インテル® C++ コンパイラー 11.1 Mac OS* X 版 プロフェッショナル・エディション インストール・ガイドおよび リリースノート

資料番号: 321413-002JA
2009 年 7 月

目次

1 概要

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。報告されている問題の修正リストは、インテル® ソフトウェア開発製品レジストレーション・センターで提供されている各アップデート製品に含まれる README.TXT ファイルを参照してください。

Update 1

- -O0 の動作の変更に関する注意事項の追加
- 報告されている問題の修正

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

1.2 製品の内容

インテル® C++ コンパイラー 11.1 Mac OS X 版プロフェッショナル・エディション*には、次のコンポーネントが含まれています。

- インテル® プロセッサー・ベースの Mac OS X オペレーティング・システムで動作するアプリケーションのビルド用インテル® C++ コンパイラー
- インテル® デバッガー
- インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP)
- インテル® マス・カーネル・ライブラリー (インテル® MKL)
- インテル® スレッディング・ビルディング・ブロック (インテル® TBB)
- Xcode* 開発環境への統合
- 各種ドキュメント

1.3 動作環境

- インテル® プロセッサー・ベースの Apple* Mac* システム
- RAM 1GB (最小)、RAM 2GB (推奨)
- 3GB のディスク空き容量
- Mac OS X 10.5.6 と Xcode 3.1.2 または Mac OS X 10.5.7 と Xcode 3.1.3
- gcc* 4

注: 高度な最適化オプションを使用する場合や大規模なプログラムの場合、メモリーやディスク容量など、追加でリソースが必要になることがあります。

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。

1.5 テクニカルサポート

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

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

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

2 インストール

初めて製品をインストールする場合は、インストール中にシリアル番号の入力が求められますので、あらかじめご用意ください。製品のインストールと使用には、有効なライセンスが必要です。

Xcode を使用する場合、サポートされているバージョンの Xcode がインストールされていることを確認してください。将来、新しいバージョンの Xcode をインストールする場合は、そのインストール後にインテル® C++ コンパイラーを再インストールする必要があります。

製品のインストール、変更、アンインストールを行うには、管理者権限または "sudo" 権限が必要です。

DVD 版の場合は、DVD を挿入し、 DVD のディスク・イメージ・ファイル (`m_cproc_p_11.1.xxx.dmg`) まで移動してダブルクリックします。ダウンロード版の場合は、ダウンロード・ファイル (`m_cproc_p_11.1.xxx.dmg`) をダブルクリックします。

手順に従ってインストールを完了します。

2.1 インストール先フォルダー

11.1 製品は、前のバージョンとは異なる構成でフォルダーにインストールされます。新しい構成を以下に示します。一部含まれていないフォルダーもあります。

- `<root>/intel/Compiler/11.1/`*`xxx`*`/`
  - `bin`
    - `ia32`
    - `intel64`
  - `include`
    - `ia32`
    - `intel64`
  - `lib`
  - `perf_headers`
  - `Frameworks`
    - `ipp`
    - `mkl`
    - `tbb`
  - `Documentation`
  - `man`
  - `Samples`

`<root>` はデフォルトでは `/opt`、 *`xxx`* は 3 桁のリビジョン番号です。`bin`、`include`、`lib` 配下のフォルダーは次のとおりです。

- `ia32`: 32 ビットのインテル® プロセッサー・ベースの Mac OS X システム上で動作するアプリケーションのビルド用コンパイラー
- `intel64`: 64 ビットのインテル® プロセッサー・ベースの Mac OS X システム上で動作するアプリケーションのビルド用コンパイラー (インテル® 64 アーキテクチャーとも呼ばれます)

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、所定のバージョンのフォルダーが共有されます。

2.2 インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) 暗号化ライブラリーのインストール

インテル® IPP は、暗号化ルーチンのライブラリーを含むオプションのコンポーネントを提供します。暗号化ライブラリーのインストールと使用には、別途ライセンスが必要です。このライセンスは、インテル® IPP のライセンスの登録後に無償で入手できます。ただし、輸出規制が適用されます。詳細は、http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-cryptography-library/ (英語) を参照してください。

暗号化ライブラリーを入手し、Xcode 環境で使用する場合、暗号化ライブラリー・インストーラーを実行し、[カスタム] インストールを選択して、Xcode 用にインストールされたコンパイラーのディレクトリーをインストール・パスとして指定する必要があります。デフォルトでは、`/Developer/opt/intel/Compiler/11.1/xxx` です。また、暗号化ライブラリーをコマンドラインからも使用する場合は、インストーラーを再度実行して、デフォルト/通常のインストールを指定します。

