含む最初のアップデート
- SuSE Linux Enterprise Server* 12 をサポート
- SIMD ループ宣言子で MIN/MAX リダクションをサポート
- インテル® Fortran コンパイラーが 15.0.1 にアップデート
- インテル® MKL 11.2 Update 1
- 報告された問題の修正

1.1.4 インテル® Fortran Composer XE 2013 SP1 以降 (インテル® Parallel Studio XE 2015 Composer Edition での変更)

- インテル® Fortran コンパイラーがバージョン 15.0 にアップデート
  - 新しい最適化レポートのインターフェイス、構造、オプション (既存の -opt-report、-vec-report、-openmp-report、および -par-report オプションを使用しているユーザーは、インテル® コンパイラーのユーザー・リファレンス・ガイドで詳細を確認することを強く推奨します。)
- インテル® MKL がバージョン 11.2 にアップデート
- Python* なしで GNU* プロジェクト・デバッガーが利用可能

- Fortran のサポートが強化された GNU* GDB 7.7
- インテル® デバッガー (IDB) を削除
- スタティック解析は非推奨 (廃止予定)
- -o で始まるコンパイラー・オプションは非推奨 (廃止予定)
- binutils 2.24 をサポート
- Red Hat* Enterprise Linux* 7 をサポート
- Ubuntu* 13.10、14.04 LTS、Fedora* 20 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* 18、19
  - Ubuntu* 13.04
  - SUSE Linux Enterprise Server* 10
- OpenMP* 4.0 の機能を追加サポート
- オンライン・インストーラーでのカスタム・インストール設定
- PGO によるスレッドセーフなプロファイル生成が可能

- PGO .dyn ファイル名にカスタム・プリフィックスを追加する新しい INTEL_PROF_DYN_PREFIX 環境変数
- 報告された問題の修正

1.2 製品の内容

*インテル® Parallel Studio XE 2015 Update 3 Composer Edition for Fortran Linux** は、次のコンポーネントで構成されています。

- インテル® Fortran コンパイラー XE 15.0.3。Linux* オペレーティング・システムを実行する IA-32、インテル® 64 アーキテクチャー・システム、およびインテル® Xeon Phi™ コプロセッサーで動作するアプリケーションをビルドします。
- GNU* プロジェクト・デバッガー (GDB) 7.7 (インテル® Xeon Phi™ コプロセッサー用)
- GNU* プロジェクト・デバッガー (GDB) 7.8 (IA-32/インテル® 64 アーキテクチャー用)
- インテル® MKL 11.2 Update 3
- 各種ドキュメント

1.3 インテル® デバッガー (IDB) を削除

インテル® デバッガー (IDB) はこのリリースから削除されました。

代わりに GNU* プロジェクト・デバッガー (GDB) ベースのデバッガーが提供されます。

1.4 動作環境

アーキテクチャー名についての説明は、インテル® アーキテクチャー・プラットフォームの用語 (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、g++-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 2GB (4GB 推奨)
- 4GB 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/テストの場合:
    - インテル® Xeon Phi™ コプロセッサー
    - インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS)
- 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、7.0
    - Fedora* 20、21
    - Red Hat* Enterprise Linux* 5、6、7
    - SUSE Linux Enterprise Server* 11、12
    - Ubuntu* 12.04 LTS (64 ビットのみ)、13.10、14.04 LTS
    - インテル® Cluster Ready

  注: Debian* 6.0 はインテル® コンパイラー 16.0 でサポートを終了する予定です。
- 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 4.1-4.8
    - binutils 2.17-2.24
- -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。

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 -qopenmp 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。

1.5 ドキュメント

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

1.6 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

1.7 日本語サポート

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

日本語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。

1.8 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。

製品をダウンロードした後、次のコマンドを使用して、ダウンロードしたファイルを書き込み可能な任意のディレクトリーに展開します。

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

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

```
./install.sh
```

手順に従ってインストールを完了します。

利用可能なダウンロード・ファイルには各種あり、それぞれ異なるコンポーネントの組み合わせを提供していることに注意してください。ダウンロード・ページを注意深くお読みになり、適切なファイルを選択してください。

新しいバージョンをインストールする前に古いバージョンをアンインストールする必要はありません。新しいバージョンは古いバージョンと共存可能です。

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

2.1 GUI インストーラー

GUI をサポートする Linux* システムで、GUI ベースのインストーラーを利用できるようになりました。GUI をサポートしていない環境では (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。このパッケージを使用するには、インターネット接続が必要です。インターネット・プロキシーを使用している場合は、プロキシーの設定が必要になることがあります。インターネット接続が利用できない環境でインストールする場合は、このオンライン・インストール・パッケージではなく、フルパッケージを利用してください。オンライン・インストーラーをダウンロードしてシェルスクリプトとして保存し、コマンドラインから起動することもできます。

