インテル® Fortran Composer XE 2011 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: 321415-003JA

2012 年 4 月 3 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 2011 は、以前「インテル® Fortran コンパイラー・プロフェッショナル・エディション」と呼ばれていた製品の最新バージョンです。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。

Update 10

- インテル® Fortran コンパイラー 12.1.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- 報告された問題の修正

Update 9

- インテル® Fortran コンパイラー 12.1.3
  - `-assume std_intent_in` オプションを追加
  - 派生型 Co-Array の ALLOCATABLE または POINTER コンポーネントにおける制限

- インテル® マス・カーネル・ライブラリー 10.3 Update 9
- 報告された問題の修正

Update 8

- インテル® Fortran コンパイラー 12.1.2
- インテル® マス・カーネル・ライブラリー 10.3 Update 8
- Pardus 2011* のサポート
- 報告された問題の修正

Update 7

- インテル® Fortran コンパイラー 12.1.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 7
  - インテル® MKL の注意事項を更新
- 報告された問題の修正

Update 6

- 新しいトップレベル・フォルダーへの製品のインストール
- クラスターでのインストールのサポート
- インテル® Fortran コンパイラー 12.1.0
  - 追加の Fortran 2003 および Fortran 2008 機能のサポート
  - コンパイラー・オプションの追加
  - 一般的な宣言子の追加と拡張
  - OpenMP* サポートの拡張
  - コンパイラーの主要ドキュメントであるユーザー・リファレンス・ガイドの簡略化と再編成。主な変更点: インテル® コンパイラーの主要機能をまとめた新しいセクション「主な機能」の追加と、コンパイラー・オプション・リファレンスの機能別のグループ化。
- インテル® マス・カーネル・ライブラリー 10.3 Update 6
- 報告された問題の修正

Update 5

- インテル® Fortran コンパイラー 12.0.5
- インテル® マス・カーネル・ライブラリー 10.3 Update 5
- Asianux* のサポート終了予定
- 報告された問題の修正

Update 4

- インテル® Fortran コンパイラー 12.0.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 4
- 報告された問題の修正

Update 3

- インテル® Fortran コンパイラー 12.0.3
  - -sox オプションのデフォルト動作の変更、および含める情報を指定するためのオプションのキーワードの追加
- 日本語のドキュメントと診断メッセージ
- インテル® マス・カーネル・ライブラリー 10.3 Update 3
- 報告された問題の修正

Update 2

- インテル® Fortran コンパイラー 12.0.2
- インテル® マス・カーネル・ライブラリー 10.3 Update 2
- スタティック・セキュリティー解析でのデータファイル作成方法の変更
- 報告された問題の修正

Update 1

- インテル® Fortran コンパイラー 12.0.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 1
- 報告された問題の修正

製品リリース (12.0.0)

- 最初の製品リリース

1.2 製品の内容

インテル® Fortran Composer XE 2011 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 12.1.4。Linux* オペレーティング・システムを実行する IA-32 およびインテル® 64 アーキテクチャー・システムで動作するアプリケーションをビルドします。
- インテル® デバッガー 12.1.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- 各種ドキュメント

1.3 動作環境

アーキテクチャー名についての説明は、http://intel.ly/q9JVjE (英語) を参照してください。

- インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) 対応の IA-32 またはインテル® 64 アーキテクチャー・プロセッサーをベースとするコンピューター (インテル® Pentium® 4 プロセッサー以降、または互換性のあるインテル以外のプロセッサー)
  - 64 ビット・アプリケーションの開発は、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)
- 2GB のディスク空き容量 (すべての機能をインストールする場合)
- 次の Linux* ディストリビューションのいずれか (本リストは、インテル社により動作確認が行われたディストリビューションのリストです。その他のディストリビューションでも動作する可能性はありますが、推奨しません。ご質問は、テクニカルサポートまでお問い合わせください。)
  - Asianux* 3.0、4.0 (サポート終了予定)
  - Debian* 6.0
  - Fedora* 15
  - Red Hat* Enterprise Linux* 4 (サポート終了予定)、5、6
  - SUSE LINUX Enterprise Server* 10、11 SP1
  - Ubuntu* 10.04、11.04
  - インテル® Cluster Ready
  - Pardus* 2011.2 (x64 のみ)
- Linux* 開発ツール・コンポーネント (gcc、g++ および関連ツールを含む)
- `-traceback` オプションを使用するには、`libunwind.so` が必要です。一部の Linux* ディストリビューションでは、別途入手して、インストールする必要があります。

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

- IA-32 アーキテクチャー・システムまたはインテル® 64 アーキテクチャー・システム
- Java* ランタイム環境 (JRE) 5.0 (1.5)
- 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。

説明

- 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。
- 非常に大きなソースファイル (数千行以上) を `-O3`、`-ipo` および `-openmp` などの高度な最適化オプションを使用してコンパイルする場合は、多量の RAM が必要になります。
- 上記のリストにはすべてのプロセッサー・モデル名は含まれていません。リストされているプロセッサーと同じ命令セットを正しくサポートしているプロセッサー・モデルでも動作します。特定のプロセッサー・モデルについては、テクニカルサポートにお問い合わせください。
- 一部の最適化オプションには、アプリケーションを実行するプロセッサーの種類に関する制限があります。詳細は、オプションの説明を参照してください。

1.3.1 Red Hat* Enterprise Linux* 4 のサポート終了予定

インテル® Fortran Composer XE の将来のメジャーリリースでは、Red Hat* Enterprise Linux* 4 はサポートされなくなる予定です。このオペレーティング・システムを使用している場合は、インテルでは新しいバージョンへの移行を推奨しています。

1.3.2 Asianux* のサポート終了予定

インテル® Fortran Composer XE の将来のメジャーリリースでは、Asianux* のすべてのディストリビューションはサポートされなくなる予定です。

1.3.3 IA-64 アーキテクチャー (インテル® Itanium®) 開発の未サポート

本バージョンでは、IA-64 アーキテクチャー (インテル® Itanium®) システム上、または IA-64 アーキテクチャー・システム向けの開発をサポートしていません。インテル® コンパイラー 11.1 ではまだサポートされています。

1.4 ドキュメント

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

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

1.5 日本語サポート

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

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

日本語サポート版を英語のオペレーティング・システムで使用する場合や日本語のオペレーティング・システムで英語サポート版を使用する場合は、http://intel.ly/pla2A5 (英語) の説明を参照してください。