2.3 インストール後の製品の配置変更

付属のスクリプトを使用して、インストールされた製品のコマンドライン・インターフェイスをディスクの別の場所に移動することができます。

1. 端末を開きます。
2. コンパイラーのインストール・フォルダーに移動 (cd) します (例: `/opt/intel/Compiler/11.1/xxx`)。
3. コマンドを入力します。
   `./move_cproc.sh <new-install-location>`
   `<new-install-location>` は新規のディレクトリー・パスです。

このスクリプトはすべてのファイルを移動し、必要に応じてシンボリック・リンク、環境変数、起動スクリプトを更新します。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古いパスにインストールされている場合は、両コンパイラーとも新しい場所に移動されます。

Xcode 統合は、Xcode ディレクトリー・ツリーを別の場所にドラッグアンドドロップするだけで移動できます。移動した Xcode ディレクトリー・ツリーを使用してコマンドプロンプトから `idb` を使用する場合は、http://software.intel.com/en-us/articles/running-idb-from-command-line-after-relocating-xcode-environment/ (英語) を参照して、必要なその他の手順を確認してください。`idb` は Xcode IDE 内では利用できないことに注意してください。

2.4 削除/アンインストール

パフォーマンス・ライブラリー・コンポーネントを残して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. 次のコマンドを入力します: `<install-dir>/uninstall_cproc.sh`
3. 画面の指示に従ってオプションを選択します。

`root` でログインしていない場合は `root` パスワードの入力が求められます。同じバージョンのインテル® Fortran コンパイラーをインストールしている場合は、Fortran コンパイラーも削除されます。

3 インテル® C++ コンパイラー

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

3.1 新機能と変更された機能

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。

- C++0x からの追加機能
- C++ ラムダ関数
- 10 進浮動小数点
- IPP オプションを使用した valarray の実装
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- OpenMP* 3.0 のサポート
- C++ コンパイラーのデフォルトモードが gcc のデフォルトモードにより近くなりました。混在する宣言やコードなど、一部の C99 機能はデフォルトではオンではありませんが、-std=c99 を使用して有効にすることができます。

3.2 新規および変更されたコンパイラー・オプション

- -mkl[=lib]

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。

3.2.1 -O0 は -mp を含みません

バージョン 11.1 では、最適化を無効にする `-O0` オプションが 浮動小数点精度を最大化する `-mp` を含意しなくなりました。 `-mp` スイッチは廃止予定です。そのため、浮動小数点精度の影響を受けやすいアプリケーションには、明示的に `-fp-model` オプションを指定することを推奨します。

3.3 その他の変更

3.3.1 最適化レポートがデフォルトで無効

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` を指定するか、`-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。

3.3.2 環境設定スクリプトの変更

コマンドライン・ビルド環境の設定に使用されていた icc.sh (icc.csh) スクリプトが変更されました。以前のバージョンでは、cc または cce のいずれかのルート・ディレクトリーを選択することによってターゲット・プラットフォームが選択されました。バージョン 11.1 では、スクリプトは 1 つのみで、引数を指定してターゲット・プラットフォームを選択します。

コマンドの形式は以下のとおりです。

`source /opt/intel/Compiler/11.1/`*`xxx`*`/bin/iccvars.sh `*`argument`*

`xxx` はリビジョン番号です。*`argument`* は `ia32` または `intel64` のいずれかです (「インストール先フォルダー」を参照)。コンパイラーを異なるパスにインストールしている場合は、適切なフォルダーを指定しください。コンパイラー環境を構築すると、インテル® デバッガー (idb) 環境も構築されます。

3.3.3 OpenMP ライブラリーのデフォルトが "compat" に変更

バージョン 10.1 では、新しい OpenMP ライブラリー・セットが追加され、アプリケーションは、インテル® コンパイラーと gcc コンパイラーの両方からの OpenMP コードを使用することが可能でした。この "互換" ライブラリーは古い "レガシー" ライブラリーよりも高いパフォーマンスを提供します。バージョン 11.x では、互換ライブラリーが OpenMP アプリケーションのデフォルト・ライブラリーとして使用されるようになりました。`-openmp-lib compat` と等価です。古いライブラリーを使用する場合は、`-openmp-lib legacy` を指定してください。

"レガシー" ライブラリーは、インテル® コンパイラーの将来のリリースからは削除される予定です。

4 インテル® デバッガー (IDB)