2.2.1 オンライン・インストーラーによりダウンロードされるコンテンツの格納

オンライン・インストーラーは、ほかのシステムにコピーしてオフラインで使用できるように、ダウンロードしたコンテンツを標準インストール・パッケージ形式で格納します。デフォルトのダウンロード・ディレクトリーは `/tmp/<UID>` です。この場所は、オンライン・インストーラーの "`--download-dir [FOLDER]`" コマンドライン・オプションで変更できます。オンライン・インストーラーには、インストールしないでパッケージを作成できるダウンロード専用モードも用意されています。このモードは、"`--download-only`" コマンドライン・オプションで有効になります。

2.2.2 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® Parallel Studio XE Cluster Edition 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® Xeon Phi™ コプロセッサー向けのアプリケーションをビルドする場合のみインストールする必要があります。インテル® MPSS は、インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* のインストール前またはインストール後にインストールできます。

最新バージョンのインテル® MPSS を使用することを推奨します。インテル® Parallel Studio XE for Linux* を登録すると、インテル® ソフトウェア開発製品レジストレーション・センター (http://registrationcenter.intel.com) から入手できます。

ユーザー空間およびカーネルドライバーのインストールに必要な手順については、インテル® MPSS のドキュメントを参照してください。

2.5 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。

2.6.1 非インタラクティブ・カスタム・インストールのサポート

インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* は、「インタラクティブ」インストール中のユーザーの選択肢を (サイレント・インストールに使用できる) 設定ファイルに保存する機能をサポートしています。この設定ファイルは、コマンドライン・インストールで次のオプションを使用すると作成されます。

- `--duplicate config_file_name` (または `-d config_file_name`): 設定ファイルの名前を指定します。フルパスのファイル名が指定された場合、"`--download-dir`" は無視され、設定ファイルがあるディレクトリーにインストール・パッケージが作成されます。
- `--download-dir=dir_name`: 設定ファイルを作成する場所を指定します (オプション)。このオプションを指定しない場合、インストール・パッケージおよび設定ファイルはデフォルトのダウンロード・ディレクトリーに作成されます。

  `/tmp/<UID>/<package_id>`

次に例を示します。

```
l_ccompxe_online_2015.0.0XX.sh --duplicate=icc15_install_config.ini
--download-dir "/temp/custom_pkg_ic15"
```

設定ファイルおよびインストール・パッケージが "`/temp/custom_pkg_ic15`" に作成されます。

2.7 ライセンスサーバーの使用

「フローティング・ライセンス」を購入された場合は、「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 既知の問題と変更点

- インテル® Parallel Studio XE Composer Edition のより新しいメジャーバージョンをアンインストールすると、インテル® Parallel Studio XE Composer Edition の以前のバージョンがシステムに存在している場合でもシンボリック・リンクが削除されます。例えば、インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* をアンインストールすると、インテル® Composer XE 2013 SP1 がシステムに存在している場合でもシンボリック・リンクが削除されます。この問題を回避するには、以前のバージョンのディレクトリー (例えば、`composer_xe_2013_sp1.<n>.<pkg>`) を明示的に参照してください
- アンロックコードを使用したオフラインのリモート・アクティベーションは削除されました。代わりに、ライセンスファイルまたはライセンス・マネージャーを使用してください。
- オンライン・インストーラーのブートストラップ・スクリプトのデフォルトの権限は、umask プロシージャーのデフォルトの権限およびダウンロードするツールのセキュリティ設定に依存します。ダウンロードしたオンライン・インストーラーのブートストラップ・スクリプトに実行権限がない場合は、chmod コマンドを使用してブートストラップ・スクリプトに実行権限を追加してください。
- インテル® Software Manager は Red Hat* Enterprise Linux* 5.0 ではサポートされていません。この制限は将来のアップデートで修正される予定です。

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® MKL のディレクトリーへのシンボリック・リンク
- composerxe – *composer*_xe_2015 ディレクトリーへのシンボリック・リンク
- composer_xe_2015 – インストールされている最新バージョンのインテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* のサブディレクトリーへのシンボリック・リンク
- composer_xe_2015.<n>.<pkg> – 特定のリビジョン番号のファイルが含まれている物理ディレクトリー。<n> はリビジョン番号、<pkg> はパッケージビルド ID。

各 composer_xe_2015 ディレクトリーには、インストールされている最新のインテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* 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

各 composer_xe_2015.<n>.<pkg> ディレクトリーには、特定のリビジョン番号のインテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* 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
- man – man ページ
- mkl – インテル® MKL 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® Parallel Studio XE 2015 Composer Edition for Fortran Linux* 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 15 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

バージョン 12.0 よりも古いコンパイラーでは、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 以降のコンパイラーは、これらの項目を 16 バイトでアラインし、引数が 16 バイト境界でアラインされていると仮定します。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。