1.6 テクニカルサポート

インテル® ソフトウェア開発製品レジストレーション・センターでラインセンスを登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

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

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

2 インストール

本製品のインストールには、有効なライセンスファイルまたはシリアル番号が必要です。本製品を評価する場合には、インストール時に [製品を評価する (シリアル番号不要)] オプションを選択してください。

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

```
./install.sh
```

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

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

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

```
./install.sh
```

手順に従ってインストールを完了します。

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

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

2.1 クラスターでのインストール

インストールするマシンにインテル® Cluster Studio XE のライセンスがあり、クラスターメンバーの場合、「フル・インストール」を選択すると、そのクラスターのアクセス可能なすべてのノードに製品がインストールされます。「カスタム・インストール」を選択すると、現在のノードのみにインストールするオプションを選択できます。

2.2 インテルのアクティベーション・ツールを使用した製品のアクティベーション

この製品リリースでは、インテルのアクティベーション・ツール “Activate” が `/opt/intel/ActivationTool/Activation/` ディレクトリーにインストールされます。

インストール中に評価用ライセンスまたはシリアル番号を使用、または [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製品をインストールした場合、製品を購入した後にこのアクティベーション・ツール (`/opt/intel/ActivationTool/Activation/Activate`) を使用して製品をアクティベートできます。これにより、評価版から製品版へ移行することができます。このツールを使用するには、次のコマンドを実行します。

```
$ /opt/intel/ActivationTool/Activation/Activate [シリアル番号]
```

2.3 サイレントインストール

自動インストール、「サイレント」インストール機能についての詳細は、http://intel.ly/ngVHY8 (英語) を参照してください。

2.4 ライセンスサーバーの使用

「フローティング・ライセンス」を購入された場合は、ライセンスファイルまたはライセンスサーバーを使用したインストール方法について http://intel.ly/oPEdEe (英語) を参照してください。この記事には、多様なシステムにインストールすることができる FLEXlm* ライセンス・マネージャーに関する情報も記述されています。

2.5 既知のインストールの問題

- Linux* ディストリビューションの Security-Enhanced Linux (SELinux) 機能を有効にしている場合は、インテル® Fortran コンパイラーをインストールする前に `SELINUX` モードを `permissive` に変更する必要があります。詳細は、Linux* ディストリビューションのドキュメントを参照してください。インストールが完了したら、`SELINUX` モードを元の値に戻してください。

- 一部の Linux* バージョンでは、自動マウントデバイスに "実行" 許可がなく、インストール・スクリプトを直接 DVD から実行すると、次のようなエラーメッセージが表示されることがあります。

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

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

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

  その後、再度インストールを行ってください。
- 「システム要件」に記述されているように、本バージョンでは、IA-32 およびインテル® 64 アーキテクチャー・ベースのシステムで Debian* または Ubuntu* をサポートしています。ただし、ライセンス・ソフトウェアの制約上、Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システム上では、インストール時に [製品を評価する (シリアル番号不要)] オプションで IA-32 コンポーネントをインストールできません。これは、[製品を評価する (シリアル番号不要)] オプションを使用する場合のみの問題です。シリアル番号、ライセンスファイル、フローティング・ライセンス、その他のライセンス・マネージャー操作、およびオフラインでのアクティベーション操作 (シリアル番号を使用) には、影響はありません。Debian* または Ubuntu* を搭載したインテル® 64 アーキテクチャー・システムで、本バージョンの IA-32 コンポーネントの評価が必要な場合は、インテル® ソフトウェア評価センター (http://intel.ly/nJS8y8 (英語)) で評価版のシリアル番号を入手してください。

2.6 インストール先フォルダー

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

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

インテル® Fortran Composer XE 2011 の以前のリリースと Update 6 では、トップレベルのインストール・ディレクトリーが異なりますが、引き続き `composerxe` シンボリック・リンクを使用して最新の製品インストールを参照することができます。

`<install-dir>` 以下には次のサブディレクトリーがあります。

- `bin` - インストールされている最新バージョンの実行ファイルへのシンボリック・リンク
- `lib` - インストールされている最新バージョンの `lib` ディレクトリーへのシンボリック・リンク
- `include` - インストールされている最新バージョンの `include` ディレクトリーへのシンボリック・リンク
- `man` - インストールされている最新バージョンの man ページが含まれているディレクトリーへのシンボリック・リンク
- `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_2011_sp1` ディレクトリーへのシンボリック・リンク
- `composer_xe_2011_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 2011 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_2011_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_2011_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 2011 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_2011_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 2011 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 2011 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。

2.7 削除/アンインストール

製品の削除 (アンインストール) は、製品をインストールしたユーザー (root または非 root ユーザー) で実行してください。インストールに sudo を使用した場合は、アンインストールの際にも使用する必要があります。インストールされているパフォーマンス・ライブラリー・コンポーネントを残したまま、コンパイラーのみを削除することはできません。

1. 端末を開いて、<install-dir> 以外のフォルダーに移動 (cd) します。
2. その後、次のコマンドを使用します。<install-dir>/bin/uninstall.sh
3. 画面の指示に従ってオプションを選択します。
4. 別のコンポーネントを削除するには、ステップ 2 と 3 を繰り返します。

同じバージョンのインテル® C++ コンパイラーをインストールしている場合は、C++ コンパイラーもリストに表示されます。

3 インテル® Fortran コンパイラー

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

3.1 互換性

一般に、インテル® Fortran コンパイラー Linux* 版の以前のバージョン (8.0 以降) でコンパイルされたオブジェクト・コードおよびモジュールは、バージョン 12 でもそのまま使用できます。ただし、次の例外があります。

- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた CLASS キーワードを使用して多相変数を宣言しているソースは再コンパイルする必要があります。
- マルチファイルのプロシージャー間の最適化 (`-ipo`) オプションを使用してビルドされたオブジェクトは再コンパイルする必要があります。
- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた REAL(16)、REAL*16、COMPLEX(16)、COMPLEX*32 データ型を使用しているオブジェクトは再コンパイルする必要があります。
- バージョン 10.0 よりも前のコンパイラーを使用してインテル® 64 アーキテクチャー用にビルドされたモジュール変数を含むオブジェクトは再コンパイルする必要があります。Fortran 以外のソースからこれらの変数を参照する場合、不正な先頭の下線を削除するように外部名を変更する必要があります。
- バージョン 11.0 よりも前のコンパイラーを使用してコンパイルされた、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。

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 以前のリリースでは、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 12.0 (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 12.0 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。