4.1 既知の問題

4.1.1 Dwarf と Stabs デバッグ・フォーマット

デバッガーでは、デバッグ情報が Dwarf フォーマットの実行ファイルのデバッグのみをサポートしており、Stabs デバッグ・フォーマットはサポートしていません。gcc と g++ で Dwarf 出力を生成するには、コンパイルコマンドで `-gdwarf-2` フラグを使用します。インテル® コンパイラー (icc と ifort) では、`-g` フラグで Dwarf デバッグ・フォーマットを作成します。

4.1.2 コンパイル要件

Xcode 2.3 より、Dwarf デバッグ情報はオブジェクト (.o) ファイルに保存されています。これらのオブジェクト・ファイルは、デバッグ対象のアプリケーションに関連した情報を得るためにデバッガーによりアクセスされます。そのため、シンボリック・デバッグが利用可能でなければなりません。

次のように、1 つのコマンドでプログラムがコンパイルされ、リンクされた場合、

```
icc -g -o hello.exe hello.c
```

コンパイラーによりオブジェクト・ファイルは生成されますが、コマンドが完了する前に削除されます。このコマンドで作成されたバイナリーファイルにはデバッグ情報は含まれません。アプリケーションをデバッグ可能にするには、次の 2 つの方法があります。

アプリケーションを 2 つの手順でビルドして .o ファイルを明示的に作成します。

```
icc -c -g -o hello.o hello.c

icc -g -o hello.exe hello.o
```

または、-save-temps コンパイラー・スイッチを使用して作成された .o ファイルが削除されないようにします。

```
icc -g -save-temps -o hello.exe hello.c
```

デバッガーは "dsymutil" ユーティリティーの出力を使用しません。

4.1.3 非ローカルのバイナリーファイルとソースファイルのアクセス

デバッガーは、ネットワーク・マウント・ファイル・システム (NFS など) からバイナリーファイルにアクセスできません。次のようなエラーメッセージが表示されます。

Internal error: cannot create absolute path for: /home/me/hello (内部エラー: /home/me/hello の絶対パスを作成できません。)

You cannot debug "/home/me/hello" because its type is "unknown". ("/home/me/hello" はデバッグできません。型が "不明" です。)

また、デバッガーは、ネットワーク・マウント・ファイル・システム (NFS など) からソースファイルにアクセスできません。次のようなエラーメッセージが表示されます。

Source file not found or not readable, tried... (ソースファイルが見つからないか読み取りできません...)

./hello.c

/auto/mount/site/foo/usr1/user_me/c_code/hello.c

(Cannot find source file hello.c (ソースファイル hello.c が見つかりません))

The file-path specified will be correct. (指定されたファイルパスは修正されます。)

ファイルは、ローカル・ファイル・システム (例: ネットワークでマウントされていないシステム) にコピーして使用してください。

4.1.4 fork アプリケーションのデバッグ

fork を呼び出すアプリケーションの子プロセスのデバッグはまだサポートされていません。

4.1.5 exec アプリケーションのデバッグ

$catchexecs 制御変数はサポートされていません。

4.1.6 スナップショット

マニュアルで説明されているスナップショットはまだサポートされていません。

4.1.7 最適化コードのデバッグ

最適化コードのデバッグはまだ完全にはサポートされていません。最適化を有効にしてコードをコンパイルすると、一部の関数名、パラメーター、変数、パラメーターと変数の内容をデバッガーが参照できないことがあります。

4.1.8 ウォッチポイント

書き込みアクセスを検知するよう作成されたウォッチポイントは、元の値と同一の値が書き込まれたときにはトリガーしません。これは、Mac OS X オペレーティング・システムの制限によるものです。

ウォッチポイントの実装には、SIGSEGV シグナルではなく SIGBUS シグナルがデバッガーで使用されているため、SIGBUS シグナルをキャッチするシグナル・ディテクターを作成することができません。

4.1.9 グラフィック・ユーザー・インターフェイス (GUI)

本バージョンのデバッガーでは GUI はサポートされていません。

4.1.10 MPP デバッグの制限

マニュアルで説明されている MPP デバッグはサポートされていません。

4.1.11 関数ブレークポイント

関数に設定されたブレークポイント ("stop in" コマンドを使用して設定) では、最初の文でユーザープログラムの実行が停止されないことがあります。これは、生成された Dwarf デバッグ情報で関数プロローグに関する情報が不十分なために発生します。回避策として、"stop at" コマンドで該当文にブレークポイントを設定します。

