インテル® C++ コンパイラー 11.0 Windows* 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
2008.11.20

1 概要

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

1.1 製品の内容

インテル® C++ コンパイラー 11.0 Windows 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® C++ コンパイラー。Windows オペレーティング・システムを実行する IA-32、インテル® 64、および IA-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。
- IA-64 対応アプリケーション開発用インテル® アセンブラー
- 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® マス・カーネル・ライブラリー
- インテル® スレッディング・ビルディング・ブロック
- Microsoft* 開発環境への統合
- サンプルプログラム
- 各種ドキュメント

1.2 変更履歴

このセクションでは、バージョン 11.0 の最初のリリースから変更または追加された機能を示します。アップデート番号はリリースされたものとは異なる場合があります。示されたリビジョンまたはそれ以降のリビジョンでこれらの変更が含まれています。

- 11.0.069
  - IA-32 アーキテクチャー・システムのインストールでは、インストールするシステムが、最小要件であるインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) をサポートするプロセッサーを搭載しているかを検証します。
  - 報告された問題の修正
- 11.0.061
  - 最初のリリース

1.3 動作環境

1.3.1 アーキテクチャー用語

インテル® コンパイラーおよびライブラリーは、一般的なプロセッサー・アーキテクチャーとオペレーティング・システムを組み合わせた 3 つのプラットフォームをサポートしています。

このセクションでは、本ドキュメント、インストール手順、およびサポートサイトでプラットフォームの記述に使用されている用語について説明します。

IA-32 アーキテクチャー: 32 ビットのオペレーティング・システムを実行している、インテル® Pentium® II プロセッサーと互換性のある 32 ビット・プロセッサー (インテル® Pentium® 4 プロセッサー、インテル® Xeon® プロセッサーなど)、または同じ命令セットをサポートしている他社製のプロセッサーがベースのシステム。

インテル® 64 アーキテクチャー: 64 ビット・アーキテクチャーに対応するように拡張された IA-32 アーキテクチャー・プロセッサー (インテル® Core™2 プロセッサー・ファミリーなど) をベースとし、Microsoft Windows XP Professional x64 Edition または Microsoft Windows Vista* x64 Edition のような 64 ビット・オペレーティング・システムを実行するシステム。32 ビットの Windows オペレーティング・システムを実行しているシステムは、IA-32 とみなされます。Windows の 64 ビット・バージョンを実行する AMD* プロセッサー・ベースのシステムも、インテル® 64 対応アプリケーション開発用インテル® コンパイラーでサポートされています。

IA-64 アーキテクチャー: 64 ビット・オペレーティング・システムを実行している、インテル® Itanium® プロセッサー・ベースのシステム。

1.3.2 ネイティブおよびクロスプラットフォーム開発

「ネイティブ」とは、アプリケーションを実行するプラットフォームと同じプラットフォームでアプリケーションをビルドする (例えば、IA-32 システムで実行するアプリケーションを IA-32 システムでビルドする) ことを指します。「クロスプラットフォーム」または「クロスコンパイル」とは、アプリケーションを実行するプラットフォームとは異なる種類のプラットフォームでアプリケーションをビルドする (例えば、IA-64 システムで実行するアプリケーションを IA-32 システムでビルドする) ことを指します。すべての組み合わせのクロスプラットフォーム開発がサポートされているわけではありません。また、組み合わせによっては、オプションのツールとライブラリーをインストールする必要があります。

サポートされているホスト (アプリケーションをビルドするシステム) とターゲット (アプリケーションを実行するシステム) の組み合わせを次に示します。

ホスト＼ターゲット	IA-32	インテル® 64	IA-64
IA-32	○	○	○
インテル® 64	○	○	○
IA-64	X	X	○

1.3.3 最小動作環境

- 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 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、またはインテル® 64 アーキテクチャー・プロセッサーや 64 ビット AMD Athlon* プロセッサーまたは Opteron* プロセッサーをベースとするコンピューター、または IA-64 アーキテクチャー (インテル® Itanium) プロセッサーをベースとするコンピューター
- RAM 512MB (1GB 推奨)
- 2GB のディスク空き容量 (すべての機能およびすべてのアーキテクチャー)
- Microsoft Windows XP、Microsoft Windows Vista、Microsoft Windows Server* 2003、Microsoft Windows Server 2008、または Microsoft Windows HPC Server 2008 (エンベデッド・エディションはサポートされていません)