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

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 (http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf) および Fortran 2008 規格 (http://j3-fortran.org/doc/standing/links/007.pdf) を参照してください。

3.2.1 Fortran 2003 の機能

- FINAL サブルーチン
- 型バインド・プロシージャーの GENERIC キーワード
- 汎用インターフェイスの名前は派生型と同じ名前を使用可能
- ポインター代入の境界の仕様と境界の再マップリスト
- SOURCE= を使用した ALLOCATE (Update 6 では多相ソースをサポート)

3.2.2 Fortran 2008 の機能

- 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: IMAGE_INDEX、LCOBOUND、NUM_IMAGES、THIS_IMAGE、UCOBOUND
    - **注:** ATOMIC_DEFINE および ATOMIC_REF はサポートしていません
- 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
- (Update 6) 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、または組込み関数 NULL への参照の場合、無視されます。
- (Update 6) 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。

3.2.3 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。

3.2.3.1Coarray アプリケーションのデバッグ方法

Co-Array アプリケーションのデバッグ方法を以下に説明します。

1. デバッグするコードの前にストールループを追加します。
   例:

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

   ループがコンパイラーによって削除されないように VOLATILE を使用します。問題が 1 つのイメージでのみ見つかった場合は、ループを
   `IF (THIS_IMAGE() .EQ.4) THEN`
   のように囲みます。
2. デバッグをオンにしてコンパイルおよびリンクします (-g)。
3. アプリケーションを実行するマシンに少なくとも N + 1 (N はアプリケーションのイメージ数) のターミナルウィンドウを作成します。
4. ターミナルウィンドウでアプリケーションを開始します。
   `linuxprompt> ./my_app`
5. その他のターミナルウィンドウで、デフォルトのディレクトリーがアプリケーションの実行ファイルの場所と同じになるように設定します。1 つのウィンドウで "ps" コマンドを使用してプログラムを実行しているプロセスを調べます。