コンパイラーは "__dyld_func_lookup" への呼び出しを関数のプロローグの一部として作成します。この関数にブレークポイントを設定すると、デバッガーはその位置で停止しますが、ローカル変数値が有効ではありません。回避策として、ブレークポイントを関数内の最初の文に設定します。

4.1.12 コアファイルのデバッグ

コアファイルのデバッグは、サポートされていません。

4.1.13 ユニバーサル・バイナリーのサポート

ユニバーサル・バイナリーのデバッグはサポートされています。デバッガーは IA-32 上の IA-32 Dwarf セクションのバイナリーと、インテル® 64 上の IA-32 セクションまたはインテル® 64 セクションのデバッグをサポートしています。

4.1.14 $threadlevel デバッガー変数

マニュアルでは、"$threadlevel" デバッガー変数について「On Mac OS* X, the debugger supports POSIX threads, also known as pthreads. (Mac OS* X では、デバッガーは POSIX スレッド (pthreads とも呼ばれる) をサポートしています。」という記述があります。この文章では別の種類のスレッドもサポートされているようにもとれますが、 そうではなく、POSIX スレッドのみが Mac OS X 上でサポートされています。

4.1.15 オープンファイル記述子の制限

デバッガーはデバッグ対象の .o ファイルを開いてデバッグ情報を読み取るため、ファイルの制限を緩和する必要があります。

Mac OS では、オープンできるファイル記述子の数を 256 に制限していますが、 次のように上限を上げることができます。

```
ulimit -n 2000
```

デバッガーを起動する前に、このコマンドを使用してオープンファイル記述子の数の制限を上げてください。

これは、デバッガーが多くのファイルに対してオープンファイル記述子の制限数を適切に共

有できるようになるまでの回避策です。

4.1.16 $cdir ディレクトリー、$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。

4.1.17 info stack の使用

デバッガーコマンド "info stack" は、以下のオプションの構文では現在、負のフレームカウントをサポートしていません。

```
info stack [num]
```

フレームカウント num が正の場合、最内 num フレームを出力します。カウントが負またはゼロの場合、(最外 num フレームを出力するのではなく) フレームを出力しません。

4.1.18 $stepg0 のデフォルト値が変更

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。この設定では、"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インします。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 インテル® インテグレーテッド・パフォーマンス・プリミティブ

このセクションでは、インテル® C++ コンパイラー・プロフェッショナル・エディションに同梱されているインテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) の変更点、新機能、および最新情報をまとめています。

5.1 新機能と変更された機能

- インテル® Advanced Vector Extensions (インテル® AVX) のサポート
- インテル® Core™ i7 プロセッサーの新しい最適化とスレッド化制御/最適化をサポート
- 3D 画像処理: 3D 幾何学変換、3D フィルター
- 新しいデータ圧縮関数 API
- RSA_SSA1.5 と RSA_PKCSv1.5 の新しいインテル® IPP 暗号化サポート
- PNG 形式サポートを追加する UIC (Unified Image Classes) と DXT1、DXT3、DXT5 画像圧縮をサポートする新しい機能
- 球面調和関数とパーリンノイズ生成関数を含む高度な光関数
- MPEG-2 のシーン解析、VC1 の輝度補償とオーバーラップ・スムージングを含む新しいビデオ・コーディング分野の向上
- 信号処理、画像処理、ストリング処理、C++/C# 言語サポートのサンプルを \Samples フォルダーに追加。その他のサンプルは、http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-intel-ipp-intel-ipp-sample-code/ (英語) からダウンロードできます。
- 廃止予定の API のさらに多くのリファンレス情報がリファレンス・マニュアルとヘッダーファイルに追加

5.2 既知の制限事項

一部の生成関数 ("ippg" 接頭辞) で状況依存ヘルプが動作しないことがあります。

5.3 別途ダウンロード可能なインテル® IPP 暗号化ライブラリー

インテル® IPP 暗号化ライブラリーは別途ダウンロード可能です。ダウンロードとインストールの手順については、
http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/ (英語) を参照してください。

5.4 インテル® IPP コードサンプル

インテル® IPP コードサンプルは、
http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/ で Windows 版、Linux 版、Mac OS 版のダウンロード・パッケージが用意されています。

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

6 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® C++ コンパイラー・プロフェッショナル・エディションに同梱されているインテル® マス・カーネル・ライブラリー (インテル® MKL) の変更点、新機能、および最新情報をまとめています。