- 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のビルドに、Microsoft Visual Studio* 開発環境あるいはコマンドライン・ツールを使用する場合は、次のいずれか:
  - Microsoft Visual Studio 2008 Standard Edition 以降 (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
  - Microsoft Visual Studio 2005 Standard Edition 以降 (C++ と [X64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
- IA-32 対応アプリケーションのビルドに、Microsoft Visual Studio 開発環境またはコマンドライン・ツールを使用する場合は、次のいずれか:
  - Microsoft Visual Studio .NET 2003 (C++ コンポーネントがインストールされていること) [2]
  - Microsoft Visual C++* .NET 2003 [2]
- IA-64 対応アプリケーションのビルドに、Microsoft Visual Studio 開発環境またはコマンドライン・ツールを使用する場合は、次のいずれか:
  - Microsoft Visual Studio 2008 Team System Edition (C++ コンポーネントと [Itanium コンパイラおよびツール] コンポーネントがインストールされていること) [3]。さらに、Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
  - Microsoft Visual Studio 2005 Team System Edition (C++ コンポーネントと [Itanium コンパイラおよびツール] コンポーネントがインストールされていること) [3]
- IA-32 アーキテクチャー・アプリケーションのビルドに、コマンドライン・ツールのみを使用する場合は、次のいずれか:
  - Microsoft Visual C++ 2008 Express Edition
  - Microsoft Visual C++ 2005 Express Edition
- インテル® 64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合は、次のいずれか:
  - Microsoft Windows Server 2003 R2 Platform SDK
  - Microsoft Windows Software Development Kit Update for Windows Vista*
  - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- IA-64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合:
  - Microsoft Windows Server 2003 R2 Platform SDK
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

注:

1. Microsoft Visual Studio 2005/2008 Standard Edition では、[x64 コンパイラおよびツール] コンポーネントがデフォルトでインストールされます。Professional 以上のエディションでは、[カスタム] インストールが必要です。
2. Microsoft Visual Studio .NET 2003 は、Microsoft Windows Vista ではサポートされていません。本製品の将来のバージョンでは、Microsoft Visual Studio .NET 2003 は サポートされなくなる予定です。
3. IA-64 システムでは、Microsoft Visual Studio はサポートされていません。
4. アプリケーションは、上記の開発用と同じ Windows バージョンで実行できます。また、Windows XP よりも前の非エンベデッドの Microsoft Windows 32 ビット・バージョンでも実行できますが、インテルではこれらの互換性テストは行われていません。開発アプリケーションが、古いバージョンの Windows にはない Win32 API ルーチンを使用している可能性があります。アプリケーションの互換性テストをご自身の責任で行ってください。アプリケーションを実行するには、特定のランタイム DLL をターゲットシステムにコピーしなければならないことがあります。

1.3.4 Microsoft Visual Studio .NET 2003 のサポート終了予定

インテル® C++ コンパイラーの将来のメジャーリリースでは、Microsoft Visual Studio .NET 2003 のサポートは提供されなくなる予定です。インテルでは、現在のバージョンの Microsoft Visual Studio への移行を推奨しています。

1.4 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。

インストールを開始するには、製品 DVD を DVD ドライブに挿入します。自動でインストールが開始されます。自動で開始されない場合は、Windows エクスプローラで DVD ドライブのトップレベル・ディレクトリーを開き、`setup.exe` をダブルクリックします。

製品のダウンロード版を購入した場合は、ダウンロードしたファイル (`.EXE`) をダブルクリックして、インストールを開始します。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。

1.4.1 64 ビット・アプリケーション用の Visual Studio の設定

Microsoft Visual Studio 2005 または 2008 を使用し、64 ビット・アプリケーション (インテル® 64 または IA-64 アーキテクチャー向け) を開発する場合は、Visual Studio の構成を変更して、64 ビット・サポートを追加します。

Visual Studio 2005/2008 Standard Edition を使用する場合は、インテル® 64 対応アプリケーションのビルド用に構成を変更する必要はありません。その他のエディションの場合は、次の操作を行ってください。

1. [コントロール パネル] の [プログラムの追加と削除] から [Microsoft Visual Studio 2005 (または 2008)] を選択し、[変更と削除] をクリックします。[Visual Studio メンテナンス モード] ウィンドウが表示されます。[次へ] をクリックします。
2. [機能の追加と削除] をクリックします。
3. [選択した機能をインストールします] で [言語ツール] の [Visual C++] を展開します。
4. [x64 コンパイラおよびツール] ボックスがオンになっていない場合は、オンにし、[更新] をクリックします。ボックスがオンの場合は、[キャンセル] をクリックします。

Microsoft Visual Studio 2005/2008 Team System Edition を使用して、IA-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するには、上記の手順に従い、[Itanium コンパイラおよびツール] ボックスがオンになっていることを確認してください。

1.4.2 Microsoft Windows Vista でのインストール

Microsoft Windows Vista では、Microsoft Visual Studio .NET 2003 はサポートされていません。Microsoft Visual Studio 2005 ユーザーは、Visual Studio 2005 Service Pack 1 (VS 2005 SP1) と Visual Studio 2005 Service Pack 1 Update for Windows Vista (VS 2005 SP1 ページからリンクが提供) をインストールしてください。これらのアップデートをインストールした後に、管理者権限で Visual Studio が実行できることを確認してください。実行できない場合、インテル® コンパ