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

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

   他のプロセスは以下のようになります。

   ```
   <ユーザー名>  25653 25650 98 15:06 ?        00:00:49 my_app
   <ユーザー名>  25654 25651 97 15:06 ?        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、最初の行では 25653)。

   "my_app" P1、P2、P3、… を実行している N 個のプロセスの PID を呼び出します。
6. 各ウィンドウで (最初のウィンドウを除く) デバッガーを開始し、アタッチしたときにプロセスを停止するように設定します。

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

   または

   ```
   linuxprompt> gdb
   ```

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

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

   または

   ```
   (gdb) attach <P1>
   ```
8. ストールループを抜けます。

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

   または

   ```
   (gdb) set WAIT_FOR_DEBUGGER = .false.
   ```
9. デバッグを開始します。

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

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

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

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

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

3.2.3.2Co-Array の既知の問題

このバージョンでは、以下の機能は動作しません。

- 別のイメージを参照している Co-Array の配列スライスの出力 (WRITE、PRINT など)。配列全体の参照または単一要素は機能します。
- REAL(16) または COMPLEX(16) の Co-Array のデフォルト初期化
- 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

3.2.4 スタティック・セキュリティー解析機能 (旧: ソースチェッカー) にはインテル® Inspector XE が必要

バージョン 11.1 の「ソースチェッカー」機能が拡張され、「スタティック・セキュリティー解析」に名称が変更されました。スタティック・セキュリティー解析を有効にするコンパイラー・オプションはバージョン 11.1 と同じですが (例: `-diag-enable sc`)、解析結果がコンパイラー診断結果ではなく、インテル® Inspector XE で表示可能なファイルに出力されるようになりました。

3.2.5 新しい宣言子と追加された宣言子 (Update 6)

インテル® Composer XE 2011 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 VECTOR
- NOFUSION
- PARALLEL 宣言子で FIRSTPRIVATE 節を指定可能
- SIMD 宣言子で FIRSTPRIVATE 節または LASTPRIVATE 節を指定可能

3.2.6 OpenMP* の変更 (Update 6)

インテル® Composer XE 2011 では、OpenMP* サポートに関して次の点が変更されています。

- OpenMP* 3.1 のサポート
- TASKYIELD 宣言子
- ATOMIC 宣言子に新しい節を追加
- TASK 宣言子で FINAL 節および MERGEABLE 節を指定可能

3.2.7 その他の変更

- 識別子によるクロスリファレンス付きのソース・リスト・ファイルを作成するための機能の追加
- ガイド付き自動並列化
- より高速でやや精度が低い算術ライブラリー関数を使用するためのオプション
- プロセッサーのモデルや製造元に関係なく一貫した結果を返す算術ライブラリー関数を使用するためのオプション
- ビルドの依存関係をファイルに出力するための機能の追加

3.2.7.1スタティック・セキュリティー解析の動作変更

インテル® Composer XE 2011 に含まれる `inspxe-runsc` コマンドライン・ユーティリティーが変更されました。この変更は、インテル® Composer XE 2011 を使用してスタティック・セキュリティー解析 (SSA) を実行する場合にのみ影響します。SSA を使用しない場合や、このユーティリティーを使用せずに SSA を実行する場合には影響ありません。SSA はインテル® Parallel Studio XE 2011、インテル® Fortran Studio XE 2011 またはインテル® C++ Studio XE 2011 でのみ利用できます。そのため、これらの製品をお使いでない場合は影響ありません。

`inspxe-runsc` は、アプリケーションのビルド方法を示す**ビルド仕様**を実行します。通常、ビルド仕様ファイルは、ビルドを実行して、実際に行われたコンパイルとリンクを記録することにより生成されます。`inspxe-runsc` は、インテル® コンパイラーを SSA モードで使用して、再度この処理を行います。SSA 結果はリンクステップで生成されるため、`inspxe-runsc` で複数のリンクステップを持つビルドが含まれるビルド仕様を実行すると、複数の SSA 結果が生成されます。

インテル® Composer XE 2011 およびインテル® Composer XE 2011 Update 1 の `inspxe-runsc` は、すべての SSA 結果を同じディレクトリーに生成します。リンクが複数ある場合、これは、1 つのプロジェクトの SSA 結果は同じディレクトリーに 1 つだけでなければならないという規則に違反します。新しいバージョンの `inspxe-runsc` は、リンクステップごとの結果を個別のディレクトリーに生成することで、この規則に従っています。ディレクトリー名は、リンクされるファイルの名前を基に付けられます。2 つの実行ファイル file1.out と file2.out をビルドするプロジェクトのビルド仕様の場合、以前のバージョンの `inspxe-runsc` では、file1 の結果と file2 の結果 (例えば r000sc と r001sc) が同じディレクトリーに作成されます。新しいバージョンの `inspxe-runsc` でも結果は 2 つ作成されますが、file1 の結果は “My Inspector XE results – file1/r000sc”、file2 の結果は “My Inspector XE results – file2/r000sc” というように別々のディレクトリーに作成されます。2 つの結果のディレクトリーは同じ親ディレクトリーの下に作成されます。

`inspxe-runsc` には、結果の作成場所を指定するための `-result-dir` (`-r`) コマンドライン・スイッチがあります。このスイッチの動作が変更されました。以前は、r000sc のように結果が作成されるディレクトリーの名前を指定していましたが、現在は、“My Inspector XE Results - name” のように結果が作成されるディレクトリーの親ディレクトリーを指定し

ます。つまり、-r スイッチのディレクトリー名は、結果の生成される場所から 2 つ上のディレクトリーのものになります。

inspxe-runsc のこの変更により、結果ディレクトリーが効率良く移動します。この変更に伴い、ユーザーによる対応が必要になります。-r スイッチを指定して inspxe-runsc を呼び出すスクリプトを使用している場合は、新しい動作に合わせて、-r スイッチの引数を変更してください。また、新しいバージョンの inspxe-runsc によって生成される SSA 結果が、以前のバージョンの inspxe-runsc によって生成された結果と同じディレクトリーに保存されることがないように、以前の結果ファイルを新しいディレクトリーに移動する必要があります。以前のバージョンの inspxe-runsc でリンクステップが 1 つのみのビルド仕様を実行した結果は、"My Inspector XE results - name" という形式のディレクトリーに移動します。この操作を行わないと、新しく作成される結果ですべての問題が "新規" として表示されます。以前のバージョンの inspxe-runsc で複数のリンクステップを含むビルド仕様を実行した場合、SSA ではさまざまな問題がありましたが、これらの問題は新しいバージョンを使用することで解決されます。この場合、以前の結果のうち最も新しいものを "My Inspector XE results - name" という形式の新しいディレクトリーに (1 つのディレクトリーに 1 つの結果が含まれるように) コピーします。これにより、新しいバージョンで作成される結果に以前の問題ステート情報が正しく適用される可能性が高くなります。

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。

3.3.1 インテル® Composer XE 2011 の新規および変更されたコンパイラー・オプション (Update 6 以降)

- -align [no]qcommons
- -assume [no]std_intent_in (Update 9)
- -f[no-]asynchronous-unwind-tables
- -axCORE-AVX-I
- -axCORE-AVX2
- -f[no-]fma
- -f[no-]merge-debug-strings
- -fopenmp
- -gdwarf-3
- -march=atom
- -march=core-avx-i
- -march=core-avx2
- -march=corei7-avx
- -march=corei7
- -march=Pentium-m
- -opt-mem-layout-trans[=n]
- -xCORE-AVX-I
- -xCORE-AVX2
- -xSSSE3_ATOM

3.3.2 インテル® Composer XE 2011 の新規および変更されたコンパイラー・オプション (Update 5 まで)

- -assume [no]fpe_summary
- -assume [no]old_ldout_format
- -coarray
- -coarray-num-images
- -fzero-initialized-in-bss
- -fimf-absolute-error
- -fimf-accuracy-bits

- -fimf-arch-consistency
- -fimf-max-error
- -fimf-precision
- -fvar-tracking
- -fvar-tracking-assignments
- -gen-dep
- -gen-depformat
- -guide
- -guide-data-trans
- -guide-file
- -guide-file-append
- -guide-opts
- -guide-par
- -guide-vec
- -list
- -list-line-len
- -list-page-len
- -opt-args-in-regs
- -par-runtime-control
- -prof-value-profiling
- -profile-functions
- -profile-loops-report
- -show=keyword
- -simd
- -sox=keyword
- -standard-semantics

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。

3.3.3 -assume [no]std_intent_in の追加 (Update 9)

`-assume [no]std_intent_in` オプションは、Fortran 規格に従って `INTENT(IN)` 属性の仮引数が呼び出し間で変更されないことをコンパイラーが仮定するかどうかを決定します。デフォルトは `std_intent_in` で、コンパイラーは `INTENT(IN)` 引数が変更されないことを仮定します。`nostd_intent_in` の場合は仮定しません。`-standard-semantics` オプションを指定すると、`-assume std_intent_in` オプションが指定されます。

3.3.4 -sox オプションの追加キーワード、デフォルトの変更 (Update 3)

オブジェクト・ファイルおよび実行ファイルに使用されたコンパイラー・オプションとプロシージャーのプロファイル情報を追加するための `-sox` オプションは、インライン展開された関数のリストを含めたり、プロシージャーのプロファイル情報を除外したり指定できるようになりました。

`-sox` の構文は次のように変更されました。