3.1.2 割り当て多相コンポーネントの代入に関する新しいライブラリー・サポート (15.0.2)

割り当て多相コンポーネントを含む派生型の明示的/暗示的な代入時に不正な動作が発生する問題を解決するため、コード生成時に呼び出される新しいライブラリー・ルーチンがインテル® コンパイラー 15.0.2 でサポートされました。つまり、該当する代入を含む Fortran ソースをインテル® コンパイラー 15.0.2 以降でコンパイルする場合は、インテル® Fortran コンパイラー 15.0.2 以降のライブラリーとリンクする必要があります。そうでない場合、_alloc_assign_v2 ルーチンがないためリンクエラーになります。一般に、使用するコンパイラーよりも古いバージョンのライブラリーとリンクすべきではありません。

3.2 新機能と変更された機能

一部の言語機能に関する説明はコンパイラーのドキュメントにはまだ含まれていません。必要に応じて、Fortran 2003 規格 (PDF (英語)) および Fortran 2008 規格 (PDF (英語)) を参照してください。

3.2.1 Fortran 2003 の機能

- パラメーター化された派生型

3.2.2 Fortran 2008 の機能

- BLOCK 構造
- EXECUTE_COMMAND_LINE 組込みサブルーチン

3.2.3 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
- OMP CANCEL
- OMP CANCELLATION POINT

節:

- MAP
- DEPEND

プロシージャー:

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.3.1 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。

3.2.3.2 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。

3.2.4 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。

3.2.5 新しい宣言子と追加された宣言子

インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* 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 OPTIMIZATION_PARAMETER INLINE-MAX-TOTAL-SIZE=N
- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-PER-ROUTINE=N

3.2.5.1 BIND(C) と ATTRIBUTES STDCALL を一緒に使用可能

コンパイラー 15.0 では、互換性のあるプロシージャー (宣言に BIND(C) 言語バインド属性を含むプロシージャー) で ATTRIBUTES STDCALL 宣言子を指定することができます。

STDCALL によるその他の影響 (値渡しなど) はありません。必要に応じて、(ATTRIBUTES VALUE ではなく) Fortran 標準の VALUE 属性を利用できます。その他のプラットフォームでは、STDCALL と BIND(C) を一緒に指定しても効果はありません。

3.2.6 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。

- 新しい環境変数 INTEL_PROF_DYN_PREFIX。異なる実行で生成される PGO の ".dyn" ファイルを簡単に区別できるように、任意のプリフィックスを追加できます。インストルメントされたアプリケーションを開始する前に、この環境変数に任意の文字列を設定すると、.dyn ファイル名に指定した文字列がプリフィックスとして追加されます。
- SIMD ベクトル内の "レーン ID" を示す新しい __intel_simd_lane() 組込み関数。この組込み関数は、ショートベクトル・ハイパーオブジェクトのレデューサー実装の記述をサポートします。また、SIMD 対応関数内でリダクション操作の実行を可能にします。

3.2.7 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ように大きな内部バッファーを割り当てます。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の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ように

なりました。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 でこのような 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のと同じ方法で、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。

3.2.8 スタティック解析は非推奨 (廃止予定)

スタティック解析は非推奨 (廃止予定) の機能です。将来のリリースでは削除される予定です。ご意見やお問い合わせは、こちらまでお寄せください。

3.2.9 Fortran ライブラリー・バージョンを取得するための新しいランタイムルーチン

- FOR_IFCORE_VERSION は、Fortran ランタイム・ライブラリー (ifcore) のバージョンを返します。
- FOR_IFPORT_VERSION は、Fortran 移植ライブラリー (ifport) のバージョンを返します。

3.2.10 IA-32 およびインテル® 64 アーキテクチャー向けインテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令セットをサポート (インテル® コンパイラー 15.0.1)

インテル® コンパイラー 15.0.1 では、現在のインテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けインテル® AVX-512 命令のサポートに加えて、インテル® AVX-512 命令対応の IA-32 およびインテル® 64 アーキテクチャー・ベースのプロセッサーでインテル® AVX-512 命令がサポートされるようになりました。

インテル® AVX-512 命令は、

インライン・アセンブリー、/Q[a]xCORE-AVX512 (Windows*) または -[a]xCORE-AVX512 (Linux*/OS X*) コンパイラー・オプションによりサポートされます。