イラーを使用できません。詳細は、Microsoft の Visual Studio on Windows Vista (英語) および関連ドキュメントを参照してください。

Microsoft Visual Studio 2005 がインストールされている Microsoft Windows Vista にインストールする際、Visual Studio 2005 との互換性の問題がある旨を示す警告が表示されることがあります。この警告は、インストール・ウィンドウの後ろに隠れ、インストール処理がストールしているかのように見えることもあります。処理を続ける前に確認が必要なウインドウがないかどうかを Windows のタスクバーでチェックしてください。この警告では、プログラムの実行を行い、インストールを完了させることができます。インストールが終了したら、上記で説明されている 2 つの Service Pack 1 アップデートをインストールしてください。

1.4.3 既知のインストールの問題

以下のインストールの問題は、現在のバージョンにおける問題です。今後のアップデートでは修正される予定です。

- インテル® マス・カーネル・ライブラリーがインストールされている場合は、`include` フォルダーと `lib` フォルダーを [ツール] - [オプション] - [Intel(R) C++ (インテル(R) C++)] - [Compiler (コンパイラー)] ダイアログを使用して、Visual Studio に追加する必要があります。次に例を示します。
  `$(ICPP_COMPILER11)mkl\include`

1.5 製品の変更、更新、削除

製品を削除するには、Windows の [コントロール パネル] にある [プログラムの追加と削除] を使用します。

インストール済み製品のコンポーネントの追加と削除を行う場合は、現在インストールされているバージョンの製品セットアップ・プログラム (`setup.exe`) を再実行してください。Windows の [コントロール パネル] からは実行できません。製品のダウンロード版を購入した場合、デフォルトでは、セットアップ・プログラムは `C:\Program Files\Intel\Download\C++CompilerPro11.0` に展開されます。

製品のアップデート・バージョンをインストールする際、古いバージョンを最初にアンインストールする必要はありません。複数のバージョンのコンパイラーをインストールし、その中から選択して使用することができます。新しいバージョンのコンパイラーを削除した場合、以前のバージョンの Microsoft Visual Studio への統合を再インストールする必要があります。

1.6 インストール先フォルダー

11.0 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。

- `C:\Program Files\Intel\Compiler\11.0\`*`xxx`*`\cpp`
  - `bin`
    - `ia32`
    - `ia32_intel64`
    - `ia32_ia64`
    - `intel64`
    - `ia64`
  - `Documentation`
    - `compiler_c`
    - `en_US`

        - ipp
        - ja_JP
        - mkl
        - tbb
        - vshelp
    - include
    - ipp
        - em64t
        - ia32
        - ia64
    - lib
        - ia32
        - intel64
    - ia64mkl
        - benchmarks
        - em64t
        - examples
        - ia32
        - ia64
        - include
        - interfaces
        - tests
        - tools
    - perf_headers
    - Samples
    - tbb
        - em64t
        - examples
        - ia32
        - ia64
        - include
        - uninstall

xxx は 3 桁のアップデート番号です。bin、include、lib 配下のフォルダーは次のとおりです。

- ia32: IA-32 上で動作するアプリケーションのビルドに使用するファイル
- intel64 または em64t: インテル® 64 上で動作するアプリケーションのビルドに使用するファイル
- ia64: IA-64 上で動作するアプリケーションのビルドに使用するファイル
- ia32_intel64: IA-32 上での実行用のコンパイラー。インテル® 64 上で動作するアプリケーションをビルドします。
- ia32_ia64: IA-32 (またはインテル® 64) 上での実行用コンパイラー。IA-64 上で動作するアプリケーションをビルドします。

英語以外の Windows システムにインストールする場合、[Program Files] フォルダー名は異なります。インテル® 64 および IA-64 アーキテクチャー・システムでは、フォルダー名は [Program Files (X86)] またはそれに相当する名前です。

1.7 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。

1.8 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/cwin (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。

2.1 互換性

バージョン 11 では、IA-32 システムのデフォルトでのコード生成において、アプリケーションを実行するシステムでインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) がサポートされていると仮定するように変更されました。詳細は、下記を参照してください。

2.2 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++ lambda 関数
- 10 進浮動小数点
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- IPP オプションを使用した valarray の実装
- OpenMP* 3.0 のサポート
- Microsoft Visual Studio 2008 のサポート