```
-[no-]sox
-sox=keyword[,keyword]
```

keyword には、`inline` または `profile` のいずれかを指定できます。キーワードなしで `-sox` を指定すると、以前のバージョンとは異なり、コマンドライン・オプションの情報のみが追加されます。以前のリリースと同じ動作にするには、`-sox=profile` を使用してください。`-sox` オプションはコマンドラインで複数回指定することができますが、その場合は左から右の順に解釈されます。

3.4 その他の変更および注意

3.4.1 最適化レポートがデフォルトで無効に設定

バージョン 11.1 以降、コンパイラーは、ベクトル化、自動並列化、OpenMP* スレッド化ループに関する最適化レポートメッセージをデフォルトで表示しないようになりました。これらのメッセージを表示するには、`-diag-enable vec`、`-diag-enable par`、`-diag-enable openmp` を設定するか、`-vec-report`、`-par-report`、`-openmp-report` を使用する必要があります。

また、バージョン 11.1 以降、最適化レポートメッセージは `stdout` ではなく、`stderr` に送られます。

3.4.2 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。

3.4.3 OpenMP* レガシー・ライブラリーの削除

本リリースでは、OpenMP* のレガシー・ライブラリーが削除されました。"互換性がある" ライブラリーのみ提供されます。

3.4.4 RANF 移植関数の組込み関数への変更

移植ライブラリーの RANF 関数は非標準の乱数ジェネレーターです。コンパイラー 12.0 では、RANF は新しいハイパフォーマンスな組込み関数として実装されています。プログラムで USE IFPORT を使用して RANF にアクセスしている場合、変更はありません。古いバージョンが使用されます。プログラムで USE IFPORT を使用していない場合、または INTRINSIC RANF を使用している場合、古いバージョンとは異なるシーケンスを返す新しいバージョンが使用されます。RANF のシードはこれまでどおり移植サブルーチン SRAND によって設定されます。インテルは、標準の組込み関数 RANDOM_NUMBER の使用を推奨していますが、既存のアプリケーションとの互換性を確保するために RANF も提供しています。

3.4.5 既知の問題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。

3.4.6 割り当て可能な配列と OpenMP* の PRIVATE/FIRSTPRIVATE

コンパイラーは、OpenMP* の PRIVATE 指示節または FIRSTPRIVATE 指示節で定義された割り当て可能な配列を正しく初期化できません。この問題は将来のアップデートで修正される予定です。これらを組み合わせたときに問題が発生する場合は、`-switch omp3_private` オプションを追加してください。ただし、これは一時的な回避方法です。この問題の Intel issue ID は DPD200160978 です。

3.5 Fortran 2003 および Fortran 2008 機能の概要

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

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

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

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

- ユーザー定義の派生型 I/O
- パラメーター化された派生型
- CLASS オブジェクトのデフォルトの初期化
- MODULE PROCEDURE でのキーワード MODULE の省略
- 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: IMAGE_INDEX、LCOBOUND、NUM_IMAGES、THIS_IMAGE、UCOBOUND
    - **注:** ATOMIC_DEFINE および ATOMIC_REF はサポートしていません
- CONTIGUOUS 属性
- ALLOCATE の MOLD キーワード
- DO CONCURRENT
- OPEN の NEWUNIT キーワード
- G0 および G0.d フォーマット編集記述子
- 無制限のフォーマット項目繰り返しカウント指定子
- CONTAINS セクションは空にすることも可能
- 組込みプロシージャー: BESSEL_J0、BESSEL_J1、BESSEL_JN、BESSEL_YN、BGE、BGT、BLE, BLT、DSHIFTL、DSHIFTR、ERF、ERFC、ERFC_SCALED、GAMMA、HYPOT、IALL、IANY、IPARITY、IS_CONTIGUOUS、LEADZ、LOG_GAMMA、MASKL、MASKR、MERGE_BITS、NORM2、PARITY、POPCNT、POPPAR、SHIFTA、SHIFTL、SHIFTR、STORAGE_SIZE、TRAILZ
- 組込みモジュール ISO_FORTRAN_ENV の追加: ATOMIC_INT_KIND、ATOMIC_LOGICAL_KIND、CHARACTER_KINDS、INTEGER_KINDS、INT8、INT16、INT32、INT64、LOCK_TYPE、LOGICAL_KINDS、REAL_KINDS、REAL32、REAL64、REAL128、STAT_LOCKED、STAT_LOCKED_OTHER_IMAGE、STAT_UNLOCKED
- ALLOCATABLE または POINTER 属性を持たない OPTIONAL 仮引数は、対応する実引数に ALLOCATABLE 属性があるのに割り当てられない場合、POINTER 属性があるのに関連付けが解除されている場合、または NULL 組込み関数への参照の場合、無視されます。
- 仮引数がプロシージャー・ポインターの場合、そのポインターの有効な参照先か、または組込み関数 NULL への参照である実引数に関連付けられます。実引数がポインターではない場合、仮引数に INTENT (IN) 属性が含まれていなければなりません。

4 インテル® デバッガー (IDB)

次の注意事項は、IA-32 アーキテクチャー・システムおよびインテル® 64 アーキテクチャー・システムで実行するインテル® デバッガー (IDB) のグラフィカル・ユーザー・インターフェイス (GUI) についてです。このバージョンでは、`idb` コマンドは GUI を起動します。コマンドライン・インターフェイスを起動するには、`idbc` を使用します。

4.1 Java* ランタイム環境の設定

インテル® IDB デバッガーのグラフィカル環境は、Java* アプリケーションで構築されており、実行には Java* ランタイム環境 (JRE) が必要です。デバッガーは、5.0 (1.5) をサポートしています。

配布元の手順に従って JRE をインストールします。

最後に、JRE のパスを設定する必要があります。

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

4.2 デバッガーの起動

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

```
idb
```

または

```
idbc
```

(必要に応じて)

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

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

例: `chmod +x <application_bin_file>`

4.3 その他のドキュメント

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

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

4.4 デバッガー機能

4.4.1 インテル® Fortran Composer XE 2011 Update 6 の変更点

4.4.1.1 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。

4.4.1.2 プロセッサーのデバッグレジスター (ハードウェア・ベース) を使用したウォッチポイント

インテル® Fortran Composer XE 2011 Update 6 (IDB 12.1) では、プロセッサーのデバッグレジスターを使用したウォッチポイントが完全にサポートされています。設定方法は、使用するプロセッサー・アーキテクチャーにより異なります。次のアーキテクチャー関連の制限があります。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。

4.4.2 IDB の主な機能

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

4.4.2.1 [Threads (スレッド)] ウィンドウ

- データ共有検出の向上
- OpenMP* 3.0 のサポート
- Linux* OS の同期関数のサポート
- データ共有検出の解析パフォーマンスの向上

4.4.2.2 ブレークポイント機能の拡張

この拡張により、まだロードされていない共有ライブラリーのルーチンにブレークポイントを設定することができるようになりました。設定されたブレークポイントは可能な限り認識されます。(アドレス、ファイル、シンボル名がないなどの理由により) 認識されないブレークポイントには GUI で黄色い三角が表示されます。コマンドラインでは <PENDING> と表示されます。(多重定義された関数のブレークポイントなどの) 両義性は直ちに解決され、複数認識されます。このようなブレークポイントは、GUI では設定したブレークポイントをノードとするツリーとして表示されます。コマンドラインでは <MULTIPLE> として表示され、認識されます。この機能は、コマンドラインでは GDB モードでのみ利用できます。

4.4.2.3 コマンド solib-search-path の実装

コマンドライン・デバッガーidbc と GUI デバッガーのコマンドウィンドウで、gdb コマンドの `solib-search-path` がサポートされるようになりました。このコマンドは、イメージや共有ライブラリーが通常の場所 (`$LD_LIBRARY_PATH` など) にない場合、これらを検索します。

`solib-search-path` コマンドの使用方法については、次のコマンドを実行してコマンドライン・ヘルプを参照してください。