3.2.11 SIMD ループ宣言子で MIN/MAX リダクションをサポート

インテル® コンパイラー 15.0 では、SIMD ループ宣言子で MIN/MAX リダクションをサポートしました。

```
!DIR$ SIMD REDUCTION(MAX:SIMDMAX)
      DO I = 1, SIZE
         IF (X(I) > SIMDMAX) SIMDMAX = X(I)
      END DO
```

```
!DIR$ SIMD REDUCTION(MIN:SIMDMIN)
      DO I = 1, SIZE
         IF (X(I) < SIMDMIN) SIMDMIN = X(I)
      END DO

!DIR$ SIMD REDUCTION(MAX:XMAX)
      DO I = 1, SIZE
         XMAX = MAX (XMAX, X(I))
      END DO

!DIR$ SIMD REDUCTION(MIN:XMIN)
      DO I = 1, SIZE
         XMIN = MIN (XMIN, X(I))
      END DO
```

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 [no]std_value
- -assume ieee_fpe_flags
- -[no-]opt-dynamic-align
- -f[no-]fat-lto-objects
- -f[no-]eliminate-unused-debug-types
- -fast
- -init=snan
- -qopt-report
- -prof-gen=[no]threadsafe

廃止予定のコンパイラー・オプションのリストは、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・オプション」を参照してください。

3.3.1 -o で始まるコンパイラー・オプションは非推奨 (廃止予定)

-o で始まるすべてのコンパイラー・オプションは非推奨 (廃止予定) です。これらのオプションは、-q で始まる新しいオプションに変更されます。例えば、-opt-report は -qopt-report に変更されます。この変更は、-o<text> オプションを出力ファイル名とするサードパーティーのツールとの互換性を向上するために行われました。

3.3.2 -assume std_value がデフォルトでオン

コンパイラー 15.0 では、互換性のないプロシージャー (宣言に BIND(C) 言語バインド属性を含まないプロシージャー) の仮引数に (ATTRIBUTES VALUE ではなく) Fortran 標準の VALUE 属性を指定した場合、デフォルトで Fortran 標準のセマンティクスが適用され、デフォルトの引き渡しメカニズムにより実引数の再定義可能な一時コピーが渡されます。以前のバージョンでは、VALUE は常に実引数を値渡ししていました。コンパイラー 14.0 で -assume std_value は、標準に準拠したセマンティクスを指定し、-standard-semantics が指定された場合に有効になりました。

3.3.3 -standard-semantics と -fp-model strict または -fp-model except により -assume ieee_fpe_flags が有効になる

コンパイラー 15.0 では、-standard-semantics と -fp-model strict または -fp-model except のいずれかが指定されると、-assume ieee_fpe_flags も有効になります。このオプションは、プロシージャーの開始時に浮動小数点例外状態を保存し、終了時に復元します。保存/復元操作はパフォーマンスを大幅に低下させるため、このオプションは浮動小数点例外を操作または照会するアプリケーションでのみ利用すべきです。Fortran 標準の IEEE_ARITHMETIC、IEEE_EXCEPTIONS、IEEE_FEATURES 組込みモジュールを使用する場合、インテル® Fotran コンパイラーでは -fp-model strict を指定する必要があります。

3.3.4 新しい -[no-]opt-dynamic-align コンパイラー・オプション

このオプションを指定すると、コンパイラーはベクトル化されたコード、特に反復回数の多いループのパフォーマンスを最大限に引き出すため、入力データの動的アライメントに基づき条件付きの最適化を実装します。ただし、この最適化により、同じ値のアライメントされたデータとアライメントされていないデータで、ビット単位の結果が異なることがあります。このオプションを指定しない場合、コンパイラーはこれらの最適化を行わず、ビット単位の再現性が保持されます。

3.3.5 -fast オプションの変更

-fast オプションに -fp-model fast=2 が追加されました。このオプションは、gcc との互換性を高め、パフォーマンスのチューニングを容易にします。

3.3.6 新しい -init=snan コンパイラー・オプション

浮動小数点変数をシグナル型 NaN に初期化して、その値が設定される前にフェッチされたらトラップすることで、実行時に初期化されていない変数を探すのに役立つ新しいコマンドライン・オプションです。

3.3.7 新しい最適化レポートのインターフェイス、構造、オプション (インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux*)

インテル® Parallel Studio XE Composer Edition で、4 種類の最適化レポート (-opt-report、-vec-report、-openmp-report、-par-report) が 1 つの -qopt-report インターフェイスに統合されました。情報の表示方法、内容、精度が見直され、どの最適化がコンパイラーにより行われたか、最適なパフォーマンスを達成するにはどのようなチューニングを行えばよいか、ユーザーが理解しやすいように変更されました。

並列ビルドの問題により、このレポートはデフォルトで stderr に出力されません。代わりに、各オブジェクト・ファイルごとにレポートを含む出力ファイル (拡張子 .optrpt) が、コンパイルの出力ディレクトリー (オブジェクト・ファイルが生成されるディレクトリー) に生成されます。この動作を変更するには、-qopt-report-file オプション (例: -qopt-report-file=stderr) を使用します。