2.3 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。

- /arch
- /bigobj
- /homeparams
- /MP
- /QaxSSE2
- /QaxSSE3
- /QaxSSSE3

- /QaxSSE4.1
- /QaxSSE4.2
- /Qdiag-error-limit
- /Qdiag-once
- /Qfast-trancendentals
- /Qfma
- /Qfp-relaxed
- /Qfreestanding
- /Qhelp-pragma
- /Qinstruction
- /Qm32
- /Qm64
- /Qopenmp-link
- /Qopenmp-task
- /Qopenmp-threadprivate
- /Qopt-block-factor
- /Qopt-jump-tables
- /Qopt-loadpair
- /Qopt-mod-versioning
- /Qopt-prefetch-initial-values
- /Qopt-prefetch-issue-excl-hint
- /Qopt-prefetch-next-iteration
- /Qopt-subscript-in-range
- /Qprof-data-order
- /Qprof-func-order
- /Qprof-gen
- /Qprof-hotness-threshold
- /Qprof-src-dir
- /Qprof-src-root
- /Qprof-src-root-cwd
- /Qprof-use
- /Qtcollect-filter
- /Qvc9
- /Qvec
- /QxHost
- /QxSSE2
- /QxSSE3
- /QxSSE3_ATOM
- /QxSSSE3
- /QxSSE4.1
- /QxSSE4.2
- /Werror-all

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。

2.3.1 /QxHost オプション

バージョン 11.0 から /QxHost オプションが新しく追加されました。このオプションは、ソースをコンパイルするシステム上のプロセッサーの種類に基づいて命令セットを自動的に選択します。動作は次のとおりです。

システムのプロセッサー	使用されるオプション
インテル® SSE2 以上の命令をサポートするインテル® プロセッサー	/QxSSE4.2、/QxSSE4.1、/QxSSSE3、/QxSSE3 または /QxSSE2 をプロセッサーに応じて使用
古いインテル® プロセッサー	/arch:ia32
インテル以外のプロセッサー	/arch:SSE3、/arch:SSE2 または /arch:ia32 をプロセッサーに応じて使用

命令セットオプションを使用する場合、指定した命令セットがアプリケーションを実行するシステムでサポートされていることを確認してください。アプリケーションのコンパイルと実行に同じシステムを使用する場合は、/QxHost オプションを使用することを推奨します。

2.4 新規および変更された Visual Studio 統合機能

Microsoft Visual Studio へのインテル® C++ 統合で次の項目が強化されています。

- .icproj プロジェクト・ファイルには、.vcproj ファイルに含まれていない補足情報のみが含まれるようになりました。これは、インテルと Microsoft のプロジェクト・システム間の同期が不要になったことを意味します。
- [ヘルプ] - [Microsoft Visual Studio のバージョン情報] に、インストールされたインテル® C++ コンパイラー・パッケージの識別子 (バージョン) についての情報が表示されます。
- [Floating Point (浮動小数点)] プロパティーの [Floating Point Model (浮動小数点モデル)] により、/fp コンパイラー・オプションのサポートが提供されます。
- [Optimization (最適化)] プロパティーの [Generate Alternate Code Paths (指定された命令セットの専用および汎用コード生成)] と [Use Intel(R) Processor Extensions (指定された命令セットの専用コード生成)] プロパティーは、変更された /Qx コンパイラー・オプションと /Qax コンパイラー・オプションに対応しています。
- 新しい [Optimization (最適化)] プロパティーの [Enable Enhanced Instruction Set (拡張命令セットを有効にする)] により、/arch コンパイラー・オプションのサポートが提供されます。
- 新しい [Optimization (最適化)] プロパティーの [Prefetch Insertion (プリフェッチ挿入)] により、/Qopt-prefetch コンパイラー・オプションのサポートが提供されます。
- プログラム全体の最適化をサポートするため、次のプロパティーが新規で追加または強化されました。
    - [Configuration (構成)] - [General (全般)] - [Whole Program Optimization (プログラム全体の最適化)]
    - [C++ (C++)] - [Optimization (最適化)] - [Interprocedural Optimization (プロシージャー間の最適化)]
    - [Linker (リンカー)] - [Optimization (最適化)] - [Whole Program Optimization (プログラム全体の最適化)]
- 新しい [C++ (C++)] - [Diagnostics (診断)] - [Optimization Diagnostics (最適化診断)] セクション (5 つのプロパティー) が定義されています。

- [Configuration (構成)] - [General (全般)] プロパティーの [Build Log File (ビルド・ログ・ファイル)] で、ビルドログの名前を変更できます。
- 未定義の変数が $(varname) として参照された場合、空の文字列が使用され、警告が表示されます。これは、Microsoft Visual C++ の動作と同じです。
- インテル® C++ プロジェクトで、Visual Studio の自動プロジェクト・バックアップ/復元機能を利用できるようになりました。
- [Manifest Tool (マニフェスト・ツール)] プロパティー・ページがインテル® C++ プロジェクトで使用できるようになりました。
- [Linker (リンカー)] (または [Librarian (ライブラリアン)]) - [General (全般)] プロパティーの [Link Library Dependencies (リンク・ライブラリーの依存関係)] がプロジェクトで使用できるようになりました。実行ファイルおよび DLL プロジェクトの場合、これは、依存ライブラリー・プロジェクトからの出力ライブラリーを自動的にリンクするかどうかを制御します。スタティック・ライブラリー・プロジェクトの場合、依存プロジェクトのスタティック・ライブラリーをビルド時に親ライブラリーにマージするかどうかを制御します。