```
(idb) help set solib-search-path
```

```
(idb) help show solib-search-path
```

または、次のように省略形で指定することもできます。

```
(idb) h set sol
(idb) h sho sol
```

4.4.2.4逆アセンブル表示用の新しいコマンド

IDB デバッガーでは、アセンブラー・ウィンドウまたはコマンドウィンドウで 2 種類の逆アセンブルビューを利用できるようになりました。

コマンドウィンドウでは、次の新しいコマンドを利用できます。

```
(idb) set disassembly-flavor [att|intel]
(idb) show disassembly-flavor
```

また、次のコマンドを実行するとこのコマンドのヘルプを参照できます。

```
(idb) help set
(idb) help show
```

GUI では、アセンブラー・ウィンドウで [Change Style (スタイルの変更)] を右クリックしてインテルと ATT スタイルを切り替えることができます。ATT は AT&T スタイルを表します (GNU* スタイルとも呼ばれています)。

4.5 既知の問題と変更点

4.5.1 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、セクション 3.2.3.1 「Co-Array アプリケーションのデバッグ方法」を参照してください。

4.5.2 [Signals (シグナル)] ダイアログが動作しない

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

4.5.3 GUI のサイズ調整

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

4.5.4 $cdir ディレクトリー、$cwd ディレクトリー

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

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

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

4.5.5 info stack の使用

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