-vec-report、-openmp-report、-par-report オプションは廃止予定ですが、現在は -qopt-report オプションの対応する値にマップされます。レポートの内容および形式、デフォルトの出力先は新しい opt-report と同じになります。

変更の詳細についてドキュメントを参照することを強く推奨します。詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・リファレンス」 > 「コンパイラー・オプションのカテゴリーと説明」 > 「最適化レポートオプション」を参照してください。

3.3.8 新しい PGO インストルメンテーション・モード -prof-gen=[no]threadsafe

PGO インストルメンテーションに、OpenMP* 3.1 などの高度な並列化を含むアプリケーションで PGO データ収集を可能にするモードが追加されました。これにより、IA-32 およびインテル® 64 アーキテクチャーで PGO が強化され、インテル® MIC アーキテクチャーのネイティブ・プログラミング・モデルで PGO がサポートされます。

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、GNU* GDB (`gdb-ia` および `gdb-mic`)、インテル® パフォーマンス・ライブラリー、インテル® 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 パラメーター化された派生型で文字長引数の特定の使用法がまだ完全に実装されていない

パラメーター化された派生型 (PDT) では、文字長引数の次の使用法はまだ完全に実装されていません。

- 文字長引数を含む PDT 引数定数
- %RE と %IM は未実装

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 Co-Array 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、IF (THIS_IMAGE() .EQ. 4) THEN のようにループをラップします。

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> 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。

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

8. ストールループを抜けます。

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

9. デバッグを開始します。

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_2015.NN/compiler/lib/mic/libicaf.so
mic0:/lib64/libicaf.so

sudo scp /opt/intel/composer_xe_2015.NN/compiler/lib/mic/libintlc.so
mic0:/lib64/libintlc.so

sudo scp /opt/intel/composer_xe_2015.NN/mpirt/lib/mic/libmpi_mt.so
mic0:/lib64/libmpi_mt.so

sudo scp /opt/intel/composer_xe_2015.NN/mpirt/bin/mic/mpiexec.hydra
mic0:/bin/mpiexec.hydra

sudo scp /opt/intel/composer_xe_2015.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 Co-Array アプリケーションの一部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_2015/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 使用する利点とインテル® MPI ライブラリーとの互換性

インテル® Fortran コンパイラー 14 の Co-Array は、インテル® MPI ライブラリー 5.0 と互換性がありません。Co-Array を使用する場合は、インテル® Fortran コンパイラー 15 以上を使用していることを確認してください。そうでない場合は、インテル® MPI ライブラリー 4.x を使用してください。

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 標準のすべての機能と Fortran 2008 標準の多くの機能をサポートします。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。

- BLOCK 構造
- EXECUTE_COMMAND_LINE 組込みサブルーチン

4 GNU* GDB デバッガー

このセクションでは、インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* とともに提供される GNU* GDB の変更点、新機能、カスタマイズ、および既知の問題をまとめています。

4.1 機能

インテル® Parallel Studio XE 2015 Composer Edition for Fortran Linux* とともに提供される GNU* GDB は、GDB 7.8 を拡張したものです。このデバッガーは、以前のインテル® デバッガーの代わりに提供されています。GDB 7.8 の機能に加えて、次のような新機能が追加されています。

- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーのサポート
- インテル® トランザクショナル・シンクロナイゼーション・エクステンション (インテル® TSX) のサポート
- インテル® Memory Protection Extensions (インテル® MPX) およびインテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) のレジスターサポート
- データ競合の検出 (*pdbx*):
  POSIX* スレッド (pthread) または OpenMP* モデルを使用してスレッド化されたアプリケーションにおけるデータ競合の検出
- 分岐トレーススストア (*btrace*):
  クラッシュ、信号通知、例外などのイベントが発生した後に簡単にバックトラックできるように実行フローで実行された分岐を記録
- Fortran サポートの向上
- インテル® Parallel Studio XE 2015 Update 2 Composer Edition で GNU* GDB 7.7 から GDB 7.8 に変わりました。
- インテル® Processor Trace (インテル® PT) による第 5 世代インテル® Core™ プロセッサーのサポート:

```
(gdb) record btrace pt
```

4.2 GNU* GDB の使用

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

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

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

4.3 ドキュメント

製品とともに提供される GNU* GDB のドキュメントは、以下の場所にあります。

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

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

4.4 既知の問題と変更点

4.4.1 オフロード・デバッグ・セッションの安全な終了方法

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

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

4.4.2 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。

4.4.3 Eclipse* IDE 内で GCB を使用

Eclipse* IDE 内でインテルが提供する GNU* GDB を使用する前に、source コマンドで compilervars.sh を実行する必要があります (コンパイラー環境の設定を参照)。