2.4.1 インテル® C++ プロジェクト・ファイルの互換性

インテル® C++ プロジェクト・ファイル (`.icproj`) の形式がバージョン 11 で変更されました。インテル® C++ の古いバージョンで作成されたプロジェクトを開くと、プロジェクトの変換が必要である旨のメッセージが表示されます。バージョン 11 のプロジェクトを古いバージョンのインテル® C++ 統合で使用することはできません (ただし、古いバージョンのコンパイラーは、[Tools (ツール)] - [Options (オプション)] - [Intel C++ (インテル(R) C++)] - [Compilers (コンパイラー)] から使用できます)。

2.5 その他の変更および既知の問題

2.5.1 ビルド環境コマンドスクリプトの変更

ビルド環境を構築するコマンド・ウィンドウ・スクリプトが変更されました。ビルド環境ウィンドウを開くのに、定義済みのスタート・メニュー・ショートカットを使用していない場合は、次のコマンドを使用して適切な環境を構築してください。

```
"C:\Program Files\Intel\Compiler\11.0\xxx\cpp\Bin\iclvars.bat" argument
```

xxx は、アップデート番号です。*argument* は、`ia32`、`ia32_intel64`、`intel64`、`ia32_ia64`、`ia64` 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てください。

2.5.2 デフォルトの命令セットがインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) を必要とするものに変更

IA-32 アーキテクチャー向けのコンパイルでは、`/arch:SSE2` (旧: `/QxW`) がデフォルトになりました。`/arch:SSE2` でビルドされたプログラムは、インテル® Pentium® 4 プロセッサーや特定の AMD プロセッサーなど、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) をサポートするプロセッサー上で実行する必要があります。互換性を保証するランタイムチェックは行われません。プログラムが命令をサポートしていないプロセッサーで実行された場合は、無効な命令フォルトが発生する場合があります。これにより、インテル® SSE 命令が x87 命令の代わりに使用され、高い精度ではなく、宣言された精度で計算が行われることがあるため、浮動小数点結果が変更される可能性があることに注意してください。

すべてのインテル® 64 アーキテクチャー・プロセッサーでインテル® SSE2 がサポートされています。

汎用 IA-32 の以前のデフォルトを使用する場合は、`/arch:IA32` を指定してください。

2.5.3 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 Microsoft 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 では、デフォルトで互換ライブラリーが OpenMP アプリケーションで使用されます。`/Qopenmp-lib:compat` と等価です。古いライブラリーを使用する場合は、`/Qopenmp-lib:legacy` を指定してください。

"レガシー" ライブラリー (libguide.lib、libguide40.lib など) 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。

2.5.4 インテル® デバッガー (idb) の提供の停止

インテル® デバッガー (idb コマンド) は、インテル® コンパイラー製品の Windows 版では提供されなくなりました。現在もサポートされている Microsoft Visual Studio 内でデバッギングを行う場合には影響はありません。

2.5.5 サンプリング・ベースのプロファイルに基づく最適化機能の削除

ハードウェア・サンプリング・ベースのプロファイルに基づく最適化機能は提供されなくなりました。この変更に伴い、`/Qprof-gen-sampling` と `/Qssp` の 2 つのコンパイラー・オプション、および `profrun` と `pronto_tool` の 2 つの実行ファイルが削除されました。インストルメント形式のプロファイルに基づく最適化機能は従来どおり利用できます。

3 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® 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。

3.1 変更履歴

- 画像処理分野に ippiCopy* および ippiTranspose* 関数を実装
- 音声符号化および信号処理分野に新しい関数を実装 (詳細は、Documentation ディレクトリーに含まれている "NewFunctionsList.txt" を参照)
- 各種画像コーデックのプラグアンドプレイとしてインターフェイスを標準化する新しい UIC (unified image codec) フレームワークの実装
- インテル® Atom™ プロセッサーのサポート
- 高速データ圧縮ライブラリー lzo のサポートおよび zlib、gzip、bzip2 のパフォーマンスの向上
- インテル® IPP バイナリーと API を使用した画像処理の DMIP 遅延モード用の新しいサンプル
- 暗号化でインテル® QuickAssist API をサポート
- 新しい分野 - エラー訂正コーディング用の有限フィールドの演算に基づくデータ完全性関数
- 分野/機能の生成 (スパイラル)
- ビデオ拡張 - ノイズ除去/インターレース除去/モザイク除去
- 画像検索記述子 (MPEG7)、色レイアウト、エッジ・ヒストグラム
- Microsoft RT オーディオのサポート (拡張)
- 新しい音声符号化規格 G729.1 コーデックのサポート