```
info stack [num]
```

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

4.5.6 $stepg0 のデフォルト値の変更

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

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

4.5.7 一部の Linux* システムでの SIGTRAP エラー

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

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

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

4.5.8 MPI プロセスのデバッグには idb GUI は使用不可

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

4.5.9 GUI でのスレッド同期ポイントの作成

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

4.5.10 IA-32 アーキテクチャー向けのスタック・アライメント

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

4.5.11 GNOME 環境の問題

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

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

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

4.5.12 オンラインヘルプへのアクセス

システムで IDB デバッガー GUI の [Help (ヘルプ)] メニューからオンラインヘルプにアクセスできない場合は、http://intel.ly/ng91l0 から Web ベースのドキュメントを利用できます。

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

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

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

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

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

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

4.5.16 コアファイルのデバッグ

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

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

または

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

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

4.5.17 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 SIGSEGV 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。

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

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

4.5.19 Pardus* システムでスレッド情報が利用できない

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

5 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® マス・カーネル・ライブラリー (インテル® MKL) の変更点、新機能、および最新情報をまとめています。問題の修正の詳細は、http://intel.ly/neQlw2 (英語) を参照してください。

5.1 インテル® MKL 10.3 Update 10 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 (インテル® AVX) 対応の 32 ビット・プログラムで dznrm2 および dnrm2 のパフォーマンスが向上
- LAPACK: LAPACK バージョン 3.4.0 のサポートを追加
- データ適合: インテル® Xeon® プロセッサー E7-4870/E5-2690 において、以下の領域における SearchCells1D() 関数のパフォーマンスが向上。
    - 補間点の数が 32 を超える任意の非一様および擬一様領域
    - 補間点の数が 32 未満のすべての種類の領域

5.2 インテル® MKL 10.3 Update 9 の新機能

- LAPACK: 非常に小さなサイズ (~10 x 10) で [C/Z]GEEV のパフォーマンスが向上
- FFT: 大幅なパフォーマンス向上のためにインプレースの 1D FFT 実関数をスレッド化
- FFT: 32 ビットのオペレーティング・システムを実行するインテル® Xeon® プロセッサー E5 ファミリーのシステムにおける 2 の累乗の倍精度複素 1D FFT のスケーラビリティー向上のために新しいアルゴリズムを追加
- 乱数ジェネレーター: 新しいインテル® マイクロアーキテクチャー (開発コード名: Ivy Bridge) ベースのプロセッサーで利用可能なハードウェアをサポートする、RdRand 命令に基づく非決定性乱数ジェネレーターをサポート
- ベクトル算術関数: インテル® Core™ プロセッサーにおいて、Erf() 関数および Pow3o2() 関数のパフォーマンスが向上
- データ適合: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、スプラインベースの評価、微分、積分ルーチンのパフォーマンスが向上

5.3 インテル® MKL 10.3 Update 8 の新機能

- データ適合コンポーネント: ベクトルスプラインの構築、セル探索や二分探索、スプライン補間の評価、微分、積分の 1 次元アルゴリズムをカバーする新しいデータ適合関数のセットを追加。以下のサポートを含む。
    - 1 次スプライン、2 次スプライン、3 次スプライン、ステップワイズ法、ユーザー定義スプライン
    - 最適なパフォーマンスのために構成パラメーターを用いたセル探索
    - ユーザー定義補間 (内挿) および補外 (外挿)
    - ベクトル値関数
    - 列優先および行優先格納形式
- 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 (インテル® AVX) 対応のコア数の多いシステムにおいて、非常に疎な行列の CSR (Compressed Sparse Row) 形式の行列-ベクトル乗算 (?CSRMV) のパフォーマンスが向上
- FFT: インテル® AVX 対応システムにおいて、1D 倍精度 FFT のパフォーマンスが向上
- 統計関数: インテル® Core™ プロセッサーにおいて、分散共分散行列および相関行列の計算 (FAST 法) のパフォーマンスとスケーラビリティーが向上
- 問題の修正

5.4 インテル® MKL 10.3 Update 7 の新機能

- BLAS: 最近のすべてのインテル® Xeon® プロセッサーにおいて、出力行列が小さく外積が大きい (つまり、入力行列が矩形) の場合に DSYRK/SSYRK のマルチスレッド・パフォーマンスが向上

- BLAS: 最近のすべてのインテル® Xeon プロセッサーにおいて、小さな問題 (<10、beta =1) で ?GEMM のパフォーマンスが向上
- BLAS: 32 ビットのプログラムを実行するインテル® Xeon プロセッサー 5500/5600/7500 番台 において、INCX=1 の小さな問題で DSCAL のパフォーマンスが向上
- BLAS のような拡張: 正方行列のインプレース転置のキャッシュ効率とスレッディングの向上
- PARDISO: PARDISO 用の独立したスレッド化コントロールの追加; mkl_domain_set_num_threads() 関数での MKL_DOMAIN_PARDISO の使用
- ポアソン・ライブラリー: 2D/3D 周期的境界条件のサポートの追加
- ドキュメント・ディレクトリーへのリンク・ライン・アドバイザーの追加
- libtool などのスクリプトツールとともに使用するコマンドライン・リンク・ツールの追加
- mkl_domain_set_num_threads() 関数でドメインの指定に使用する定数の名前を変更 (例: MKL_BLAS は MKL_DOMAIN_BLAS に変更); MKL_PARDISO を除き、古い名前も継続して使用可能
- 問題の修正

5.5 インテル® MKL 10.3 Update 6 の新機能

- スパース BLAS: mkl_?csrbsr 変換関数に BSR 形式から CSR 形式への変換時にゼロの要素を検出し削除する新しいオプションを追加
- 問題の修正

5.6 インテル® MKL 10.3 Update 5 の新機能

- BLAS: パフォーマンスの向上: {S,C,Z}TRSM (インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサー); {S,D}GEM2VU (インテル® AVX 対応プロセッサー、インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー 5500 番台)
- BLAS: スケーリングの向上: ?TRMV (すべてのアーキテクチャーの大規模行列); DGEMM (インテル® Xeon® プロセッサー 5400 番台でスレッド数が奇数の場合)
- LAPACK: LAPACK 3.3.1 の拡張とそれぞれの LAPACKE インターフェイスに対応
- LAPACK: 一般固有値問題に使用される ?SYGST と ?HEGST のパフォーマンスが向上
- LAPACK: LU 分解された行列の逆行列のパフォーマンスが向上 (?GETRI)
- PARDISO: 転置と共役転置の解算出 (ATx=b と AHx=b) の追加; 圧縮されたスパース列 (CSC) 形式のサポート
- PARDISO: MKL_PARDISO_OOC_MAX_SWAP_SIZE 環境変数とインコア PARDISO を使用して、必要なメモリー量が利用可能なメモリー量をやや上回る場合に、アウトオブコア PARDISO のパフォーマンスが向上
- 最適化ソルバー: RCI Trust-Region ソルバーに Inf と NaN のチェックを追加
- FFT: インテル® SSE3 以降に対応したインテル® プロセッサーにおいて、サポートされているすべての精度で 2x2x2 から 10x10x10 の小さな立方体に対する 3D FFT のパフォーマンスが向上
- FFT サンプル: インテル® MKL の DFTI と FFTW の一般的な使用例を示すサンプルコードを変更
- VSL: インテル® Core™ i7-2600 プロセッサーを搭載した 64 ビットのオペレーティング・システムのマシンにおいて、単精度の MT19937 および MT2203 基本乱数ジェネレーターのパフォーマンスが向上
- VSL: インテル® Core™ i7-2600 プロセッサーおよびインテル® Xeon® プロセッサー 5400 番台で SOBOL 準乱数ジェネレーターの整数バージョンのパフォーマンスが向上。
- 問題の修正

5.7 インテル® MKL 10.3 Update 4 の新機能

- BLAS: インテル® Xeon® プロセッサー 5400 番台以降において DTRMM のパフォーマンスが向上
- BLAS: すべての 64 ビット対応プロセッサー、特にインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーにおいて DTRSM のパフォーマンスが向上
- LAPACK: LAPACK 3.3.1 リリースでの問題の修正に対応