6.1 本バージョンでの変更

6.1.1 新機能

- LAPACK 3.2
  - 238 個の新しい LAPACK 関数
  - 超精密反復法の改良
  - ハウスホルダー QR 因数分解の非負対角
  - 低プロファイル行列でのハイパフォーマンス QR とハウスホルダー反射
  - 高速で正確な新しいヤコビ法 SVD
  - 矩形フル圧縮形式のルーチン
  - ピボットコレスキー
  - 混合精度反復法の改良 (コレスキー)
  - より安定した DQDS アルゴリズム
- DZGEMM 拡張 BLAS 関数の実装 (http://www.netlib.org/blas/blast-forum/ の説明を参照)。リファレンス・マニュアルの BLAS セクションの `*gemm` 関数ファミリーの説明を参照してください。
- PARDISO で実数、複素数、単精度データをサポート

6.1.2 ユーザービリティー/インターフェイスの向上

- スパース行列形式変換ルーチン:
  - CSR (3-配列バリエーション) ↔ CSC (3-配列バリエーション)
  - CSR (3-配列バリエーション) ↔ 対角形式
  - CSR (3-配列バリエーション) ↔ スカイライン
- Fortran95 BLAS と LAPACK のコンパイル・モジュール・ファイル (.mod) が含まれています。
  - モジュールは、インテル® Fortran コンパイラーで事前にビルドされており、インクルード・ディレクトリーにあります (フルパス情報については、インテル® MKL ユーザーズ・ガイドを参照してください)。

    - ほかのコンパイラー用のソースも提供されています。
    - インターフェイスについてのドキュメントは、インテル® MKL ユーザーズ・ガイドを参照してください。
- FFTW3 インターフェイスが直接メイン・ライブラリーに統合されました。
    - デフォルトのインテル® Fortran コンパイラー規則と名前修飾で互換性のないコンパイラーでラッパーを作成するためのソースコードも提供されています。
    - 詳細は、リファレンス・マニュアルの付録 G を参照してください。
- DFTI_DESCRIPTOR_HANDLE が型の名前を表すようになりました。ユーザープログラムで型として参照できます。
- 最適化ソルバードメインのヤコビ行列計算ルーチンにパラメーターが追加され、ユーザーデータにアクセスできるようになりました (詳細は、リファレンス・マニュアルの djacobix 関数の説明を参照してください)。
- 64 ビット・アーキテクチャーでインテル® MKL の単精度 BLAS 関数 (頭文字 "s" または "c" の関数) から 64 ビット浮動小数点精度関数へのインターフェイス・マッピング呼び出しが追加されました (詳細は、インテル® MKL ユーザーズ・ガイドの「sp2dp」を参照してください)。
- 互換ライブラリー (「ダミーライブラリー」) が削除されました。

6.1.3 パフォーマンスの向上

- インテル® 64 アーキテクチャー用にさらにスレッド化された BLAS レベル 1、2 関数
    - レベル 1 関数 (ベクトル-ベクトル): (CS,ZD,S,D)ROT、(C,Z,S,D)COPY、(C,Z,S,D)SWAP
        - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.7-4.7 倍向上
        - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 14-130 倍向上
    - レベル 2 関数 (行列-ベクトル): (C,Z,S,D)TRMV、(S,D)SYMV、(S,D)SYR、(S,D)SYR2
        - キャッシュのデータ位置に応じて、4 コアのインテル® Core™ i7 プロセッサー上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 1.9-2.9 倍向上
        - キャッシュのデータ位置に応じて、24 コアのインテル® Xeon® プロセッサー 7400 番台システム上でバージョン 10.1 Update 1 に対してパフォーマンスが最大 16-40 倍向上
- インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー (5300 番台、5400 番台、7400 番台) で、DSYRK の 32 ビット逐次バージョンに導入された再帰アルゴリズムのパフォーマンスが最大 20% 向上。
- インテル® Xeon® 7460 プロセッサーで、大規模な問題の LU 因数分解 (DGETRF) がバージョン 10.1 Update 1 に対して 25% 向上。また小規模な問題でも劇的に向上。
- BLAS *TBMV/*TBSV 関数でレベル 1 BLAS 関数を使用。インテル® Core™ i7 プロセッサー上で 最大 3%、インテル® Core™2 プロセッサー 5300 番台と 5400 番台で最大 10% のパフォーマンスが向上。
- DGEMM パフォーマンスを強化するスレッド化アルゴリズムの向上
    - 8 スレッドで最大 7% の向上、3、5、7 スレッドで最大 50% の向上 (インテル® Core™ i7 プロセッサー)
    - 3 スレッドで最大 50% の向上 (インテル® Xeon® プロセッサー 7400 番台)
- 非素数サイズのスレッド化 1D 複素数-複素数 FFT
- 3D 複素数-複素数変換の新しいアルゴリズムにより 1 スレッドまたは 2 スレッドで小さな問題サイズ (最大 64x64x64) についてより優れたパフォーマンスを提供