4.4.4 入れ子された可変長配列はユーザー派生型では正しく動作しない

入れ子された可変長配列を含むユーザー派生型の可変長配列 (VLA) を使用している場合、デバッガーは入れ子された可変長配列を解決できません。この問題は、将来のバージョンで修正される予定です。

5 インテル® MKL

このセクションでは、インテル® MKL の変更点、新機能、および最新情報をまとめています。問題の修正については、こちらを参照してください。

5.1 インテル® MKL 11.2 Update 3 の新機能

- 大規模な SMP システムでスケーリングが向上するように、インテル® MKL のメモリー・マネージャーが拡張されました。
- インテル® MKL のインテル® Xeon Phi™ コプロセッサー・ベースのシステムへの自動オフロードをより細かく制御するため、次の新しいサービス関数が追加されました:mkl_mic_get_meminfo、mkl_mic_get_cpuinfo、mkl_mic_set_flags、mkl_mic_get_flags、mkl_mic_clear_status、mkl_mic_get_status。
- BLAS:
  - すべてのインテル® Xeon® プロセッサーで (D/S)SYMV の並列パフォーマンスを向上
  - 64 ビットのインテル® MKL でインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) の (C/D/S/Z/DZ/SC)ROT パフォーマンスを向上
  - 64 ビットのインテル® MKL でインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) の (C/Z)ROT パフォーマンスを向上
  - 64 ビットのインテル® MKL でインテル® AVX2 の k が大きな ?SYRK/?HERK、?SYR2K/?HER2K、?GEMM の並列パフォーマンスを向上
  - インテル® Xeon Phi™ コプロセッサーで ?SYRK/?HERK、?SYR2K/?HER2K パフォーマンスを向上
- LAPACK:
  - インテル® AVX またはインテル® AVX2 ベースのマルチソケット・システムで特異ベクトル計算の SVD パフォーマンスを向上
  - ピボット演算を使用しない新しい不完全 LU 分解を追加

5.2 インテル® MKL 11.2 Update 2 の新機能

- BLAS:
  - インテル® Xeon Phi™ コプロセッサーで k >> m、k >> n の場合の ?GEMM パフォーマンスを向上
  - 64 ビットのインテル® MKL でインテル® AVX2 の ?HEMM/?SYMM の並列パフォーマンスとシリアルパフォーマンスを向上
  - インテル® AVX2 の ?HERK/?SYRK および ?HER2K/?SYR2K の並列パフォーマンスとシリアルパフォーマンスを向上
  - CBLAS インターフェイスと ?GEMM3M ルーチンに MKL_DIRECT_CALL サポートを追加
  - インテル® AVX-512 の CGEMM パフォーマンスを向上
  - AMD* Opteron* 6000 シリーズで SGEMM および ZGEMM パフォーマンスを向上
  - 64 ビットのインテル® MKL でインテル® AVX2 の CGEMM および ZGEMM のパフォーマンスをわずかに向上
- LAPACK:
  - 固有ベクトルが不要な場合に対称固有値ソルバーのパフォーマンスを最大 3 倍向上
  - 特異ベクトルが必要な場合に ?GESVD パフォーマンスを 2 ～ 3 倍向上
  - 非正方行列でインテル® AVX2 の ?GETRF パフォーマンスを 最大 14 倍向上
  - CNR (条件付き数値再現性) が有効な場合と無効な場合の ?GETRF パフォーマンスのギャップを 5% 以下に減少
  - インテル® AVX2 向けのインテル® Optimized LINPACK Benchmark の共有メモリー (SMP) 実装のパフォーマンスを最大 40% 向上
- クラスター用並列直接法スパースソルバー:
  - 分散 CSR 形式の解による右辺ベクトルの上書き機能を追加
  - 分散 CSR 形式ですべての計算ノードの方程式の解を集約する機能を追加
- インテル® MKL PARDISO:
  - インテル® Xeon Phi™ コプロセッサーで全体的なスケーラビリティーを大幅に向上
  - インテル® Xeon® プロセッサーで解の算出ステップのスケーラビリティーを向上
  - アウトオブコア・モードでメモリー・フットプリントを軽減
  - 因数分解ステップ後に入力行列で使用されたメモリーを解放する機能を追加。これにより、反復改善が不要でユーザーによって無効にされている場合、メモリー消費量が軽減されます。
- 拡張固有値ソルバー:
  - インテル® Xeon® プロセッサーでパフォーマンスを向上
- VSL:
  - サマリー統計:
    - タスクの次元と観測数がほぼ同じ場合に分散/共分散行列計算および相関行列計算ルーチンのパフォーマンスを向上
  - RNG:
    - インテル® Xeon® プロセッサーで Sobol および Niederreiter 準乱数ジェネレーター (RNG) のパフォーマンスを向上
- 畳み込みおよび相関:
  - 3D 畳み込みのパフォーマンスを向上