- OOC PARDISO: アウトオブコア処理に必要なメモリー量の推定が向上
- FFT: スレッディングの改善による 1D 実数 FFT スケーリングの向上
- FFT: 新しいシングル・ダイナミック・ライブラリー・リンク・モデルを使用するように C および Fortran の FFT サンプルを更新
- VML: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、すべての精度で、Hypot 実関数と Abs 複素関数の単精度のパフォーマンス拡張バージョン、および Arg、Div、Mul、MulByConj 複素関数のパフォーマンスが向上
- サービス関数: インテル® MKL のサービス関数の追加と拡張 (詳細は、http://intel.ly/pkUQXI (英語) のオンライン・リリースノートを参照してください)
- 問題の修正

5.8 インテル® MKL 10.3 Update 3 の新機能

- BLAS: インテル® Xeon® プロセッサー 5400 番台を搭載した 32 ビットの Windows* システムにおいて DSYRK、DTRSM、DGEMM のマルチスレッド・パフォーマンスが向上
- LAPACK: 対称/エルミート行列関数および補助関数における連立線形方程式ソルバーの向上、CS (余弦/正弦) 分解を含む Netlib LAPACK 3.3 の実装
- PARDISO: 0 ベースの順列ベクトルの入力をサポート
- PARDISO: pardisoinit() ルーチンのドキュメント化
- PARDISO: 複数の右辺 (RHS) を含む PARDISO のシリアル・パフォーマンスが向上
- PARDISO: 小さな行列のパフォーマンスを向上させる解のステップの並列化の独立制御。詳細は、iparm(25) の説明を参照してください。
- PARDISO: 後方代入の減少による右辺全体の部分解計算。詳細は、iparm(31) の説明を参照してください。
- FFT: 最大 3 ～ 7 次元の実数 FFT 変換の実装
- FFT: 2 つの実数配列として表される分割複素数データを使用した多次元複素数変換の並列化
- クラスター FFT: FORTRAN 90 インターフェイスの拡張による実数-複素数変換への対応、および新しいサンプルの追加
- VML: 新しい Pack/Unpack 複素関数と Gamma/LGamma 実関数の追加
- VML: インテル® Xeon® プロセッサー 5600 番台およびインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーで、すべての関数におけるショートベクトル (< 100) の演算、すべての関数におけるアライメントされていない入力ベクトルの演算、sPow2o3 関数、拡張パフォーマンス (EP) バージョンの Add および Sub 複素関数のパフォーマンスが向上
- VSL: 乱数ジェネレーター (RNG) ストリームからメモリーへの保存、またはメモリーからの復元を行うための関数の追加
- VSL: 新しい UniformBits32 関数および UniformBits64 関数の追加
- VSL: MT2203 BRNG でサポートされる一意のストリーム数を 1024 から 6024 に拡張
- 問題の修正

注: インテル® MKL に含まれる GMP* 数学関数は、将来のリリースでは削除されます。

5.9 インテル® MKL 10.3 Update 2 の新機能

- BLAS: インテル® Xeon® プロセッサー 5600 番台において転置関数のパフォーマンスが向上
- BLAS: 転置ルーチンのサンプルの追加
- FFT: 必要な精度の関数のみをリンクすることでアプリケーションのフットプリントを小さくする方法を示した Fortran サンプルの追加
- FFT: CCE ストレージを使用するインプレース実数変換にストライドの一貫性チェックを追加
- FFT: 多次元変換のスレッド化の追加
- VSL: クアッドコア インテル® Xeon® プロセッサー 5500 番台において単精度/倍精度の多変量ガウス分布乱数ジェネレーターのパフォーマンスが向上
- VML: インテル® Xeon® プロセッサー 5500 番台において Add、Mul、Sub 関数のインプレース操作のパフォーマンスが向上
- 問題の修正

5.10 インテル® MKL 10.3 Update 1 の新機能

- PARDISO/DSS: F90 オーバーロード API の追加 (詳細は、インテル® MKL リファレンス・マニュアルを参照してください)
- PARDISO: 統計情報をより見やすく改良
- スパース BLAS: 最新のインテル® プロセッサーにおいて ?BSRMM 関数のパフォーマンスが向上
- FFT: 負のストライドのサポート
- FFT サンプル: DFTI と FFTW3 の両方のインターフェイスを使用した分割複素 FFT の C および Fortran サンプルの追加
- VML: SSE2 および SSE3 対応システムにおいて、インプレースの Add/Sub/Mul/Sqr 実関数のパフォーマンスが向上
- ポアソン・ライブラリー: ポアソン・ライブラリー関数のデフォルトの動作をシリアルから並列に変更
- 問題の修正

5.11 インテル® MKL 10.3 の新機能

- BLAS
  - 一度に 2 つの行列-ベクトル積を計算するための新しい関数: [D/S]GEM2VU、[Z/C]GEM2VC
  - 混合精度の一般的な行列-ベクトル積を計算するための新しい関数: [DZ/SC]GEMV
  - 2 つのスケールされたベクトルの和を計算するための新しい関数: *AXPBY
  - 主要関数においてインテル® AVX による最適化: SMP LINPACK、レベル 3 BLAS、DDOT、DAXPY
- LAPACK
  - 行優先順に対応した LAPACK 用の C インターフェイス
  - 1 つの新しい計算ルーチン (*GEQRFP)、2 つの新しい補助ルーチン (*GEQR2P と *LARFGP)、LAPACK 3.2.1 のアップデートを含む Netlib LAPACK 3.2.2 との統合
  - 主要関数においてインテル® AVX による最適化: DGETRF、DPOTRF、DGEQRF
- PARDISO
  - マルチコア環境で問題と解のステップのパフォーマンスが向上
  - スパースの右辺の解算出と部分解ベクトルを出力する部分解算出の追加
  - アウトオブコア (OOC) 因数分解のパフォーマンスが向上
  - ゼロベース (C スタイル) の配列インデックスのサポート
  - 対称行列のスパースデータ構造で行列の対角上のゼロが不要
  - 新しい ILP64 PARDISO インターフェイスにより、LP64 ライブラリーにリンクされている場合に LP64 と ILP64 の両バージョンを使用可能

    - OOC モードでディスクにファイルを格納するのに必要なメモリーを並べ替え直後に予測可能
- スパース BLAS
    - 形式変換関数ですべてのデータ型に対応 (単精度/倍精度の実数/複素数データ)、および関数の戻り値として並べ替えあり/並べ替えなし配列を使用可能
- FFT
    - すべての 1D/2D/3D FFT においてインテル® AVX による最適化
    - SSE4.2 命令セットをサポートするすべてのシステムにおいて、基数が混在する単精度/倍精度データの 2D/3D FFT のパフォーマンスが向上
    - 2D/3D FFT における 2 つの実数配列として表される分割複素数データのサポート
    - 長さが大きな素数である 1D 複素数-複素数変換のサポート
- VML
    - (ax+b)/(cy+d) の計算を行うための新しい関数。a、b、c、d はスカラー、x、y は実数ベクトル: v[s/d]LinearFrac()
    - 主要関数においてインテル® AVX による最適化
    - デノーマル数をゼロに設定するための新しいモデル、複素ベクトルのオーバーフロー・サポート、各 VML 関数に対して精度を設定するための追加パラメーターを含む新しい関数
- VSL
    - 新しいサマリー統計関数群。基礎統計、共分散/相関関係、プールされたグループ/部分/厳密な共分散/相関関係、分位数/変量分位数、外れ値検出アルゴリズム、欠測値をサポート
        - パフォーマンスが最適化されたアルゴリズム: 欠測値をサポートするための MI アルゴリズム、厳密な共分散を計算するための TBS アルゴリズム、外れ値を検出するための BACON アルゴリズム、(変量データの) 分位数を計算するための ZW アルゴリズム、プールされた共分散を計算するための 1PASS アルゴリズム
    - SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
    - インテル® AVX による最適化: MT19937 と MT2203 BRNG
- ランタイムにディスパッチされるダイナミック・ライブラリーの追加により、ランタイムに検出された CPU またはライブラリー関数呼び出しに応じて、依存性のあるライブラリーを動的にロードする単一のインターフェイス・ライブラリーへのリンクが可能
- カスタム・ダイナミック・ライブラリー・ビルダーは、Linux* および Mac OS* X オペレーティング・システムにおいてランタイムにディスパッチされるライブラリーを使用
- 新しいディレクトリー構造により、インテル® MKL ライブラリーとインテル® Parallel Studio XE 製品ファミリーの統合が単純化され、これまでの "em64t" ディレクトリーが "intel64" ディレクトリーに変更
- スパースソルバー機能をインテル® MKL のコア・ライブラリーに完全統合。また名前に "solver" を含むライブラリーを製品から削除

5.12 既知の問題

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

5.13 注意事項

インテル® MKL の将来のバージョンでは以下の変更が予定されています。「テクニカルサポート」を参照してください。

- ファイル名に solver を含むライブラリーの内容をコア・ライブラリーに移動する予定です。これらの solver ライブラリーはその後削除される予定です。

- インテル® MKL の GMP (GNU* Multiple Precision) 関数インターフェイスは、将来のリリースでは削除される予定です。
- タイミング関数 mkl_set_cpu_frequency() は古い関数です。代わりに、『インテル® MKL リファレンス・マニュアル』で説明されている mkl_get_max_cpu_frequency()、mkl_get_clocks_frequency()、mkl_get_cpu_frequency() を使用してください。
- PARDISO ドメインを指定するために定義されている MKL_PARDISO 定数は、mkl_domain_set_num_threads() 関数で使用できなくなりました。代わりに、MKL_DOMAIN_PARDISO を使用してください。
- 畳み込みルーチンと相関ルーチンは、将来のリリースでは 10.2 Update 3 との下位互換性はなくなります。

5.14 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。

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を許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む)に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」

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

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

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

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

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

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

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。