- 対称正定行列の演算時におけるアウトオブコア (OOC) PARDISO のハイレベルな並列化実装
- すべての行列の型でインコアとアウトオブコアの両方で PARDISO のメモリー使用量が減少
- 実対称行列、複素エルミート行列、複素対称行列に対し PARDISO OOC で使用されるメモリーがインテル® MKL 10.1 で使用されていたメモリーの半分以下まで減少
- PARDISO/DSS における順序付けの並列化とシンボリック因子分解
- インテル® Core® i7 プロセッサーとインテル® Core™2 プロセッサーで次の VML 関数において最大 2 倍のパフォーマンスの向上 (平均で 30% の向上): v(s,d)Round、v(s,d)Inv、v(s,d)Div、v(s,d)Sqrt、v(s,d)Exp、v(s,d)Ln、v(s,d)Atan、v(s,d)Atan2
- インテル® Advanced Vector Extension (インテル® AVX) で次の関数の最適化バージョンが利用可能
    - BLAS: DGEMM
    - FFT
    - VML: exp、log、pow
    - 上記の関数にアクセスする `mkl_enable_instructions()` 関数に関する重要な情報については、インテル® MKL ユーザーズ・ガイドを参照してください。

6.2 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://software.intel.com/en-us/articles/intel-math-kernel-library-support-resources/ (英語) を参照してください。

6.3 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。

6.4 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。本リリースのインテル® MKL の一部の FFT 関数は、ヒューストン大学からライセンスを受けて、UHFFT ソフトウェア生成システム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。

7 インテル® スレッディング・ビルディング・ブロック

このセクションでは、インテル® C++ コンパイラー プロフェッショナル・エディションに同梱されているインテル® スレッディング・ビルディング・ブロック (インテル® TBB) の変更点、新機能、および最新情報をまとめています。

- インテル® C++ コンパイラー 10.x を glibc 2.3.2、2.3.3、または 2.3.4 とともに使用したときに、TBB アルゴリズムまたはコンテナーのコンテキストで実行されるユーザーコードで処理できない例外が発生すると、セグメンテーション違反が発生します。
- インテル® スレッド・チェッカーまたはインテル® スレッド・プロファイラーを使用した際により正確な結果を得るには、インテル® TBB とともに使用する前にそれらの製品の最新のアップデート・リリースをダウンロードしてください。
- 同じプログラムで連続してインテル® TBB と OpenMP コンストラクトをともに使用していて、OpenMP コードにインテル® コンパイラーを使用している場合、KMP_BLOCKTIME に小さな値 (例えば、20 ミリ秒) を設定するとパフォーマンスが向上します。この設定は、kmp_set_blocktime() ライブラリー呼び出しを使用して OpenMP コード内で行うこともできます。KMP_BLOCKTIME および kmp_set_blocktime() の詳細は、コンパイラーの OpenMP に関するドキュメントを参照してください。
- 一般に、アプリケーションやサンプルの非デバッグ ("リリース") ビルドは、インテル® TBB ライブラリーの非デバッグバージョンとリンクし、デバッグビルドはインテル® TBB ライブラリーのデバッグバージョンとリンクします。デバッグ・ライブラリーとリリース・ライブラリーの詳細については、製品ドキュメントのサブディレクトリーに含まれているチュートリアルを参照してください。

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。

MPEG-1、MPEG-2、MPEG-4、H.263、H.264、MP3、DV SD/25/50/100、VC-1、G.722.1、G.723.1A、G.726、G.728、G.729、GSM/AMR、GSM/FR、JPEG、JPEG 2000、Aurora、TwinVQ、AC3 および AAC は、ISO、IEC、ITU、SMPTE、ETSI およびその他の組織によって制定されている国際標準規格です。これらの標準規格の実装、または標準規格対応のプラットフォームの使用には、インテルを含むさまざまな組織からのライセンス許諾が必要になる場合があります。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国および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。