- 高度な画像処理技術であるオプティカル・フローのサポート
- 復号用の新しいビデオ AVS コーデックのサポート
- 3D をサポートする新しい画像処理関数 Geom WarpAffine
- リードソロモン・アルゴリズム用の新しい暗号化関数のサポート
- スタティック・ライブラリーのスレッド化
- AAC 復号における ALS デコーダー・プロファイルのサポート

4 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。

4.1 変更履歴

- BLAS のパフォーマンスの向上
  - 32 ビット
    - インテル® Xeon® プロセッサー 5300 番台で (Z,C)GEMM が 40-50% 向上
    - インテル® Xeon® プロセッサー 5400 番台で GEMM コードが 10% 向上
  - 64 ビット
    - インテル® Xeon® プロセッサー 5400 番台の 1 つのスレッドで DGEMM が 2.5-3% 向上
    - インテル® Core™ i7 プロセッサー・ファミリーで SGEMM が 50% 向上
    - インテル® Core™ i7 プロセッサー・ファミリーの 1 つのスレッドで CGEMM が 3% 向上
    - インテル® Core™ i7 プロセッサー・ファミリーの 1 つのスレッドで ZGEMM が 2-3% 向上
    - インテル® Core™ i7 プロセッサー・ファミリーで DTRSM の右辺のケースが 30% 向上
- 直接法スパースソルバー (DSS/PARDISO) の改良
  - アウトオブコア PARDISO のパフォーマンスが平均 35% 向上しました。
  - DSS/PARDISO に個別の前方/後方置換のサポートが追加されました。
  - DSS インターフェイスに反復改善をオフにする新しいパラメーターが追加されました。
  - PARDISO インターフェイスにスパース行列構造を確認する新しいパラメーターが追加されました。
- コールバック関数メカニズムから長い計算の状況を追跡する機能と計算を中断する機能が追加されました。`mkl_progress` という関数をユーザー・アプリケーションで定義して、MKL LAPACK ルーチンのサブセットから呼び出すことができます。詳細は、『リファレンス・マニュアル』の「LAPACK 補助ルーチンとユーティリティー・ルーチン」の章を参照してください。この機能をサポートしている LAPACK 関数を確認するには、各関数の説明を参照してください。
- 転置関数がインテル® MKL に追加されました。詳細は、『リファレンス・マニュアル』を参照してください。
- C++ の std::complex 型を MKL 固有の複素数型の代わりに使用できるようになりました。
- Boost uBLAS 行列-行列乗算ルーチンの実装により、インテル® MKL BLAS で DGEMM の高度に最適化されたバージョンを使用できるようになりました。詳細は、『ユーザーズガイド』を参照してください。
- スパース BLAS の改良
  - すべてのデータ型 (単精度、複素数および倍精度複素数) のサポートが追加されました。

    - 圧縮スパース行形式で格納された 2 つのスパース行列の和と積を計算するルーチンが追加されました。
- ベクトル・マス・ライブラリー関数 CdfNorm、CdfNormInv、および ErfcInv が最適化され、パフォーマンスが向上しました。
- インテル® Core™ i7 プロセッサー・ファミリーにおけるパフォーマンスの向上
    - 次の VML 関数が 3-17% 向上: Asin、Asinh、Acos、Acosh、Atan、Atan2、Atanh、Cbrt、CIS、Cos、Cosh、Conj、Div、ErfInv、Exp、Hypot、Inv、InvCbrt、InvSqrt、Ln、Log10、MulByConj、Sin、SinCos、Sinh、Sqrt、Tanh。
    - 一様乱数生成が 7-67% 向上しました。
    - Wichmann-Hill、Sobol、および Niederreiter BRNG に基づく VSL 分布生成器が 3-10% 向上しました (64 ビットのみ)。
- 設定ファイルの機能が削除されました。インテル® MKL の動作を設定する代わりの方法については、『ユーザーズガイド』を参照してください。
- スタティック・ライブラリーから DLL 生成の制限がすべてなくなりました。
- インテル® MKL の関数が MPI プログラムから呼び出されると、デフォルトでは (明示的に制御されない限り) 1 つのスレッドで動作します。
- VML 関数 (CdfNorm、CdfNormInv および ErfcInv) が追加されました。
- DftiCopyDescriptor 関数が追加されました。
- DSS/PARDISO の LP64 インターフェイスは、64 ビット・オペレーティング・システム上で内部配列に 64 ビット・アドレッシングを使用するようになりました。この変更により、ソルバーでより大きな方程式を解くことができるようになりました。
- インテル® MKL のデフォルトの OpenMP ランタイム・ライブラリーが libguide から libiomp に変更されました。詳細は、Documentation ディレクトリーにある『ユーザーズガイド』を参照してください。
- ドキュメントの更新
    - ScaLAPACK をサポートする PBLAS (Parallel BLAS) の説明が『リファレンス・マニュアル』に追加されました。
    - Microsoft Visual Studio でのサンプルプログラムの使用に関する説明が『ユーザーズガイド』に追加されました。
    - MKL ドキュメントが Microsoft Visual Studio の [ヘルプ] メニューから F1 キーおよびコードエディターのダイナミック・ヘルプ機能を使用してアクセスできるようになりました。詳細は、『インテル® MKL ユーザーズガイド』を参照してください。
- インストール中に環境変数を設定できなくなりました。コマンドプロンプトで PATH、LIB および INCLUDE 環境変数を設定するには、tools\environment ディレクトリーにある 3 つのスクリプトファイル (mklvars32.bat、mklvarsem64t.bat および mklvars64.bat) を使用してください。
- インテル® Pentium® III プロセッサー用に最適化されたコードパスおよびこのプロセッサー固有のダイナミック・リンク・ライブラリーが削除されました。このプロセッサー上でインテル® MKL は引き続き使用できますが、デフォルトのコードパスが使用されるため、パフォーマンスは低下します。
- 区間線形ソルバー関数が削除されました。
- ドキュメントの更新
    - FFTW ラッパーの説明が製品パッケージから削除され、『リファレンス・マニュアル』の付録 G に統合されました。
    - 新しい関数の説明が『リファレンス・マニュアル』に追加され、Boost uBLAS 行列-行列乗算のサポートが『ユーザーズガイド』で説明されています。