5.3 インテル® MKL 11.2 Update 1 の新機能

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

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

5.4 インテル® MKL 11.2 の新機能

- インテル® ストリーミング SIMD 拡張命令 4.1 (インテル® SSE4.1) およびインテル® ストリーミング SIMD 拡張命令 4.2 (インテル® SSE4.2) 命令セット対応のすべてのインテル® Atom™ プロセッサー向けの最適化を提供
- インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令セットをサポート (BLAS、DFT、VML の最適化は制限あり)
- BLAS および LAPACK ドメインで verbose モードをサポート (インテル® MKL 関数呼び出しの入力引数をキャプチャー可能)
- インテル® MPI ライブラリー 5.0 をサポート
- インテル® MKL を使用して特定の複雑な問題を解く方法を説明する新しいドキュメント、インテル® MKL クックブック (http://software.intel.com/en-us/mkl_cookbook (英語)) を提供
- すべてのプロセッサーにおいて小行列の ?GEMM パフォーマンスを向上する MKL_DIRECT_CALL または MKL_DIRECT_CALL_SEQ コンパイル機能を追加 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照)
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーにおいて、シングル・ダイナミック・ライブラリー (mkl_rt) をリンクする機能を追加
- カスタマイズ可能なエラーハンドラーを追加 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) リファレンス・マニュアル』の「mkl_set_exit_handler() 」の説明を参照)
- リソース共有メカニズムによりインテル® Xeon Phi™ コプロセッサーの自動オフロード機能を拡張 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) リファレンス・マニュアル』の mkl_mic_set_resource_limit() 関数および MKL_MIC_RESOURCE_LIMIT 環境変数の説明を参照)
- クラスター用並列直接法スパースソルバー:
  - インテル® MKL PARDISO 直接法スパースソルバーの分散メモリーバージョンである、クラスター用並列直接法スパースソルバーを追加
  - 分散行列の行列集約ステップのパフォーマンスが向上
  - 複数の因数分解ステップにおける並べ替え情報の再利用が可能に
  - 分散 CSR 形式、分散行列、RHS、分散ソリューションのサポートを追加
  - 複数の右辺が含まれる式の解の算出をサポート
  - 因数分解および解の算出ステップのクラスターサポートを追加
  - ピュア MPI モードのサポートおよびハイブリッド構成での単一 OpenMP* スレッドのサポートを追加
- BLAS:
  - インテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) 対応の 64 ビット・プロセッサーにおいて ?GEMM のスレッド・パフォーマンスが向上
  - インテル® AVX-512 命令セット用の ?GEMM、?TRSM、DTRMM を最適化
  - インテル® MIC アーキテクチャーにおいて、外積 [large m, large n, small k] および Tall Skinny 型行列 [large m, medium n, small k] の ?GEMM のパフォーマンスが向上

    - インテル® MIC アーキテクチャーにおいて自動オフロードモードの ?TRSM および ?SYMM のパフォーマンスが向上
    - インテル® AVX2 対応の 64 ビット・プロセッサーにおいてレベル 3 BLAS 関数のパフォーマンスが向上
    - コンパイル中に MKL_DIRECT_CALL または MKL_DIRECT_CALL_SEQ が定義されている場合、すべてのプロセッサーにおいて小行列の ?GEMM パフォーマンスが向上 (詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照)
    - インテル® SSE4.2、インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX)、およびインテル® AVX2 命令セット対応の 64 ビット・プロセッサーにおいて、beta=1、k=1 の場合の DGER および DGEMM のパフォーマンスが向上
    - インテル® AVX-512 命令セット用の (D/Z)AXPY を最適化
    - インテル® AVX2 およびインテル® AVX-512 命令セット用の ?COPY を最適化
    - インテル® AVX-512 命令セット用の DGEMV を最適化
    - インテル® AVX およびインテル® AVX2 対応の 64 ビット・プロセッサーにおいて SSYR2K のパフォーマンスが向上
    - すべてのインテル® プロセッサー用の ?AXPBY のスレッド・パフォーマンスが向上
    - インテル® AVX-512 において side=R、uplo={U,L}、transa=N、diag={N,U} の場合の DTRMM のパフォーマンスが向上
- LINPACK:
    - ヘテロジニアス Intel® Optimized MP LINPACK Benchmark for Clusters において行列生成のパフォーマンスが向上
    - Intel® Optimized MP LINPACK Benchmark パッケージのインテル® MIC アーキテクチャー用オフロード・オプションでインテル® AVX2 ホストをサポート
    - インテル® AVX2 対応の 64 ビット・プロセッサーにおいて Intel® Optimized MP LINPACK Benchmark for Clusters パッケージのパフォーマンスが向上
- LAPACK:
    - ?(SY/HE)RDB のパフォーマンスが向上
    - 固有ベクトルが必要な場合の ?(SY/HE)EV のパフォーマンスが向上
    - 固有ベクトルが不要な場合の ?(SY/HE)(EV/EVR/EVD) のパフォーマンスが向上
    - 劣決定 (M が N 未満) の場合の ?GELQF、?GELS および ?GELSS のパフォーマンスが向上
    - ?GEHRD、?GEEV および ?GEES のパフォーマンスが向上
    - LAPACKE インターフェイスにおいて NaN チェッカーのパフォーマンスが向上
    - ?GELSX、?GGSVP のパフォーマンスが向上
    - 固有ベクトルが不要な場合の ?(SY/HE)(EV/EVR/EVD) のパフォーマンスが向上
    - ?GETRF のパフォーマンスが向上
    - M>=N で特異ベクトルが必要ないときの (S/D)GE(SVD/SDD) のパフォーマンスが向上
    - インテル® MIC アーキテクチャーにおいて自動オフロードモードの ?POTRF UPLO=U のパフォーマンスが向上
    - インテル® MIC アーキテクチャーにおいて ?SYRDB の自動オフロードを追加、固有ベクトルが不要な場合に ?SY(EV/EVD/EVR) がスピードアップ