4.2 既知の制限事項

インテル® MKL ユーザーズガイドの「Integrating a Microsoft Visual Studio* IDE Project with Intel MKL」(英語) セクションの内容は、本バージョンには適用されません。その前のセクションで説明されているように、インテル® MKL の Include フォルダーと Lib フォルダーのパスを、コンパイラーが使用するフォルダーリストに手動で追加する必要があります。

スパースソルバーと最適化ソルバーの制限事項

- スパースソルバーと最適化ソルバー・ライブラリー関数はスタティック形式でのみ提供されます。

FFT 関数の制限事項

- DFTI_TRANSPOSE モードは、デフォルトケースでのみ実装されます。
- DFTI_REAL_STORAGE モードにはデフォルト値のみ指定可能で、DftiSetValue 関数 (例えば、DFTI_REAL_STORAGE = DFTI_REAL_REAL) によって変更することはできません。
- ILP64 バージョンのインテル® MKL では、現在 1 つの次元の長さが 2^31-1 を超える FFT をサポートしていません。2^31-1 を超える 1D FFT、またはいずれかの次元が 2^31-1 を超える多次元 FFT では、"DFTI_1D_LENGTH_EXCEEDS_INT32" エラーメッセージが返されます。この制限は、各次元の長さが 2^31-1 を超えない限り、2^31-1 個を超える成分を持つ多次元 FFT には適用されないことに注意してください。
- クラスター FFT 関数の配列サイズでいくつかの制限があります。詳細は、『リファレンス・マニュアル』 (mklman.pdf) を参照してください。
- 動的にリンクされているアプリケーションでクラスター FFT 関数を使用する場合、インテル® MKL のスタティック・インターフェイス・ライブラリーもリンクする必要があります。
  例: -Wl,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -Wl,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread

LAPACK 関数の制限事項

- ILAENV 関数 (ローカル環境の問題依存パラメーターを選択するために LAPACK ルーチンから呼び出される) は、ユーザーのバージョンでは代用できません。
- CPU の周波数が一定でない場合、second() および dsecnd() 関数は正しくない結果を返すことがあります。
- バージョン 10.0 以降、ダイナミック・ライブラリーにリンクする場合、次の 2 つの問題があります。
  - LAPACK が不正な入力パラメーターを使用して呼び出された場合、ユーザーが提供した XERBLA は使用されません。代わりに、デフォルトの XERBLA が使用されます。

    - LAPACK が不正なパラメーターを使用して LP64 インターフェイスから呼び出された場合、セグメンテーション違反が発生します。原因は、負のメモリー量の割り当てが行われるためです。

ベクトル・マス・ライブラリー (VML) 関数と ベクトル・スタティスティカル・ライブラリー (VSL) 関数の制限事項

- mkl_vml.fi を使用すると、TYPE ERROR_STRUCTURE 長に関する警告が生成されることがあります。
- インテル® MKL 関数への参照を含むカスタム DLL をビルドする必要がある場合、インテル® MKL DLL ビルダーツールを使用します。その他の DLL ビルド機能はサポートされていません。

ScaLAPACK 関数の制限事項

- PJLAENV 関数はユーザーのバージョンでは代用できません。この関数は、ローカル環境の問題依存パラメーターを選択するために ScaLAPACK ルーチンから呼び出されます。
- MPICH2 で -genvlist を使用して MKL_BLACS_MPI などのグローバル環境変数を取得する際に、問題が発生することがあります。この場合、[コントロール パネル] から必要な環境変数を設定してください。[スタート] メニューから [設定] - [コントロール パネル] - [システム] - [詳細設定] を選択して、[環境変数] ボタンをクリックします。

ILP64 バージョンのインテル® MKL における制限事項:

- ILP64 バージョンのインテル® MKL には、ライブラリーのすべての機能は含まれていません。ILP64 バージョンに含まれる機能の一覧は、Documentation ディレクトリーにある『ユーザーズガイド』を参照してください。

Java サンプルにおける制限事項

- JDK のパスにスペースが含まれている場合、Java サンプルは動作しません。この場合、引用符を使用して JAVA_HOME を設定してください。
  例: set JAVA_HOME="C:\Program Files\Java\jdk1.6.0_06"

MP LINPACK のハイブリッド・バージョンをビルドするときに DHPL_CALL_CBLAS オプションは使用できません。

インテル® コンパイラーでインテル® MKL のサンプル・ソース・コードをコンパイルする場合は、/Od オプションを使用することを推奨します。現在のビルドスクリプトでは、このオプションは指定されません。また、インテル® コンパイラーでは、デフォルトでベクトル化を行うように変更されました。

ダミー・ライブラリーの制限事項

- ダミー・ライブラリーは、#pragma 構造体では使用できません。ダミー・ライブラリーは、インテル® コンパイラーからドライバーとしてリンクできません。詳細は、『ユーザーズガイド』の第 3 章を参照してください。

VSL 関数はすべて、エラーステータスを返します。例えば、VSL API のデフォルトは、以前のバージョンのインテル® MKL ではサブルーチン形式でしたが、現在では関数形式です。つまり、Fortran のユーザーは、VSL ルーチンを関数として呼び出す必要があります。

関数の呼び出し例:
    errstatus = vslrnggaussian(method, stream, n, r, a, sigma)
サブルーチンの呼び出し例:
    call vslrnggaussian(method, stream, n, r, a, sigma)
ただし、インテル® MKL では、下位互換用にサブルーチン形式のインターフェイスも用意しています。サブルーチン形式のインターフェイスを使用するには、手動で (include ディレクトリーにある) mkl.fi ファイルの include 'mkl_vsl.fi' という行を include 'mkl_vsl_subroutine.fi' に変更し、mkl_vsl.fi ファイルの代わりに mkl_vsl_subroutine.fi ファイルを組み込みます。VSL の API 変更は、C/C++ ユーザーには影響しません。

メモリー割り当て: より高いパフォーマンスを得るため、インテル® MKL によって割り当てられたメモリーは解放されません。これは仕様で、インテル® MKL ルーチンがメモリーバッファーを操作するのは 1 回 (割り当て) だけです。ツールによっては、これをメモリーリークとして報告することがあるため、注意してください。必要に応じて、メモリーを解放することができます。プログラムでインテル® MKL の MKL_FreeBuffers() 関数を使用するか、各呼び出しの後に MKL_DISABLE_FAST_MM 環境変数を設定します (詳細は、Documentation ディレクトリーにある『ユーザーズガイド』を参照してください)。しかし、これらの方法を使用してメモリーを解放しても、メモリーリークが報告されなくなるとは限りません。実際、ライブラリーを複数回呼び出す場合、各呼び出しごとに新しいメモリーの割り当てが必要になり、報告される数は増えることもあります。上記の方法で解放されなかったメモリーは、プログラムの終了時にシステムによって解放されます。この制限を回避するには、上記のようにメモリー管理を無効にします。

その他: GMP コンポーネントはソルバー・ライブラリーにあります。インテル® 64 および IA-64 プラットフォームでは、これらのコンポーネントは LP64 インターフェイスのみをサポートします。

マルチスレッドのインテル® MKL をリンクするときは /MT を使用することを推奨します。Microsoft Visual C++ .NET 2003 で /MD を使用すると、リンクエラーが発生します。

5 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® 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。

- atomic<long long> および atomic<unsigned long long> テンプレートは、Microsoft Visual C++ 7.1 (Microsoft Visual Studio .NET 2003) コンパイラーではサポートされていません。
- 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。 Windows システムでは、/MD オプションを使用してコンパイルした場合はインテル® TBB ライブラリーのリリース・ライブ

ラリー、/MDd オプションを使用してコンパイルした場合はデバッグ・ライブラリーを使ってビルドしてください。他の組み合わせでは、ランタイム・エラーが発生します。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、"Documentation\tbb" 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。

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、商品性に関する保証、第三者の特許権、著作権、その他知的財産権の侵害への保証を含む) にも一切応じないものとします。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。そのようなエラッタは、インテルの保証範囲外です。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 Atom、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。