- PBLAS および ScaLAPACK:
  - 大規模な分散ブロッキング係数の P?GEMM ルーチンで自動オフロードが可能に
- スパース BLAS:
  - インテル® AVX-512 命令セット用の SpMV カーネルを最適化
  - スパース BLAS で対角形式を使用する場合のリリースサンプルを追加
  - インテル® SSE4.2、インテル® AVX、およびインテル® AVX2 命令セット対応システムにおいてスパース BLAS レベル 2 およびレベル 3 のパフォーマンスが向上
- インテル® MKL PARDISO:
  - 任意のソルバーステージで後から使用できるようにインテル® MKL PARDISO ハンドルをディスクに格納する機能を追加
  - 非対称行列およびアウトオブコア・モードにピボット制御のサポートを追加
  - 非対称行列およびアウトオブコア・モードに対角抽出のサポートを追加
  - 非線型方程式の反復ソルバーとしてインテル® MKL PARDISO を使用するサンプルを追加
  - 反復改善が無効な場合、因数分解ステージ後にオリジナル行列で割り当てたメモリーを解放する機能を追加
  - 並べ替えアルゴリズムのアウトオブコア (OOC) 部分サイズのメモリー推定向上により、OOC モードの因数分解ステップのパフォーマンスが向上
  - インテル® MKL PARDISO の出力メッセージを変更
  - 構造対称の因数分解中のゼロピボットをサポート
- ポアソン・ライブラリー:
  - 線形方程式を解く前提条件としてインテル® MKL ポアソン・ライブラリーを使用するサンプルを追加
- 拡張固有値ソルバー:
  - 出力メッセージを変更
  - サンプルを変更
  - スパース問題を解くための事前定義インターフェイスに入力および出力 iparm パラメーターを追加
- FFT:
  - インテル® AVX-512 命令セット用の FFT を最適化
  - インテル® MIC アーキテクチャーにおいて 2 のべき乗でない長さのパフォーマンスが向上
- VML: 各ベクトル要素の小数部を計算する v[d|s]Frac 関数を追加
- VSL RNG:
  - 二項乱数ジェネレーターで ntrial=0 をサポート
  - インテル® MIC アーキテクチャーにおいて MRG32K3A および MT2203 BRNG のパフォーマンスが向上
  - インテル® AVX およびインテル® AVX2 命令セット対応のプロセッサーにおいて MT2203 BRNG のパフォーマンスが向上
- VSL サマリー統計:
  - グループ化された/プールされた平均推定 (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) をサポート
- データ・フィッティング: ブレークポイント数が 2 または 3 の場合の自然 3 次スプライン構築関数の不正な動作を修正
- インテル® MKL 環境変数で指定したすべての設定を無視するインテル® MKL モードを追加

- mkl_set_env_mode() ルーチン (インテル® MKL 固有のすべての環境設定を無視するようにインテル® MKL に指示) を呼び出してモードをセットアップすると、MKL_NUM_THREADS、MKL_DYNAMIC、MKL_MIC_ENABLE その他のすべてのインテル® MKL 関連の環境変数が無視される; 必要な引数は mkl_set_num_threads() や mkl_mic_enable() などのインテル® MKL サービスルーチンから設定可能

5.5 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
- API シンボル、引数の順序、リンク行はインテル® MKL 11.2 Beta Update 2 で変更されました。(詳細は、『インテル® マス・カーネル・ライブラリー (インテル® MKL) ユーザーズガイド』を参照してください)
- 廃止予定の項目は、インテル® MKL 11.2 で廃止予定の項目 (英語) を参照してください。

5.6 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。

5.7 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。

6　著作権と商標について

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

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

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

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

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

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